

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет комп'ютерних наук і технологій

(повне найменування інституту, назва факультету)

Кафедра програмних засобів

(повне найменування кафедри)

Пояснювальна записка

до дипломного проєкту (роботи)

магістр

(ступінь вищої освіти)

на тему ДОСЛІДЖЕННЯ ТА ПРОГРАМНА РЕАЛІЗАЦІЯ МЕТОДІВ ОБЛІКУ
РОБОЧОГО ЧАСУ НА ПІДПРИЄМСТВАХ
RESEARCH AND SOFTWARE IMPLEMENTATION OF WORKING TIME
TRACKING METHODS AT ENTERPRISES

Виконав: студент(ка) 2 курсу, групи КНТ-111м
Спеціальності 121 Інженерія програмного
забезпечення

(код і найменування спеціальності)

Освітня програма (спеціалізація)

Інженерія програмного забезпечення

Мартиненко Г.Ю.

(прізвище та ініціали)

Керівник Каплієнко Т.І.

(прізвище та ініціали)

Рецензент Шитікова О.В.

(прізвище та ініціали)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»
(повне найменування закладу вищої освіти)

Інститут, факультет ФКНТ
Кафедра програмних засобів
Ступінь вищої освіти магістр
Спеціальність 121 Інженерія програмного забезпечення
(код і найменування)
Освітня програма (спеціалізація) Інженерія програмного забезпечення
(назва освітньої програми (спеціалізації))

ЗАТВЕРДЖУЮ

Завідувач кафедри ПЗ, д.т.н, проф.
С.О. Субботін
“ ” 2022 року

ЗАВДАННЯ
НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА(КИ)

Мартиненко Ганни Юріївни

(прізвище, ім'я, по батькові)

1. Тема проєкту (роботи) Дослідження та програмна реалізація методів обліку робочого часу на підприємствах. Research and Software Implementation of Working Time Tracking Methods at Enterprises

керівник проєкту (роботи) Каплієнко Тетяна Ігорівна, к.т.н., доцент
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом закладу вищої освіти від “04” листопада 2022 року № 363

2. Строк подання студентом проєкту (роботи) 01 грудня 2022

3. Вихідні дані до проєкту (роботи) рекомендована література, технічне завдання

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1. Аналіз проблеми та постановка завдань дослідження. 2. Матеріали і методи. 3. Опис програми. 4. Розробка програми. 5. Тестування та експериментальне дослідження програми. 6. Додатки.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень) Слайди презентації

6. Консультанти розділів проєкту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-5 Основна частина	Каплієнко Т.І., доцент		
Нормоконтролер	Белова А.В., асистент		

7. Дата видачі завдання “ 04 ” вересня 2022 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1	Постановка завдання роботи.	1 тиждень	Завдання
2	Аналіз проблеми та постановка завдань дослідження.	2 тиждень	Розділ 1
3	Матеріали і методи.	3 тиждень	Розділ 2
4	Опис програми.	4 тиждень	Розділ 3
5	Розробка програми.	5-6 тижні	Розділ 4
6	Тестування та експериментальне дослідження програмного забезпечення.	7 тиждень	Розділ 5
7	Оформлення пояснювальної записки та документів до неї.	8-9 тижні	Додатки
8	Нормоконтроль та рецензування.	10–11 тижні	
9	Захист роботи.	12 тиждень	

Студент(ка)

_____ **Мартиненко Г.Ю.**
(підпис) (прізвище та ініціали)

Керівник проєкту (роботи)

_____ **Каплієнко Т.І.**
(підпис) (прізвище та ініціали)

РЕФЕРАТ

Пояснювальна записка до кваліфікаційної роботи магістра: 175 с., 12 табл., 50 рис., 2 дод., 23 джерела.

ЗАСТОСУНОК, ОБЛІК РОБОЧОГО ЧАСУ, ПЕРЕПУСТКА, ANDROID, API, JAVA QR-КОД, RFID.

Об'єкт дослідження – методи обліку робочого часу підприємства.

Предмет дослідження – комп'ютерні програми для обліку робочого часу працівників на підприємстві.

Мета роботи – створення програмної системи для автоматизації та прискорення процесу обліку робочого часу нормувальниками підприємства.

Матеріали, методи та технічні засоби: структурне програмування, мова програмування Java, пристрій, на який можливо встановлення Android-застосунку.

Результати. Створено програмний засіб, що встановлюється на Android-телефони, використовує під час роботи методи RFID та QR зчитування, автоматизує та систематизує облік робочого часу працівників для нормувальників підприємства.

Висновки. Розроблено програмний засіб для обліку робочого часу працівників підприємства, що прискорює, автоматизує та систематизує цей процес.

Галузь використання – відділ кадрів або відділ нормування праці, великих корпорацій або середніх масштабів компанії.

ABSTRACT

Explanatory note to the diploma qualifying work of the master: 175 pages, 12 tables, 50 figures, 2 appendices, 23 sources.

APP, TIMETABLE, PERMIT, ANDROID, API, JAVA QR CODE, RFID.

The object of the study is methods of accounting for the company's working time.

The subject of the research is computer programs for recording the working hours of employees at the enterprise.

The purpose of the work is to create a software system for automating and accelerating the process of working hours accounting by the company's raters.

Materials, methods and technical means: structural programming, Java programming language, a device on which the Android application can be installed.

The results. A software tool has been created which can be installed on Android phones, uses RFID and QR reading methods during operation, automates and systematizes the accounting of employees' working hours for the company's raters.

Conclusions. A software tool has been developed for recording the working time of the company's employees, which accelerates, automates and systematizes this process.

The field of use is the personnel department or the department of labor regulation, large corporations or medium-sized companies.

ЗМІСТ

	С.
ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК	8
ВСТУП.....	9
1 АНАЛІЗ ПРОБЛЕМИ ТА ПОСТАНОВКА ЗАВДАНЬ ДОСЛІДЖЕННЯ...	11
1.1 Поняття DCS	11
1.2 Застосування DCS в бізнесі	12
1.2.1 СКУД.....	12
1.2.2 Приклад реалізації системи трекінгу.....	13
1.2 Постановка завдання роботи.....	14
2 МАТЕРІАЛИ І МЕТОДИ.....	16
2.1 RFID	16
2.2 QR.....	22
2.3 Концепція системи	25
3 ОПИС ПРОГРАМИ.....	31
3.1 Порівняльний аналіз мов програмування	31
3.2 Опитування зацікавлених осіб	33
3.3 Архітектура програми	35
3.4 Висновки з розділу 3	43
4 РОЗРОБКА ПРОГРАМИ	44
4.1 Файли, методи та функції	44
4.2 Інтерфейс	50
5 ТЕСТУВАННЯ ТА ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ.....	71
5.1 Опис застосування.....	71
5.2 Керівництво користувача.....	71
5.3 Тестування	74
5.4 Дослідно-промислова експлуатація	76
ВИСНОВКИ	78
ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ	80
ДОДАТОК А Текст програми	83

ДОДАТОК Б Слайди презентації	166
------------------------------------	-----

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

activity	– вікно з елементами керування застосунку;
API	– Application Programming Interface;
Android	– платформа для керування пристроями;
DCS	– Data collection system, система збору даних;
IDE	– Integrated Development Environment;
INTERFACE	– об'єкт взаємодії користувача та програми;
Internet	– всесвітня система взаємополучених комп.мереж;
HR	– Human Resources;
застосунок	– програма для пристроїв під керуванням Android;
ІТ	– інформаційні технології;
БД	– база даних;
ВС	– вбудована система;
ПЗ	– програмне забезпечення;
ПП	– програмний продукт;
СКУД	– Система Керування та Управління Доступом.

ВСТУП

На даний момент на підприємствах найчастіше функціонують відділи нормування праці та відділи кадрів або відділи HR. З ростом кількості співробітників підприємства збільшується навантаження на працівників цих відділів [1].

Приблизно 70-90% робочого часу нормувальники витрачають на систематизацію та облік даних з обсягу відпрацьованого часу інших працівників [1].

В цьому, власне, і полягає їх робота, але часто трапляється так, що нормувальник з затримками володіє інформацією про робочі часи працівників або не володіє взагалі внаслідок географічних, технічних та інших перешкод [1].

Для компанії, що виконує роботи на різних об'єктах навіть одного міста, постає проблема в кількості нормувальників, що дорівнює кількості місць, де виконується роботи для ефективного обліку робочого часу [1].

Можна використовувати турнікети, систему перепусток та систему керування та управління доступом (СКУД), але у випадку короткочасного виконання робіт та великої кількості об'єктів це рішення не є економічно доцільним [1].

Найчастіше роботи виконуються на базі інших підприємств, працівники отримують перепустки на відповідні об'єкти. З міркувань безпеки та надійності отримання даних СКУД інших підприємств на об'єктах не є вирішенням проблеми [1].

У разі надзвичайної ситуації на виробництві нормувальник має достеменно знати, чи перебував той чи інший працівник на відповідному об'єкті у певний проміжок часу, оскільки це несе фінансові, репутаційні та інші наслідки для компанії в залежності від ступеню критичності ситуації. Отже для нормувальників важливо володіти актуальною інформацією щодо задіяності працівників на об'єктах та обсягів робочого часу працівників [1].

Існує декілька варіантів вирішення даної проблеми. Підприємство або розширює штат нормувальників, які збирають та оброблюють інформацію, або автоматизує обробку певної частини інформації. Економічно доцільніше одноразове розроблення програмного продукту, ніж постійне утримання додаткової робочої сили [1].

Перед обранням технологій та засобів розробки важливо розуміти перелік необхідного функціоналу майбутнього програмного продукту. Коли сформовано ціль розробки, варто також врахувати технічні особливості місць використання (стабільність доступу до мережі Інтернет, тощо) [1].

Найчастіше автоматизується інформаційна функція (отримання інформації в режимі реального часу) та систематизація великих обсягів даних. Також важливо забезпечити збереження інформації в умовах нестабільного доступу до мережі Internet та достовірність отриманих даних [1].

Також було вирішено реалізувати для працівників доступ до актуальної дозвільної документації для можливості надання останньої для перевірок, які відбуваються на підприємстві [1].

Для реалізації програмного продукту доцільно використовувати звичні пересічному користувачу засоби взаємодії. Ідеально підходить варіант телефона під керуванням Android, оскільки важко знайти співробітника, який не користується останнім, також таким чином буде дотримано вимогу про достовірність даних, що отримуються, оскільки так можна виключити людський фактор та факти шахрайства з датами та часом [1].

1 АНАЛІЗ ПРОБЛЕМИ ТА ПОСТАНОВКА ЗАВДАНЬ ДОСЛІДЖЕННЯ

1.1 Поняття DCS

Система збору даних (DCS – Data Control System) є комп'ютерним застосунком, що полегшує процес збір даних, дозволяючи систематично збирати конкретну структуровану інформацію, надаючи можливість аналізу даних [2]-[4]. Зазвичай DCS відображає форму, яка приймає введення даних від користувача, а потім перевіряє цей ввід до передачі даних у постійне сховище, наприклад у базу даних.

Багато комп'ютерних систем реалізують форми введення даних, але системи збору даних, як правило, є більш складними, і, можливо, багато пов'язаних форм містять докладні поля введення даних користувачем, перевірку даних та навігаційні посилання серед форм.

DCS можна вважати спеціалізованою формою системи управління контентом (CMS – Content Management System), особливо коли вони дозволяють оприлюднювати інформацію, що збирається, редагувати, модифікувати, видаляти та обробляти інформацію.

Деякі системи загального призначення містять функції DCS [5], [6].

Як правило, комп'ютерне програмне забезпечення, що використовується для збору даних, потрапляє в одну з наступних категорій практичного застосування [7].

- опитування або анкети [8], [9];
- реєстри даних [10], [11];
- система управління справами у юридичній галузі [12];
- вимірювання продуктивності системи для підвищення ефективності [13], [14];
- іспити та вікторини [15], [16];
- інтернет-форми та системи подання звітності.

Програмне забезпечення типу DCS є широко застосованим у багатьох галузях, оскільки є економічно вигідним, бо автоматизує найбільш

трудомісткий процес – збору інформації, та частково виключає людський фактор.

1.2 Застосування DCS в бізнесі

Найбільш економічно доцільним є використання DCS систем для методичного збору інформації, це підтверджує розповсюдженість данного методу.

У комерційних проєктах дана методика використовується досить часто, різниця полягає саме у реалізації прикладного та програмного прошарку системи в залежності від потреб та можливостей підприємств.

На підприємствах міста Запоріжжя, для збору інформації про робочий час та її структуризації використовується людська праця лише в одному випадку з двадцяти. В інших дев'ятнадцяти випадках в певній мірі цей процес автоматизовано.

Для більшості підприємств з статичним місцезнаходженням ідеально підходить варіант СКУД, оскільки закриває потреби у зборі та передачі даних, але для невеликих компаній, що виконують проєктні роботи у різних містах, варіант СКУД не є прийнятним. У такому випадку підходить варіант трекінгу працівників певними засобами.

Основними перевагами трекінгу перед СКУД є мобільність, економічна доцільність та пристосованість.

Нижче наведено опис прикладів реалізації збору даних про робочий час працівників на підприємствах.

1.2.1 СКУД

СКУД (Система Контролю та Управління Доступом) - найбільш розповсюджена система контролю робочого часу, використовується на 75% підприємств міста Запоріжжя.

Система являє собою встановлені турнікети, що відкривають доступ до підприємства шляхом зчитування електронної індивідуальної перепустки. Таким чином фіксується умовний час початку та завершення роботи працівника.

У таблиці 1.1 наведено основні переваги та недоліки реалізації СКУД.

Таблиця 1.1 – Основні переваги та недоліки реалізації СКУД

Переваги	Недоліки
Достовірність інформації	Статичність
Незалежність від погодних умов	Висока собівартість

Для розпізнавання перепустки використовуються зчитувачі спільної з перепустками частоти. Технологія зчитування – магнітна стрічка на перепустці.

Також зустрічаються варіанти RFID зчитувачів різної частоти, перепустка у такому варіанті являє собою аналог банківської картки з чипом для дистанційного зчитування.

1.2.2 Приклад реалізації системи трекінгу

Нижче наведено опис однієї з реалізацій системи трекінгу та обліку робочого часу на одному з підприємств України.

Кожен працівник має фітнес-годинник, кожна площадка, де виконуються роботи, має зчитувач. Працівник, наближаючись на певну відстань до зчитувача, ініціює зчитування даних фітнес-годинника, таким чином фіксується умовний час початку робочого дня.

У таблиці 1.2 наведено основні переваги та недоліки реалізації системи трекінгу.

Таблиця 1.2 – Основні переваги та недоліки реалізації системи трекінгу

Переваги	Недоліки
Мобільність системи	Залежність від погодних умов
Сучасність технології	Залежність від висот місцевості
Достовірність інформації	Висока собівартість
	Високий поріг навичок для впровадження

Серед переваг даного методу є надійність даних, оскільки працівник насправді перебуває у зазначеному місці.

Серед недоліків, технічно неподоланим бар'єром є різниця географічних висот зчитування та радіо перешкоди сигналу самого підприємства при відправці даних зі зчитувачів на сервер. Також така реалізація потребує значних витрат, бо вимагає забезпечення всього робочого персоналу фітнес-годинниками.

Наразі не існує методу контролю робочого часу працівників, що задовільнить потреби у мобільності, середній собівартості, невеликому обсягу навичок для користування.

Аналогів системи, що проєктується, не існує, тому прийнято рішення про актуальність розробки даного програмного продукту.

1.2 Постановка завдання роботи

Для ведення обліку робочого часу працівників необхідно розробити систему збору та передачі даних.

Найменування продукту – застосунок «Trackers».

Програмний продукт (ПП) призначений для збору даних про робочий час працівників. Область застосування – підприємства, що виконують роботи на географічно розподілених об'єктах.

Дана розробка дає можливість користувачу зафіксувати початок та завершення робочої зміни. Також користувач має доступ до інформації про співробітників, що задіяні на поточній ділянці, такої як час початку та завершення робочої зміни, дозвільні документи на виконання певних видів робіт, документи про освіту та ін..

2 МАТЕРІАЛИ І МЕТОДИ

2.1 RFID

RFID (англ. Radio frequency identification) — це радіочастотна ідентифікація. Радіочастотне розпізнавання здійснюється за допомогою закріплених за об'єктом спеціальних міток, які несуть ідентифікаційну та іншу інформацію [17].

Цей метод вже став основою побудови сучасних безконтактних інформаційних систем, і має стійку назву RFID-технології [17].

Особливості технології:

- RFID-мітки не потребують контакту або прямої видимості, дані з мітки можуть бути отримані на відстані;
- RFID-мітки мають високу швидкість і точність читання, що дозволяє виконувати велику кількість сканувань;
- RFID-мітки можна використовувати навіть в агресивних середовищах (бруд, фарба, пар, вода, пластмаса, деревина);
- пасивні RFID-мітки мають фактично необмежений термін експлуатації та низьку собівартість;
- RFID-мітки можуть нести певну кількість інформації [17].

Стисла історія RFID міток наведена нижче. У 1946 році Лев Сергійович Термен винайшов шпигунський пристрій для Радянського Союзу, який дозволив накладати аудіоінформацію на випадкові радіохвилі. Звук викликав коливання діафрагми, яка трохи змінювала форму резонатора, модулюючи відбиту радіочастотну хвилю. І хоча пристрій представляв лише пасивний передавач (т.з. «жучок»), цей винахід зараховують до перших попередників RFID-технології [17].

Технологія, найближча до даної — система розпізнавання «свій-чужий», винайдена британцями в 1939 році. Вона активно застосовувалася союзниками під час Другої Світової Війни, щоб визначити, своїм або чужим є об'єкт в небі. Подібні системи досі використовуються як у військовій, так і

в цивільній авіації. Ще однією віхою у використанні RFID-технології є робота Гарі Стокмана (Harry Stockman) під назвою «Комунікації за допомогою відбитого сигналу» (англ. «Communication by Means of Reflected Power») (доповіді IRE, стор. 1196–1204, жовтень 1948). Стокман відзначає, що «значні роботи з дослідження й розробки були зроблені до того, як були вирішені основні проблеми в зв'язку за допомогою відбитого сигналу, а також до того, як були знайдені сфери застосування даної технології» [17].

Патент США Маріо Кардулло (Mario Cardullo) № 3,713,148 від 1973 («Пасивний радіопередавач з пам'яттю»), був, по суті, прабатьком сучасної RFID-технології. Вперше пасивний пристрій на відбитій енергії був продемонстрований в 1971 році владі Нью-Йорка і іншим потенційним покупцям як пристрій з 16 бітами пам'яті для стягування мита на дорогах. Патент Кардулло покриває використання радіохвиль, світла і звуку як засіб передачі інформації [17].

Оригінальний бізнес-план був представлений інвесторам в 1969 для використання у сфері транспорту (ідентифікація самохідних машин, автоматична платіжна система (система стягування мита), електронні номерні знаки, електронні платіжні відомості, водіння машин, моніторинг стану транспортних засобів), в банківській справі (електронні книги перевірок, електронні кредитні картки), у сфері безпеки (персональна ідентифікація, автоматичні ворота, спостереження) і в медицині (ідентифікація пацієнта, історії хвороби) [17].

Перша демонстрація сучасних RFID-чипів (на ефекті зворотного розсіювання), як пасивних, так і активних, була проведена в Дослідницькій Лабораторії Лос Аламоса (англ. Los Alamos Scientific Laboratory) в 1973 році. Портативна система працювала на частоті 915 МГц і використовувала 12 бітових міток [17].

Перший патент, пов'язаний власне з назвою RFID, був виданий Чарльзу Уолтону (Charles Walton) в 1983 році (патент США за № 4,384,288) [17].

Класифікація RFID-міток. Існує декілька способів систематизації RFID-міток і систем:

- за робочою частотою;
- за джерелом живлення;
- за типом пам'яті [17].

За типом джерела живлення RFID-мітки діляться на:

- пасивні;
- активні;
- напівпасивні [17].

Пасивні RFID-мітки не мають вбудованого джерела енергії. Електричний струм, що індукується в антені електромагнітним сигналом від зчитувача, забезпечує достатню потужність для функціонування кремнієвого CMOS-чипа, розміщеного в мітці, і передачі у відповідь сигналу [17].

Комерційні реалізації низькочастотних RFID-міток можуть бути вбудовані в стикер (наклейку) [17].

У 2006 Hitachi виготовила пасивний пристрій, названий μ -Chip (мю-чип), розмірами $0,15 \times 0,15$ мм (не включаючи антену) і тонше за паперовий лист (7.5 мкм) [17].

Напівпасивні RFID-мітки дуже схожі на пасивні мітки, але оснащені батареєю, яка забезпечує чип енергоживленням. При цьому дальність дії цих міток залежить тільки від чутливості приймача зчитувача і вони можуть функціонувати на більшій відстані і з кращими характеристиками [17].

За типом використовуваної пам'яті RFID-мітки діляться на:

- RO (англ. Read Only) – дані записуються тільки один раз, відразу при виготовленні. Такі мітки придатні тільки для ідентифікації, ніяку нову інформацію в них записати не можна, і їх практично неможливо підробляти;
- WORM (англ. Write Once Read Many) – окрім унікального ідентифікатора такі мітки містять блок одноразово записуваної пам'яті, яку надалі можна багато разів читати;

– RW (англ. Read and Write) – такі мітки містять ідентифікатор і блок пам'яті для читання/запису інформації, дані в них можуть бути перезаписані багаторазово [17].

Зчитувачі – прилади, які читають інформацію з міток і записують в них дані. Ці пристрої можуть бути постійно підключеними до облікової системи або працювати автономно. Залежно від частотного діапазону мітки, дистанція стійкого зчитування і запису даних може бути різною. Розрізняють стаціонарні та мобільні [17].

Стаціонарні зчитувачі кріпляться нерухомо на стінах, дверях, рухомих складських пристроях (штабеляторах, навантажувачах). Вони можуть бути виконані у вигляді замку, вмонтовані в стіл або закріплені поряд з конвеєром на шляху проходження виробів [18].

В порівнянні з мобільними, зчитувачі такого типу зазвичай мають більшу зону читання та потужність, і здатні одночасно обробляти значний потік інформації. Стаціонарні зчитувачі на виробництві інтегруються в інформаційну систему, що дозволяє поетапно фіксувати переміщення маркованих об'єктів в реальному часі, або ідентифікувати положення мічених предметів в просторі [18].

Мобільні – володіють порівняно меншою дальністю дії і часто не мають постійного зв'язку з програмою контролю і обліку. Мобільні зчитувачі мають внутрішню пам'ять, в яку записуються дані з прочитаних міток (потім цю інформацію можна синхронізувати з системою обліку) і, як і стаціонарні зчитувачі, здатні записувати дані в мітку (наприклад, інформацію про проведений контроль) [18].

Основні переваги RFID-технології:

- для RFID не потрібний контакт або пряма видимість;
- RFID-мітки читаються швидко і точно (наближаючись до 100%-вої ідентифікації);

- RFID може використовуватися навіть в агресивних середовищах, а RFID-мітки можуть читатися через бруд, фарбу, пар, воду, пластмасу, деревину;
- пасивні RFID-мітки мають фактично необмежений термін експлуатації;
- RFID-мітки несуть велику кількість інформації і можуть бути інтелектуальними;
- RFID-мітки можуть бути не тільки для читання, але і з записом інформації [18].

RFID-системи застосовуються в різноманітних випадках, коли потрібен оперативний і точний контроль, відстеження і урахування численних переміщень різноманітних об'єктів. Типові застосування:

- електронний контроль за доступом і переміщеннями персоналу на території підприємств;
- керування виробництвом, товарними і митними складами (особливо значними), магазинами, видачею і переміщенням товарів і матеріальних цінностей;
- автоматичний збір даних і при необхідності нарахування оплати на залізницях, платних автомобільних дорогах, на вантажних станціях і терміналах;
- контроль, планування і керування рухом, інтенсивністю графіка і вибором оптимальних маршрутів;
- громадський транспорт — керування рухом, оплата проїзду й оптимізація пасажиропотоків;
- системи електронних платежів для усіх видів транспорту, включаючи організацію платних доріг, автоматичний збір плати за проїзд і транзит, платні автостоянки;
- забезпечення безпеки (у комплексі з іншими технічними засобами аудіо- і відеоконтролю);
- захист і сигналізація на транспортних засобах;

- покращення обліку й руху матеріалів у системі логістики, зокрема на складах [18].

Принцип роботи RFID систем. Система складається з трьох основних компонентів:

- зчитувача "рідера";
- транспондера (так званою "міткою" або "тегом", від англ. tag);
- комп'ютерної системи опрацювання даних [18].

Зчитувач має приймально-передавальний пристрій - він посиляє сигнал до мітки та приймає відповідний сигнал від мітки. Мікропроцесор (пасивної мітки), який віддає сигнал живиться за рахунок індукційного струму з котушки або антени, в якій виробляється струм, коли мітка проходить через електромагнітне поле. Мітки можуть мати більш складну реалізацію з мікропроцесором, що перевіряє і декодує дані, а не просто відповідає ідентифікатором, а також пам'ять, що зберігає дані для наступної передачі, якщо це необхідно [18].

Основні компоненти тегу — інтегральна схема, що керує зв'язком із зчитувачем і антена. Чип має пам'ять, що береже ідентифікаційний код або інші дані. Тег виявляє сигнал від рідера і починає передавати дані, збережені в його пам'яті, обернено в зчитувач [19].

Немає ніякої потреби в контакті або прямої видимості між зчитувачем і міткою, оскільки радіосигнал легко проникає через неметалеві матеріали. Таким чином, теги навіть можуть бути сховані усередині тих об'єктів, що підлягають ідентифікації [19].

Галузь застосування системи визначається її частотою. RFID-системи діляться на:

- високочастотні (850–950 Mhz і 2.4-5 Ghz), що використовуються там, де потрібна велика відстань і висока швидкість зчитування, наприклад контроль залізничних вагонів, автомобілів, системи збору відходів. Наприклад, рідери встановлюють на шлагбаумах, а транспондер закріплюється на вітровому або бічному склі автомобіля. Велика дальність

дії дає можливість безпечної установки рідерів поза межами досяжності людей;

- проміжні частоти (10MHz—15MHz) – там, де повинні бути передані великі кількості даних;
- низькочастотні (100–500 Khz) – використовуються там, де припустимо невеличка відстань між об'єктом і рідером. Звичайна відстань зчитування становить 0,5 метра, а для тегів, умонтованих у маленькі «кнопочки», дальність читання, як правило, ще менше — близько 0,1 метра. Велика антена рідера може якоюсь мірою компенсувати таку дальність дії невеличкого тегу, але випромінювання високовольтних ліній, моторів, комп'ютерів, ламп і т.і., заважає її роботі. Більшість систем керування доступом, безконтактні картки, керування складами і виробництвом використовує низьку частоту [19].

2.2 QR

QR-код – це двовимірний штрих-код, який можна швидко розпізнати сканувальним обладнанням і камерою мобільного телефона. Коди здатні переносити різні формати даних: вебсторінки, зашифрований текст, e-mail, SMS, географічні координати, gif, jpg, png- зображення тощо [20].

Систему QR-кодів (англ. quick response — швидкий відгук) винайшов Масахіро Хара в 1994 році в японській компанії Denso Wave. На початковий дизайн його надихнули чорно-білі фігури з китайської настільної гри Го [20].

Спочатку коди використовували для відстеження автівок на виробництві. Їх спеціально виготовили, щоб сканувати об'єкти, які швидко рухаються. На відміну від звичайних штрих-кодів, QR-коди містять більше інформації, оскільки вони можуть зберігати дані як по горизонталі, так і по вертикалі [20].

Штрих-коди використовують лише горизонтальну інформацію. І хоча чудово працюють, наприклад, для сканування товарів у супермаркетах, QR-

коди мають набагато вищу здатність передавати інформацію. Така універсальність зробила їх популярними для різних сфер [20].

Усе більше компаній помічали зручність QR-кодів та переваги для бізнесу. Зокрема, їх почали застосовувати в харчовій та фармацевтичній галузях. Хара вирішив не зберігати патентні права, і QR-коди швидко знайшли застосування в повсякденному житті людей [20].

У 2000 році QR-коди додали до міжнародних стандартів ISO. Незважаючи на те, що термін “QR code” і є зареєстрованим товарним знаком DENSO Corporation, використання кодів не обкладається жодними ліцензійними відрахуваннями. Це дозволило використовувати їх по всьому світу [20].

Із винайденням та поширенням смартфонів популярність QR-кодів вийшла на новий рівень. Зараз QR-коди використовують у різних галузях: як під час виготовлення та дистрибуції продукції, так і в повсякденних комунікаціях — у сфері послуг та рекламі [20].

Концепція та використання. Сучасний QR-код складається з 7 частин. Кожна з них має піксельний шаблон, схожий на кросворд. Елементи мають певне призначення — кожен передає певну частину інформації за допомогою коду. Напрямок друку, розміщення та пробіли навколо частин коду допомагають відрізнити QR-код від того, що його оточує [20].

Більшість QR-кодів сканується за допомогою смартфона. Існує багато безкоштовних програм для сканування QR-кодів — як для пристроїв на Android, так і на iOS. Останніми роками виробники смартфонів вбудовують такі сканери прямо в камери девайсів, тому достатньо просто ввімкнути камеру й навести фокус на QR-код, щоб він спрацював [20].

QR-коди для смартфонів добре прижилися у сфері реклами, адже вони забезпечують швидший доступ до вебсайту бренду, на відміну від ручного введення URL-адреси. Окрім елементарної зручності для споживача, унікальність QR-кодів полягає в тому, що вони збільшують коефіцієнт конверсії — шанс того, що контакт споживача з рекламою перетвориться на

продаж. Зацікавлені потенційні клієнти швидко потрапляють на вебсайт, без затримок і зусиль. Натомість, вводячи адресу вручну, користувач може втратити інтерес до сайту [20].

QR-коди розміщують на вивісках, білбордах, автобусах, візитних картках, у книжках і на будь-яких об'єктах, про які користувачі можуть потребувати інформації [20].

У 2008 році в Японії розробили QR-коди для кладовищ. Їх викарбовували на надгробках, щоб відвідувачі мали змогу переглядати інформацію про померлого, а члени сім'ї — відстежувати відвідування [20].

Використовують QR-коди і в банкінгу — їх зображають на банкнотах. У 2011 році в Данії випустили першу в світі офіційну монету з QR-кодом, щоб відсвяткувати сторіччя будівлі Королівського нідерландського монетного двору. Монету можна просканувати за допомогою смартфона й потрапити на вебсайт з інформацією про історичну подію та дизайн монети. У 2014 році Центральний банк Нігерії випустив банкноту номіналом 100 найр на честь свого столітнього ювілею. Це була перша банкнота, у дизайні якої було використано QR-код. Відсканований код переводить на сайт, який розповідає історію сторіччя Нігерії. Згодом так почали робити і в інших країнах [20].

Окрім цього, QR-коди активно використовують для оплати послуг. Банки й платіжні системи зберігають у такий спосіб інформацію про банківські рахунки або кредитні картки та проводять платежі за допомогою QR-коду, зокрема в мобільних додатках [20].

QR-коди стали сучасною і зручною альтернативою штрих-кодам. Новий формат дав змогу розширити способи використання та відкрив нові можливості для бізнесів [20].

QR-коди — це зручний інструмент для комунікації брендів [20].

Їх оригінально використовують у рекламі та для зв'язку зі споживачами [20].

Із розвитком нових технологій QR-коди отримують нові сміливі втілення [20].

2.3 Концепція системи

Прийнято рішення для переліку працівників, що виконують роботи на інших підприємствах та мають перепустки цих підприємств, використовувати в якості міток перепустки відповідних підприємств. Перепустка являє собою пасивну RO RFID мітку. Частота відносної половини перепусток – 125Khz, інша половина має частоту 13.56Mhz.

На перепустку наносять qr-код у випадку, коли працівник задіяний на об'єкті, де немає внутрішнього виробництва перепусток або існують географічні перешкоди для його отримання.

Приклад індивідуальної перепустки наведено на рис. 2.1.



Рисунок 2.1 – Приклад перепустки працівника

Таким чином необхідно забезпечити можливість зчитування перепусток обох частот, цим потребам відповідає мобільний портативний

зчитувач. Приклад зчитувача, який може бути застосований у парі з мобільним телефоном під керуванням Android для розпізнавання перепустки, наведено нижче на рис. 2.2.



Рисунок 2.2 – RFID-зчитувач для мобільних пристроїв під керівництвом оперативної системи Android

Пристрій підходить для використання на мобільних телефонах, що вже застосовуються прорабами та керівниками площадок для інших потреб.

Цільові можливості роботи пристрою на обох частотах та основні параметри наведено на пакуванні на рис. 2.3.



Рисунок 2.3 – Основні параметри зчитувача

Для ділянок та об'єктів, де мобільні телефони не можуть бути використані через агресивні умови праці, прийнято рішення використовувати промислові термінали. Вони мають підвищений рівень захисту від зовнішніх чинників, таких як пил, бруд, атипичні температурні показники, мають захист від падінь та ін.

Окрім цього вони представляють собою звичайний мобільний телефон під керівництвом ОС Android, але цікаві вони тим, що мають вже вбудований зчитувач RFID-міток частот, що відповідають наявним перепускам працівників. На рис. 2.4 наведено приклад промислового терміналу, що відповідає потребам підприємства та планується застосовуватись на виробництві.

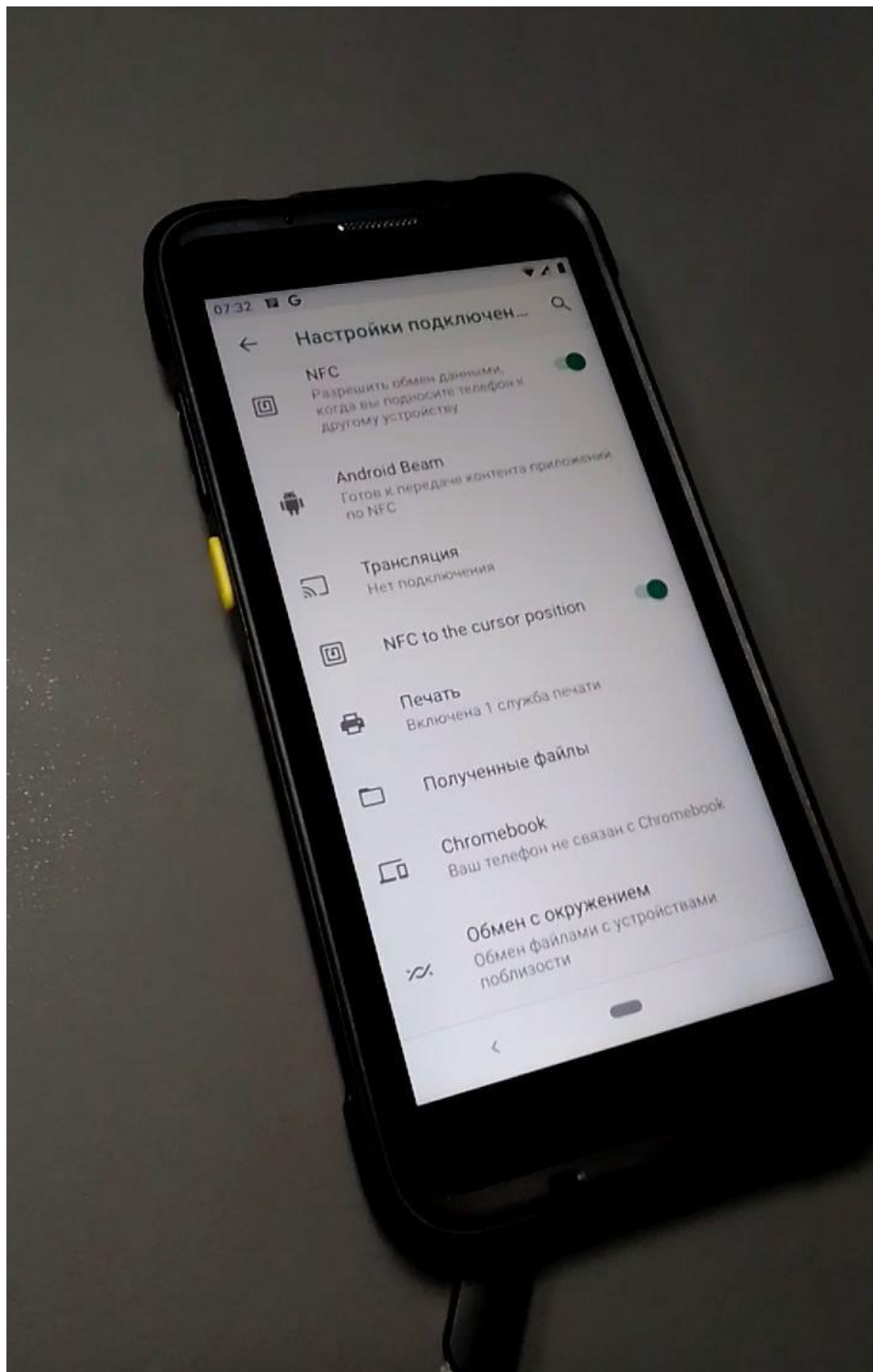


Рисунок 2.4 – Пример промышленного терминала

Зчитувач в таких терміналах вмонтовано в задню кришку пристрою в районі камери, використано пластикову основу, тому не створюються перешкоди для сканування міток. Вбудований у промисловий термінал зчитувач наведено на рис. 2.5.



Рисунок 2.5 – Вмонтований у промисловий термінал зчитувач

Для організації контролю перепусток з QR-кодом використовуються камери телефону або терміналу та спеціальні бібліотеки.

Для організації контролю перепусток з RFID-мітками використовуються портативні зчитувачі для телефону або вбудовані до терміналу зчитувачі та спеціальні бібліотеки.

Систему збору інформації реалізовано наступним чином: за кожною площадкою чи об'єктом є закріплений керівник та бригада працівників.

На початку виконання робіт керівник сканує перепустки працівників одним зі способів, за допомогою програмного забезпечення, що проєктується, дані ідентифікуються та автоматично передаються на сервер, для передачі даних використовуються мобільний Internet.

У разі проблем з підключенням до мережі дані зберігаються локально та будуть передані з наступною партією даних.

По завершенню робочої зміни працівників відбувається аналогічна процедура, фіксується завершення робочого дня.

За потреби, програмне забезпечення, що проєктується, надає доступ до дозвільних документів працівників, що задіяні на даний момент на об'єкті. Дана функція корисна на випадок позапланових перевірок на підприємствах.

Також програмний продукт виключає помилки та людський фактор у фіксації інформації, оскільки не дозволяє повторне відкриття і закриття зміни та інші помилкові дії.

Захист від шахрайства реалізовано шляхом вибору пасивних міток, що неможливо перезаписати.

Для ліквідації шахрайства з часом пристрою та мережею використано дитячі Google-аккаунти, що використовуються на корпоративних телефонах та терміналах, таким чином обмежено можливість корегування дати та часу.

3 ОПИС ПРОГРАМИ

3.1 Порівняльний аналіз мов програмування

Для розробки програмного коду мобільних застосунків найчастіше використовують наступні мови програмування: Java, Swift, Kotlin [21].

Нижче наведено порівняльний аналіз найпоширенішої мови для написання нативних застосунків – Java з кросплатформною мовою програмування Swift [21].

В таблиці 3.1 наведено порівняння мов Java і Swift.

Таблиця 3.1 – Порівняння мов Java і Swift

	Java	Swift
Нативність для Android	+	-
Популярність	Дуже популярний	Дуже популярний
Фреймворки	Багато фреймворків	Мало фреймворків
Вивчення	Легше вчити	Складніше вчити
Схожість з C++	+	-
Надійність та захищеність від розповсюджених помилок	+	-

Що стосується популярності, Java і Swift не поступаються один одному: вони обидва дуже широко застосовуються в розробці [21].

Все залежить від цільової бази пристроїв. Swift є кросплатформною мовою програмування, натомість Java за рахунок нативності може використовувати значно більші можливості пристрою [21].

Одним з важливих чинників при виборі мови програмування є число фреймворків. Вони суттєво спрощують роботу розробника, роблять її більш комфортною [21].

За рахунок фреймворків Java використовує максимум можливостей для реалізації застосунку, надає більш широкий вибір інструментів для втілення специфічних задач [21].

Що стосується простоти в вивченні, Java безумовно є переможцем - його синтаксис простіше і легше, ніж у Swift, хоча Java типізована. За рахунок специфічного інструментарію та нативності Java користується більш широким попитом при розробці застосунків [21].

Переваги мови Java:

- синтаксис мови Java схожий на синтаксис мови C++, і програмістам, що знають мови C і C++, його вивчення не складає труднощів;
- крім того, Java – повністю об'єктно-орієнтована мова, навіть більшою мірою, ніж C++. Всі сутності в мові Java є об'єктами, за винятком небагатьох основних типів (primitive types), наприклад чисел;
- висока надійність. Творці забезпечили мову Java засобами, що дозволяють виключити саму можливість створювати програми, в яких були б приховані найбільш поширені помилки [21].

Для цього в мові Java зроблено наступне.

1. Виключена можливість явного виділення і звільнення пам'яті, пам'ять у мові Java звільняється автоматично за допомогою механізму збірки сміття. Програміст захищений під час розробки від помилок, пов'язаних з неправильним використанням пам'яті.

2. Заборонена арифметика покажчиків. Тепер програмісти в принципі не можуть стерти дані з пам'яті внаслідок неправильного використання покажчиків.

3. Виключено множинне наслідування. Його замінили новим поняттям – інтерфейсом. Інтерфейс дає програмісту майже все, що той може отримати від множинного спадкоємства, уникаючи при цьому складнощів, що виникають при управлінні ієрархіями класів [21].

Відповідно до порівняльної характеристики мов програмування та переваг розробки мовою Java прийнято рішення про використання останньої при розробці Android-застосунку.

3.2 Опитування зацікавлених осіб

Для виявлення основних потреб користувачів було виконано внутрішнє опитування цільової аудиторії користувачів – співробітників компанії.

Виявлено, що найчастіше виникає потреба в автоматизації фіксації початку та завершенні робочої зміни.

Враховано побажання середнього керівного складу – співробітника площадки підприємства, що виконує фіксацію даних в системі, а саме структуризація та впорядкування інформації про дозвільні документи співробітників.

Також враховано функції передачі даних на сервер, примусової передачі даних за місяць, резервної передачі даних, локальне зберігання даних.

За результатами опитування було сформовано вимоги до функціональних характеристик, вимоги до складу і параметрів технічних засобів, вимоги до надійності, вимоги до інформаційної та програмної сумісності.

Надалі застосунок іменується системою (система взаємодій окремих функцій та методів).

Основні вимоги до функціональних характеристик:

- система повинна надавати можливість фіксації початку та завершення робочої зміни за допомогою перепустки з QR-кодом;
- система повинна надавати можливість створення облікового запису працівника;
- система повинна надавати можливість фіксації початку та завершення робочої зміни за допомогою перепустки з RFID-міткою двох частот – 125KHz та 13.56MHz;
- система повинна надавати перелік співробітників, що знаходяться на зміні на даний момент часу;

- система має надавати можливість ручної ініціації передачі денного звіту на сервер;
- система має надавати можливість ручної ініціації передачі місячного звіту на сервер;
- система повинна надавати доступ до даних про дозвільні документи тих співробітників, які знаходяться на зміні на даний момент часу;
- система має містити обробку помилок, повідомляти про це користувача;
- система має надавати кольорову індикацію статусу сканування перепустки;
- система має зберігати дані у базі даних.

Вимоги до надійності:

- система має відповідати користувачу при введенні невірних даних;
- система повинна виконуватись в окремому потоці для забезпечення безперервної роботи;
- система повинна відповідати на дії користувача відповідно.

Вимоги до складу і параметрів технічних засобів, які необхідні для функціонування системи:

- пристрій під керуванням ОС Android;
- RFID-зчитувач з підтримкою частоти 125KHz та 13.56MHz;
- доступ до мережі Internet;
- наявність камери;
- наявність дитячого аккаунта Google;
- можливість роботи в окремому потоці.

Вимоги до інформаційної та програмної сумісності – для функціонування системи необхідна наявність версії Android 9 і вище та API відповідного рівня.

3.3 Архітектура програми

Для реалізації програмного продукту обрано структурний підхід, який полягає в декомпозиції системи –розбитті її на окремі функції, що будуть автоматизовані. Система розбивається на функціональні підсистеми, які також діляться на підфункції і т. д. Таким чином за програмну реалізацію функцій відповідає окремий блок коду. На рис. 3.1 наведено умовний поділ загального обсягу програмного коду на блоки обробки функцій, блоки реалізації інтерфейсу користувача, реалізації запитів до локальної та глобальної баз даних (БД).

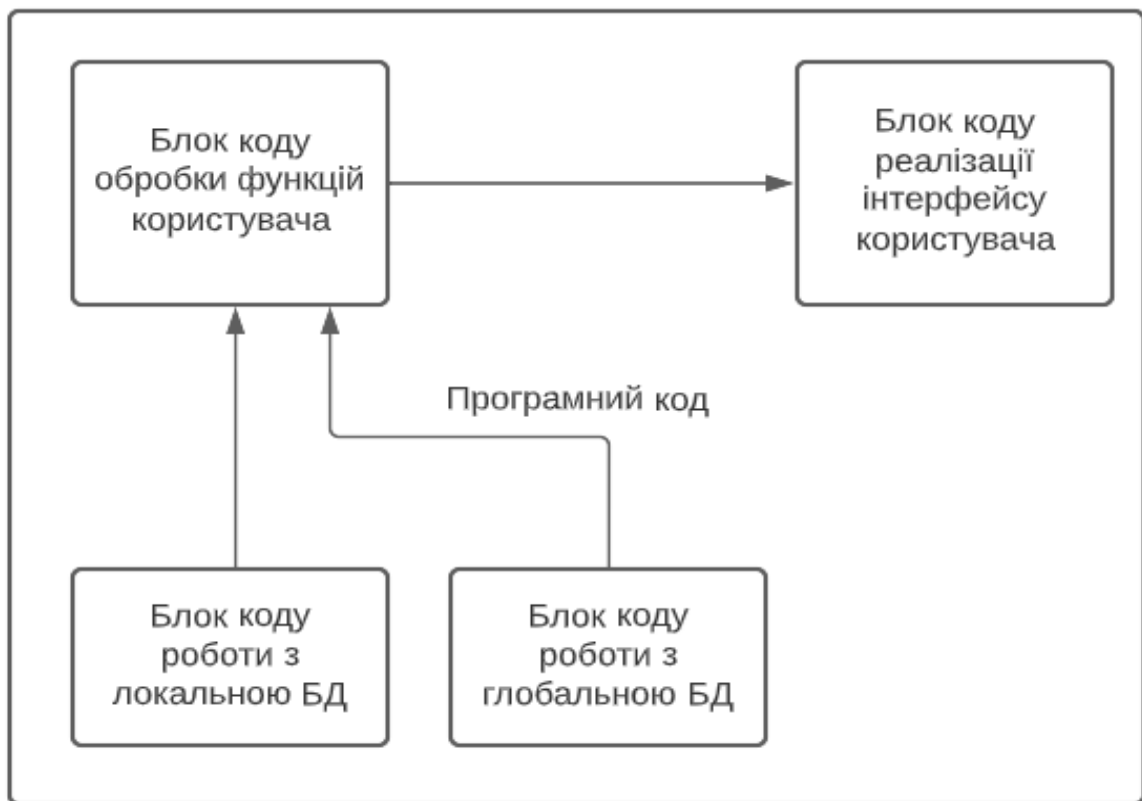


Рисунок 3.1 – Умовний поділ програмного коду на блоки

Також на рис. 3.2 наведено концептуальну схему взаємодії програмного продукту з таблицями глобальної БД серверу та локальної БД

пристрою для зберігання даних, з нормувальниками або керівниками площадок та відділом кадрів підприємства.

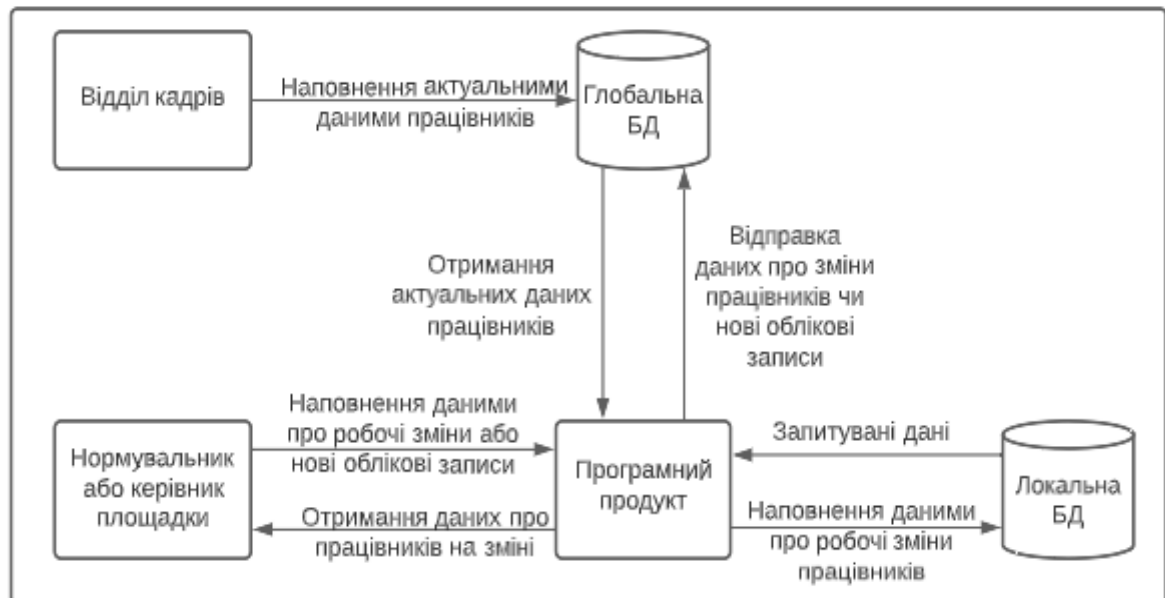


Рисунок 3.2 – Концептуальна схема взаємодії програмного продукту

Програмний продукт взаємодіє з таблицями глобальної БД для збору та централізації інформації про робочі зміни підприємства.

На основі роботи відділу кадрів такими даними як фото, ППП та дата прийому на роботу, спеціальність, дозвільні документи та документи про освіту, наповнюється глобальна БД, яка використовується застосунком, як сховище облікових записів користувачів.

Відповідно до потреб користувача прийнято рішення про програмну реалізацію функцій, use-case діаграми яких наведені на рис. 3.3.

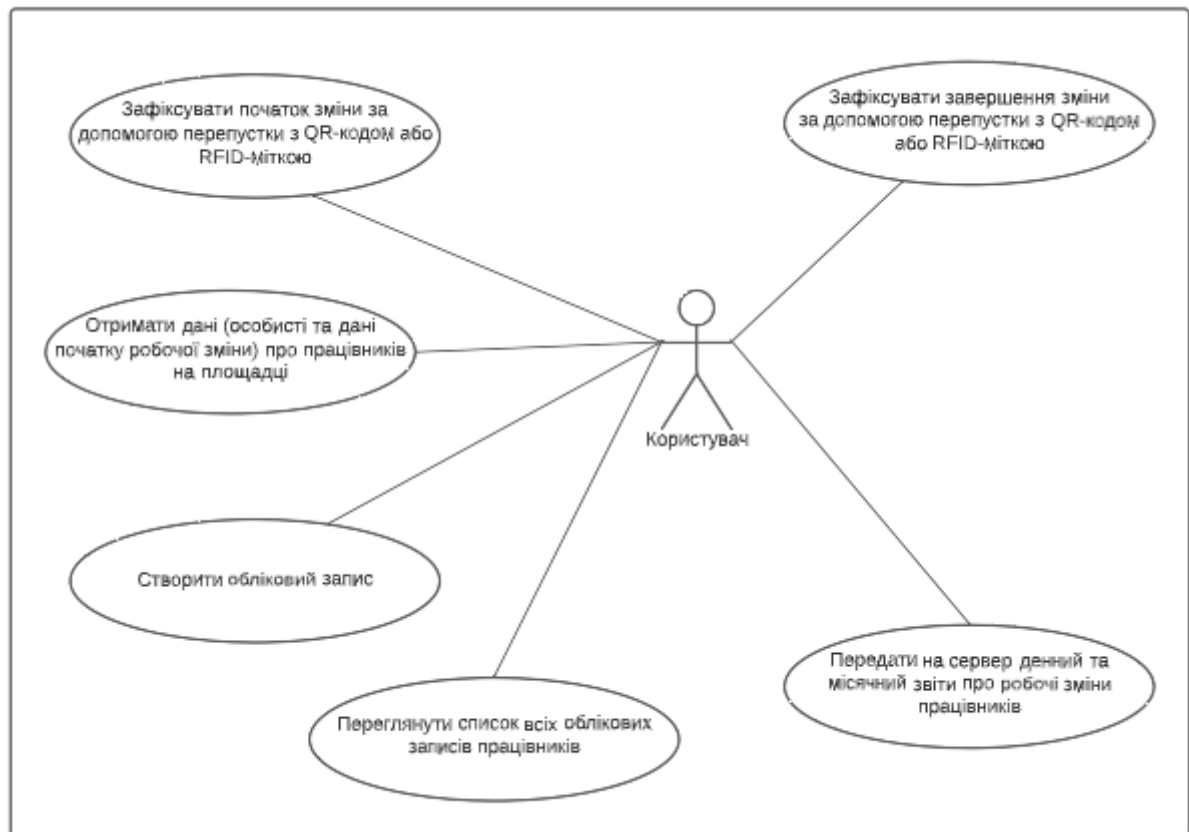


Рисунок 3.3 – Use-case діаграма функцій користувача

Прийнято рішення про програмну реалізацію наступних можливостей для користувача:

- фіксування початку зміни за допомогою перепустки з QR-кодом або RFID-міткою;
- фіксування завершення зміни за допомогою перепустки з QR-кодом або RFID-міткою;
- отримання особистих даних (ППП, фото, дата працевлаштування, робоча спеціальність, дозвільні документи) працівників, що знаходяться на зміні;
- створення облікового запису;
- перегляду облікових записів всіх користувачів;
- ручна передача денного та місячного звітів на сервер.

Відповідно до потреб у даних для реєстрації облікового запису розроблено BPMN 2.0 діаграму реєстрації облікового запису користувача, яку наведено на рис. 3.4.

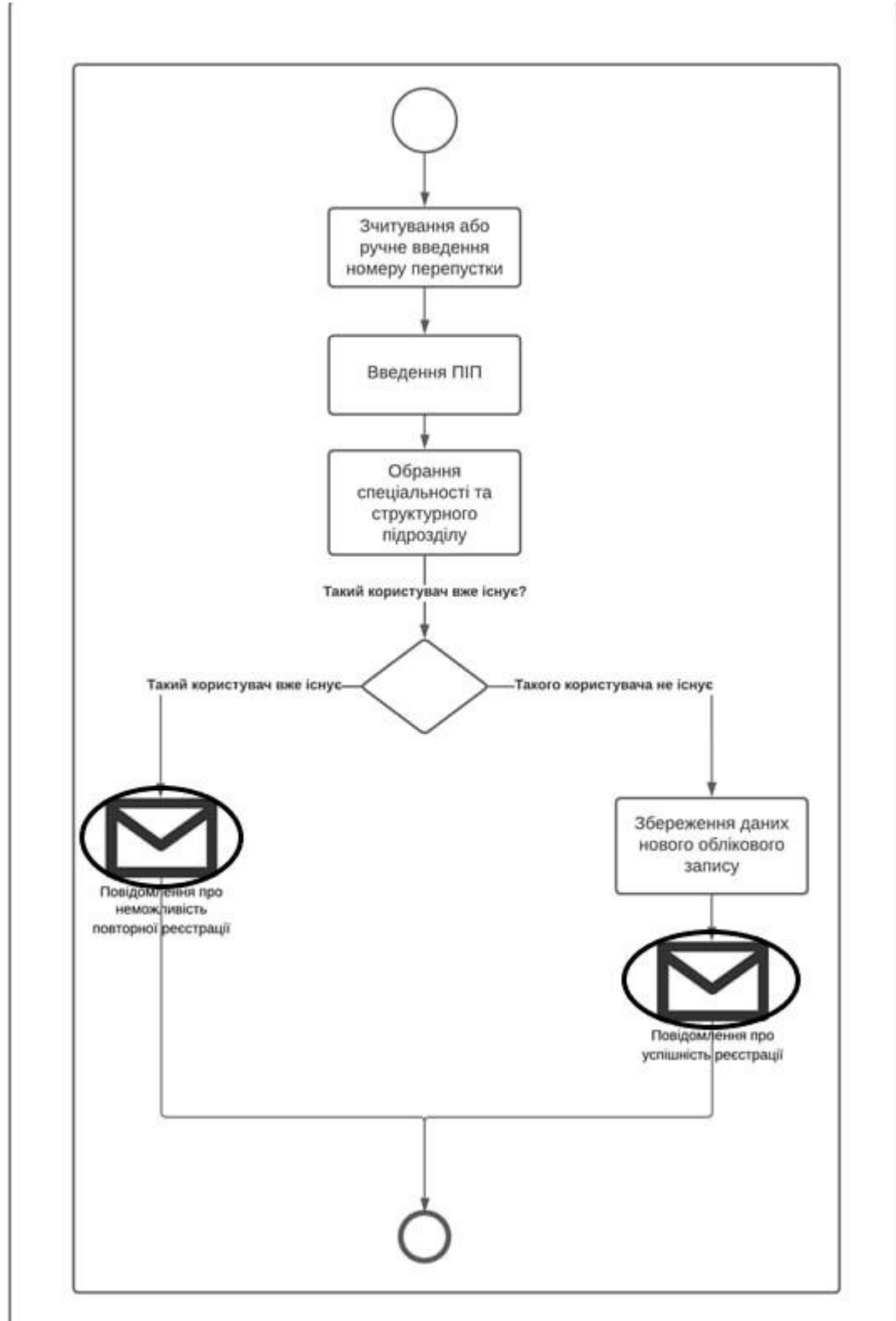


Рисунок 3.4 – Діаграма реєстрації облікового запису користувача

Процес реєстрації облікового запису користувача виглядає наступним чином.

Крок 1. Користувач переходить до реєстрації.

Крок 2. Користувач проводить зчитування перепустки або ручне введення номеру перепустки.

Крок 3. Користувач вводить ППП.

Крок 4. Користувач обирає структурний підрозділ.

Крок 5. Користувач обирає спеціальність.

Крок 6. Користувач ініціює перевірку введених даних.

Крок 7. Система проводить перевірку даних на унікальність.

Крок 8. Система надає результати перевірки користувачу.

Крок 9. Користувач ініціює збереження даних.

Крок 10. Система зберігає дані нового облікового запису

Крок 11. Користувач отримує повідомлення про успішне створення нового облікового запису.

Якщо такий користувач вже існує, система надає відповідне повідомлення про вже виконану реєстрацію.

Якщо користувач не надає номер перепустки, ППП, структурний підрозділ та спеціальність, реєстрація неможлива.

Також розроблено BPMN 2.0 діаграми подій для загального випадку фіксації відкриття та завершення робочої зміни, наведені на рис. 3.5, рис. 3.6.

Доступ до функцій фіксації відкриття та завершення робочої зміни мають лише зареєстровані облікові записи користувачів.

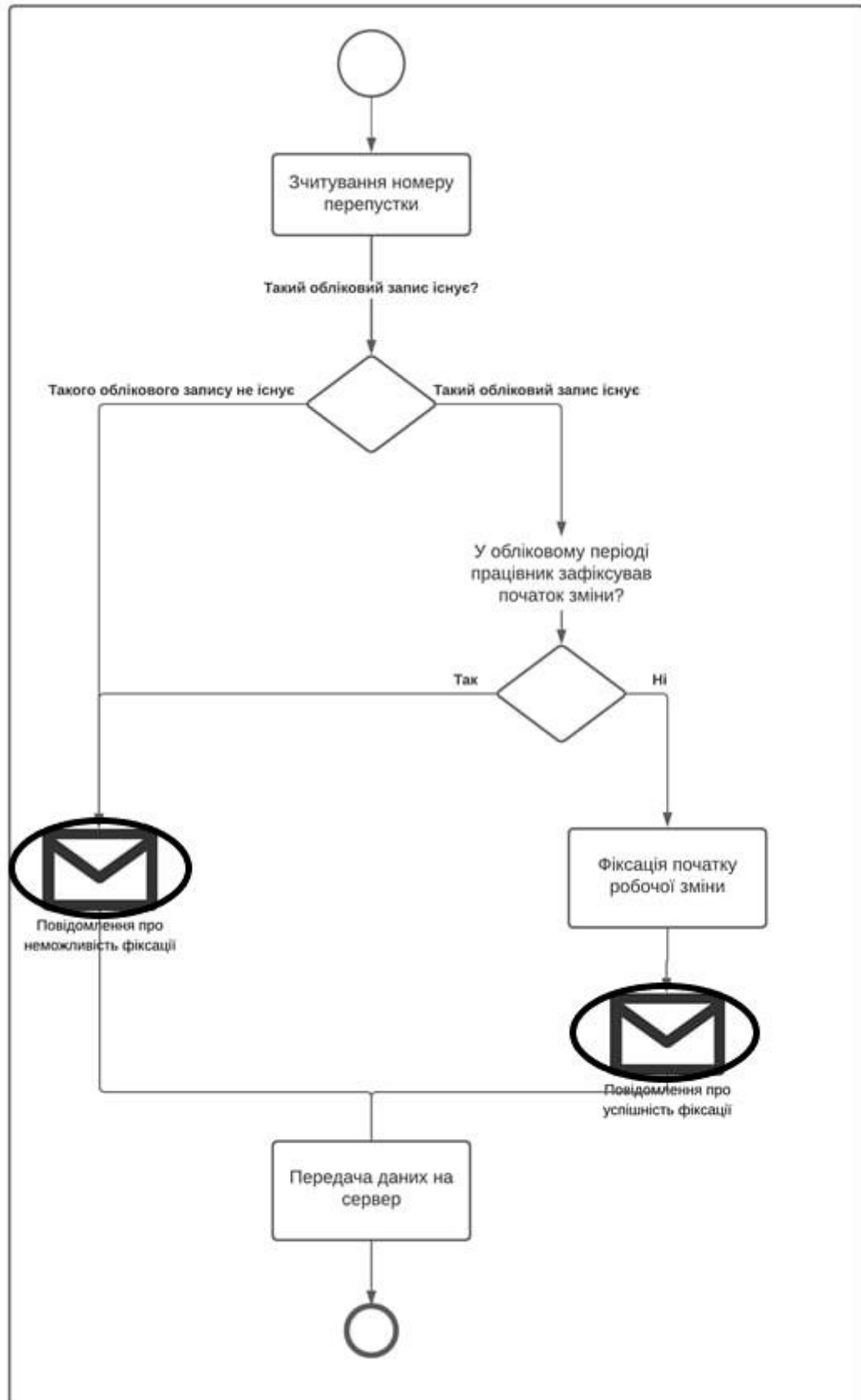


Рисунок 3.5 – Діаграма фіксації початку зміни

Процес фіксації початку робочої зміни виглядає наступним чином.

Крок 1. Користувач переходить до розділу, який відповідає його потребі.

Крок 2. Користувач сканує перепустку.

Крок 3. Система проводить перевірку на існування облікового запису.

Крок 4. Система проводить перевірку на наявність даних про початок робочої зміни.

Крок 5. Система фіксує дані початку робочої зміни.

Крок 6. Користувач підтверджує збереження даних.

Крок 7. Система зберігає дані про початок робочої зміни.

Крок 8. Користувач отримує повідомлення про успішне фіксування даних початку робочої зміни.

Якщо облікового запису не знайдено, користувач отримує відповідне повідомлення. Якщо вже зафіксовано початок робочої зміни, користувач отримує відповідне повідомлення.

У випадку фіксування початку чи завершення робочої зміни користувач візуально бачить результати перевірки на наявність облікового запису шляхом виведення результатів сканування перепустки – ППП працівника.

Користувач також бачить дату та час, який буде зафіксовано.

За результатами перевірки на коректність даних користувач, окрім текстового повідомлення зі статусом перевірки, має також візуальну кольорову індикацію, зелений колір фону – для успішної фіксації та червоний у випадку наявності помилок.

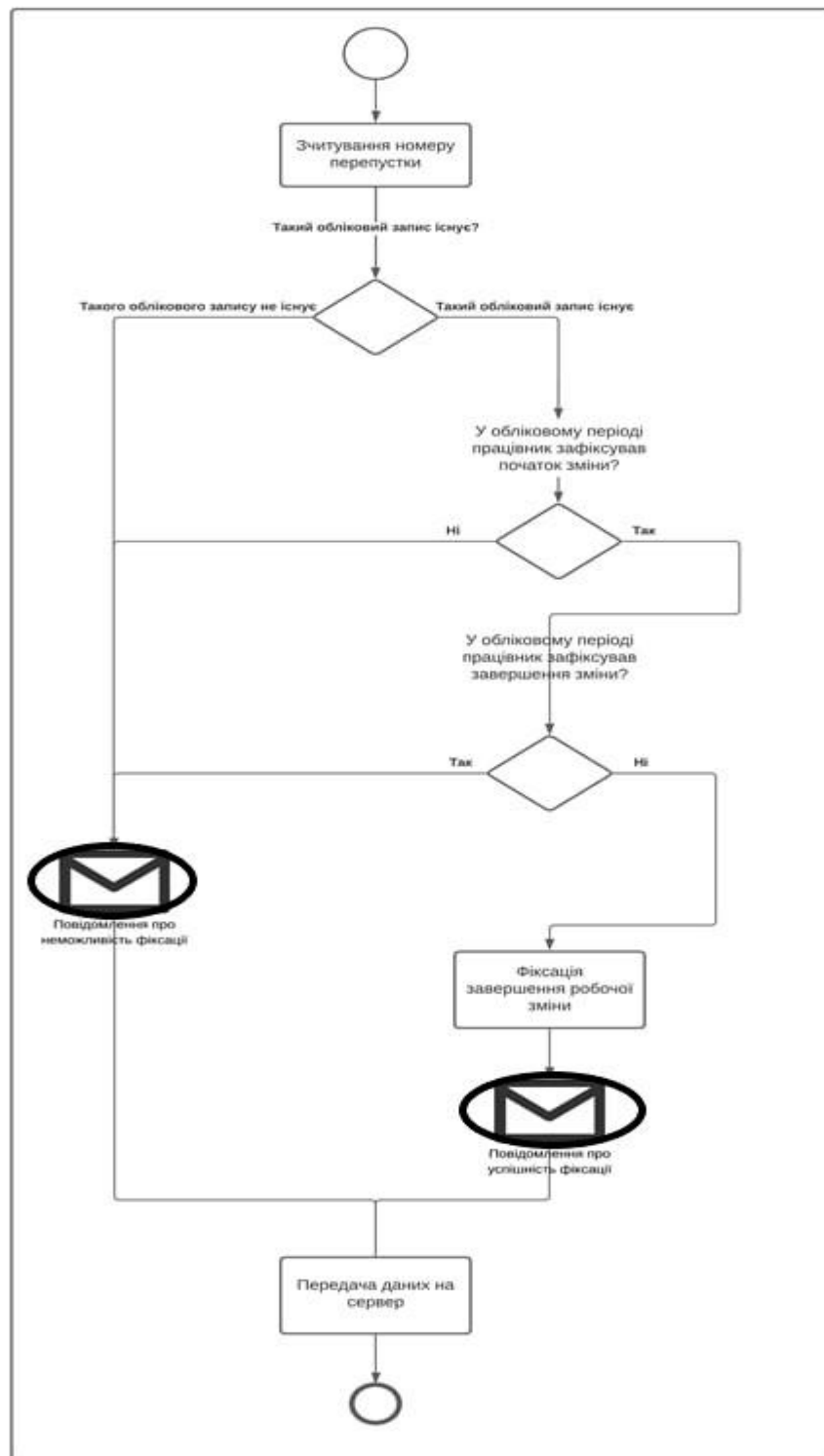


Рисунок 3.6 – Діаграма фіксації завершення зміни

Процес фіксації завершення робочої зміни виглядає наступним чином.

Крок 1. Користувач переходить до розділу, який відповідає його потребі.

Крок 2. Користувач сканує перепустку.

Крок 3. Система проводить перевірку на існування облікового запису.

Крок 4. Система проводить перевірку на наявність даних про початок робочої зміни.

Крок 5. Система проводить перевірку на наявність даних про завершення робочої зміни.

Крок 6. Система фіксує дані завершення робочої зміни.

Крок 7. Користувач підтверджує збереження даних.

Крок 8. Система зберігає дані про завершення робочої зміни.

Крок 9. Користувач отримує повідомлення про успішне фіксування даних завершення робочої зміни.

Якщо облікового запису не знайдено, користувач отримує відповідне повідомлення. Якщо не зафіксовано початок робочої зміни, користувач отримує відповідне повідомлення. Якщо вже зафіксовано завершення робочої зміни, користувач отримує відповідне повідомлення.

3.4 Висновки з розділу 3

У розділі 3 проведено аналіз мов програмування для розробки застосунку. На основі порівняльної характеристики мов програмування, наведеної у підрозділі 3.1, та основних переваг розробки мовою Java прийнято рішення про використання останньої під час розробки застосунку.

У підрозділі 3.3 наведено результати внутрішнього опитування зацікавлених осіб. На основі даних цих опитувань сформовано перелік функцій користувача.

Для реалізації було обрано структурний підхід, також розроблено концептуальну схему взаємодії програмного продукту, use-case діаграми основних функцій користувачів, BPMN 2.0 діаграми реєстрації облікового запису користувача, фіксації початку та завершення робочої зміни.

4 РОЗРОБКА ПРОГРАМИ

4.1 Файли, методи та функції

Програмний код можна умовно поділити на файли взаємодії з таблицями БД та реалізацію сторінок інтерфейс користувача та функції, що його обслуговують.

Файл `TrackersActivity.java` реалізує сторінку сканування перепустки для фіксації початку робочої зміни

В таблиці 4.1 наведено опис призначення файлу `TrackersActivity.java`, методів, класів та змінних програмного коду.

Таблиця 4.1 – Опис призначення файлу `TrackersActivity.java`, методів, класів та змінних програмного коду

Назва методу чи змінної	Призначення
<code>showScanResult</code>	Змінна дозволяє отримувати дані RFID-сканування та оброблювати їх.
<code>start()</code>	Метод дозволяє керувати налаштуваннями камери пристрою для QR-сканування.
<code>onQRCodeRead()</code>	Метод виконує обробку даних за результатами QR-сканування коду.
<code>Class2</code>	За допомогою класу виконується перевірка на наявність шуканої перепустки в базі даних системи.
<code>Class3</code>	За допомогою класу <code>Class3</code> виконується робота з даними про робочі зміни у локальній БД.
<code>SendData</code>	Екземпляр класу використовується для передачі щоденного звіту на сервер. Запуск відбувається автоматично.

Аналогічно реалізовано файл `Trackers_backActivity.java`, який відповідає за фіксацію перепустки під час завершення робочої зміни. Суттєва відмінність цього файлу полягає у реалізації шкали балів, що дає можливість

керівнику площадки корегувати розмір преміальних начислень кожного працівника. Елемент seekBar дозволяє ініціювати та оброблювати події корегування значення шкали.

Файл BDActivity.java реалізує можливість перегляду списку усіх облікових записів в системі.

В таблиці 4.2 наведено опис призначення файлу BDActivity.java, методів, класів та змінних програмного коду.

Таблиця 4.2 – Опис призначення файлу BDActivity.java, методів, класів та змінних програмного коду

Назва методу чи змінної	Призначення
task()	Метод у новому потоці запускає перевірку наявності доступу до мережі Internet.
checkInternet()	Метод перевіряє наявність доступу до мережі Internet.
getJSON()	Метод актуалізує локальну БД користувачів.
loadIntoListView()	Метод відповідає за візуальне наповнення списку актуальними даними для інтерфейсу користувача.
btAddPeople()	Метод містить обробку натискання кнопки переходу до створення нового облікового запису.
btnSupport()	Метод містить обробку натискання кнопки переходу до технічної підтримки відповідно.

Файл RegistrationActivity.java реалізує основне меню користувача.

В таблиці 4.3 наведено опис призначення файлу RegistrationActivity.java, методів, класів та змінних програмного коду.

Таблиця 4.3 – Опис призначення файлу RegistrationActivity.java, методів, класів та змінних програмного коду

Назва методу чи змінної	Призначення
up()	Метод виконує перехід до фіксації початку зміни.
down()	Перехід до фіксації завершення зміни.
report()	Перехід до перегляду списку співробітників на зміні.
btnSendAll()	Перехід до формування та передачі місячного звіту на сервер.
btnSupport()	Перехід до технічної підтримки.
SendData	Екземпляр класу виконує відправку щоденного звіту на сервер

Файл ReportActivity.java надає доступ до переліку співробітників, чия зміна на даний момент є відкритою. Клас Restart наповнює перелік даними з локальної БД. Файл AddActivity.java реалізує можливість створення нового облікового запису. В таблиці 4.4 наведено опис призначення файлу AddActivity.java, методів, класів та змінних програмного коду.

Таблиця 4.4 – Опис призначення файлу AddActivity.java, методів, класів та змінних програмного коду

Назва методу чи змінної	Призначення
task() task_position()	Методи ініціюють перевірку стану доступу до мережі та виконують отримання даних про структурні підрозділи та спеціальності відповідно у окремому потоці за допомогою методів getJSON() та getJSONPosition().
loadIntoLocalDB() loadPosIntoLocalDB()	Методи наповнюють візуальні випадаючі списки підрозділів та спеціальностей даними з БД
validate()	Метод виконує валідацію даних, які вводить користувач.
checkPeople()	Метод дозволяє встановити наявність дублюючого облікового запису або схожого за обліковими даними.
addPeople()	Метод за умови успішності перевірки дозволяє створити новий обліковий запис.
SendData	Екземпляр класу виконує відправку даних нового облікового запису на сервер.

Файл `SendAllReportActivity.java` виконує обробку формування та передачі місячного звіту на сервер.

В таблиці 4.5 наведено опис призначення файлу `SendAllReportActivity.java`, методів, класів та змінних програмного коду.

Таблиця 4.5 – Опис призначення файлу `SendAllReportActivity.java`, методів, класів та змінних програмного коду

Назва методу чи змінної	Призначення
<code>setDate1()</code>	Метод відповідає за відображення діалогу вибору дати місяця, за який буде передано звіт.
<code>setInitialDate1()</code>	Метод встановлює поточну дату за замовчанням.
<code>btnSupport()</code>	Метод виконує перехід до технічної підтримки.
<code>sendAll()</code>	Метод виконує обробку натискання на кнопку відправки звіту та запускає в окремому потоці екземпляр класу <code>SendData1</code> для відправки даних на сервер.

Файл `ActivityPerson.java` відповідає за карточку облікового запису користувача. В таблиці 4.6 наведено опис призначення файлу `ActivityPerson.java`, методів, класів та змінних програмного коду.

Таблиця 4.6 – Опис призначення файлу `ActivityPerson.java`, методів, класів та змінних програмного коду

Назва методу чи змінної	Призначення
<code>getDocumentURL()</code>	Метод відповідає за отримання посилання за фото користувача та його відображення.
<code>getCompany()</code>	Метод реалізує отримання та відображення структурного підрозділу користувача.
<code>getPosition()</code>	Метод відповідає за спеціальність користувача.
<code>getPhone()</code>	Метод відповідає за контактний номер телефону.
<code>getDate()</code>	Метод відповідає за дату працевлаштування.
<code>document()</code>	Метод відповідає за перехід до перегляду дозвільних документів користувача.

Файл `ActivityDocumentList.java` відповідає за перелік дозвільних документів користувача. Метод `getDocumentList()` використовується для отримання та відображення відповідного переліку документів. Тут використано файл `DocumentAdapter.java` для обробки натискання на пункт переліку та переходу до перегляду відповідного документа. Файл `ActivityDocument.java` використовується для відображення обраного до перегляду дозвільного документа. Метод `getDocumentURL()` отримує посилання на документ, що зберігається на сервері, та за допомогою класу `RetrivePDFfromUrl` в окремому потоці відображає в елементі `pdfView` шуканий документ. Файл `SupportActivity.java` реалізує можливість надіслати повідомлення до технічної підтримки.

Файли `Company.java`, `Document.java`, `MessageSupport.java`, `Position.java`, `UserAdd.java` було використано в якості прототипів структур даних при формуванні масивів структурних підрозділів, документів, звернення до технічної підтримки, спеціальності, облікового запису користувача відповідно.

Файл `DataBaseHelper.java` використовується для роботи з локальною БД. В таблиці 4.7 наведено опис призначення файлу `DataBaseHelper.java`, методів, класів та змінних програмного коду.

Таблиця 4.7 – Опис призначення файлу `DataBaseHelper.java`, методів, класів та змінних програмного коду

Назва методу чи змінної	Призначення
<code>onCreate()</code>	Метод за потреби створює таблиці у БД.
<code>onUpgrade()</code>	Метод використовується під час розробки та дозволяє оновлювати структуру БД.
<code>insertCompanyLocalDB()</code>	Метод вставляє у локальну БД дані про структурні підрозділи.
<code>getUsers()</code>	Метод надає інформацію про всі облікові записи користувачів локальної БД.

Файл `ServerApi.java` є інтерфейсом, що використовується для виконання наступних запитів на сервер:

- `callSupport` – відправка звернення до технічної підтримки;
- `getDocumentList` – отримання переліку дозвільних документів користувача;
- `getDocumentURL` – отримання посилання на конкретний документ користувача;
- `getPhotoURL` – отримання посилання на фото користувача;
- `getCompany` – отримання актуальних даних структурного підрозділа;
- `getPosition` – отримання актуальних даних спеціальності;
- `getPhone` – отримання контактного номеру телефону користувача;
- `getDate` – отримання дати працевлаштування користувача.

Нижче будуть наведені файли розмітки з описом призначення, що формують інтерфейс користувача:

- `activity_add.xml` – інтерфейс сторінки створення нового облікового запису користувача;
- `activity_bd.xml` – інтерфейс сторінки перегляду переліку всіх облікових записів користувачів;
- `activity_document.xml` – інтерфейс сторінки перегляду дозвільного документу користувача;
- `activity_document_list.xml` – інтерфейс сторінки перегляду переліку дозвільних документів користувача;
- `activity_person.xml` – інтерфейс сторінки перегляду карточки облікового запису користувача;
- `activity_registration.xml` – інтерфейс сторінки основного меню користувача;

- activity_report.xml – інтерфейс сторінки перегляду переліку співробітників, що знаходяться на зміні у поточний момент часу, відправка щоденного звіту на сервер;
- activity_send_all_report.xml – інтерфейс сторінки відправки місячного звіту на сервер;
- activity_support.xml – інтерфейс сторінки звернень до технічної підтримки;
- activity_trackers.xml – інтерфейс сторінки фіксації початку та завершення робочої зміни.

4.2 Інтерфейс

Застосунок представляє собою звичайну програму для мобільних телефонів під керуванням ОС Android. Застосунок встановлюється шляхом завантаження з Play Market або може бути встановлений шляхом запуску арк-файлу. На рис.4.1 наведено зовнішній вигляд іконки застосунку.

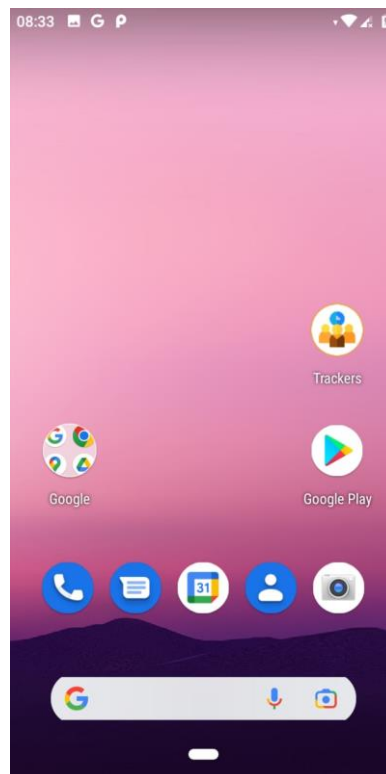


Рисунок 4.1 – Іконка застосунку

Запуск застосунку відбувається шляхом натискання на відповідну іконку.

На рис.4.2 наведено основне меню користувача. Звідси можна перейти до фіксації часу або перегляду бази даних облікових записів користувачів.

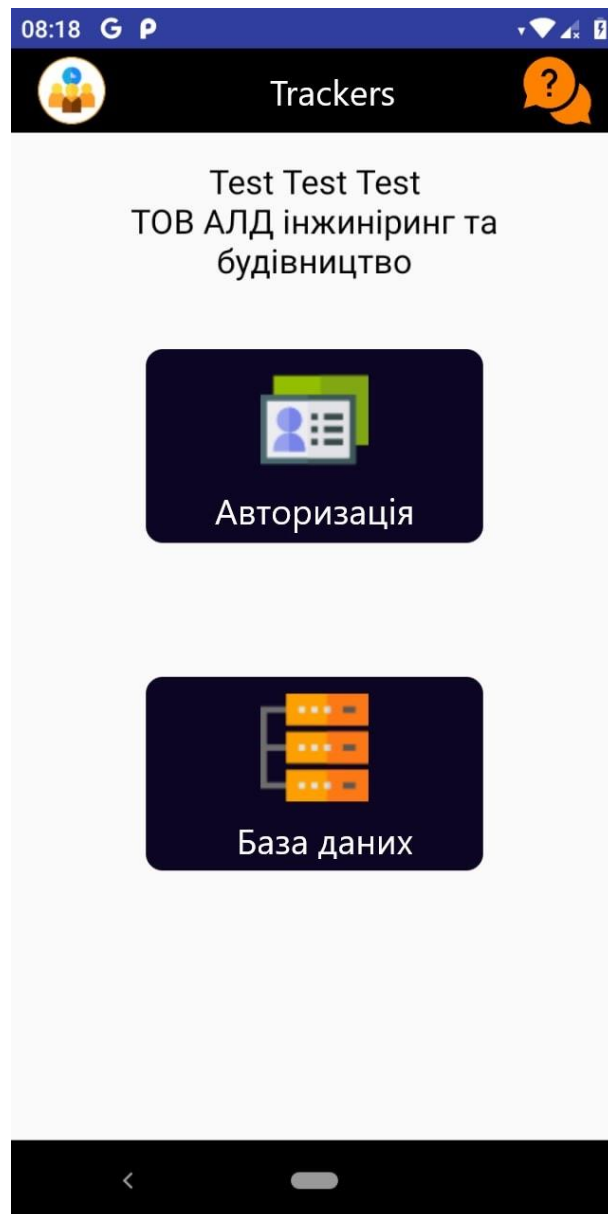


Рисунок 4.2 – Основне меню користувача

Обравши авторизацію, користувач потрапить у наступне меню, де можна потрапити на сторінки фіксації часу або переглянути звіт з тих

користувачів, чий початок робочої зміни зафіксовано, вищезазначене меню наведено на рис 4.3.

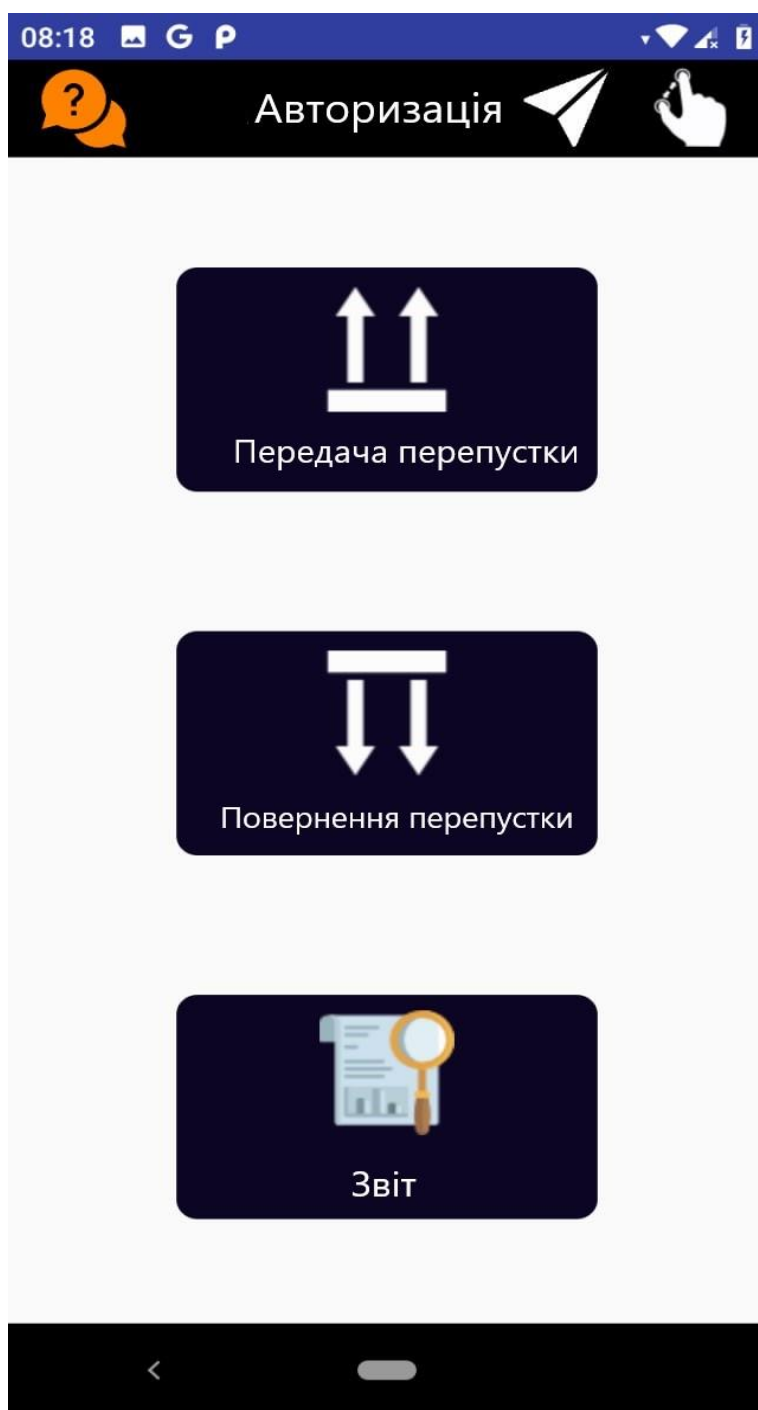


Рисунок 4.3 – Меню користувача розділу «Авторизація»

Для фіксації початку робочої зміни користувач переходить натисканням на кнопку «Передача перепустки». На сторінці – наведено на рис.4.4, що буде відкрита, користувачу доступна область камери пристрою

для сканування перепустки шляхом зчитування QR-коду, також запущено RFID-зчитувач.

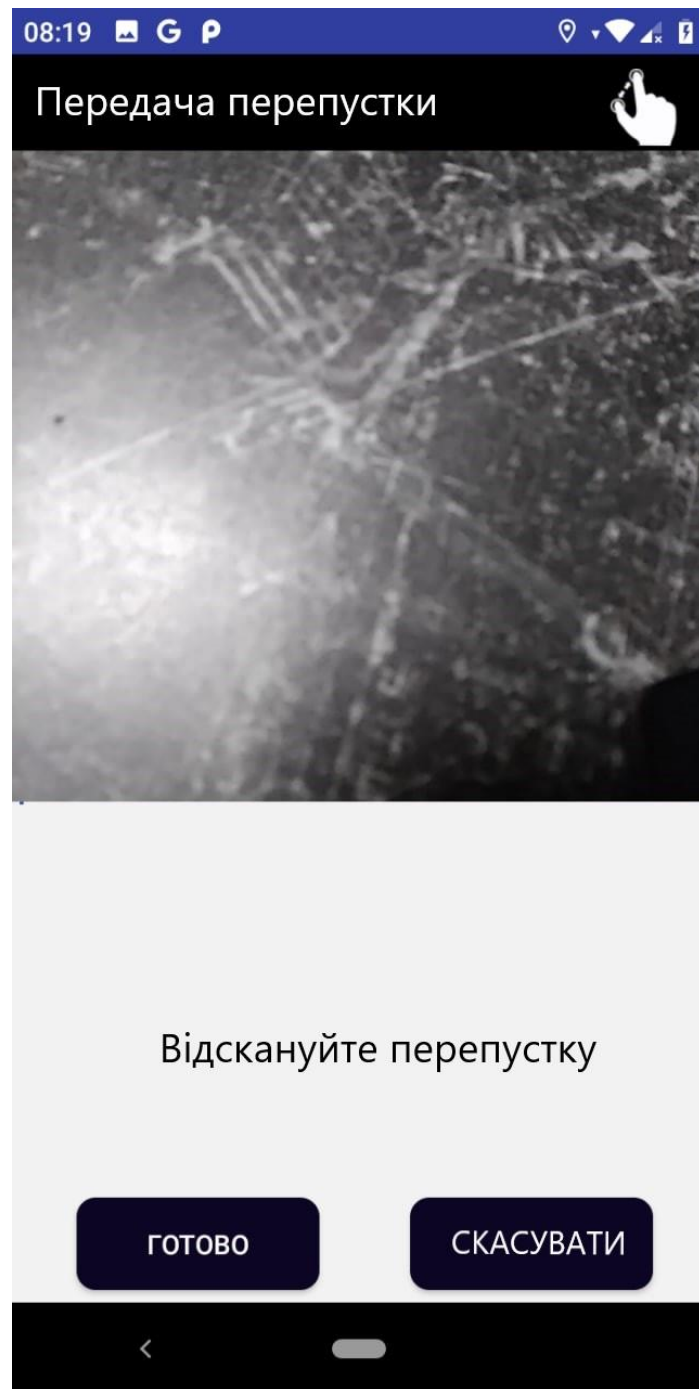


Рисунок 4.4 – Сторінка фіксації початку робочої зміни

Після сканування обраним шляхом перепустки користувач за умови відсутності помилок має побачити ППП власника відсканованої перепустки,

дату та час, який буде зафіксовано як початок робочої зміни, приклад наведено на рис. 4.5.

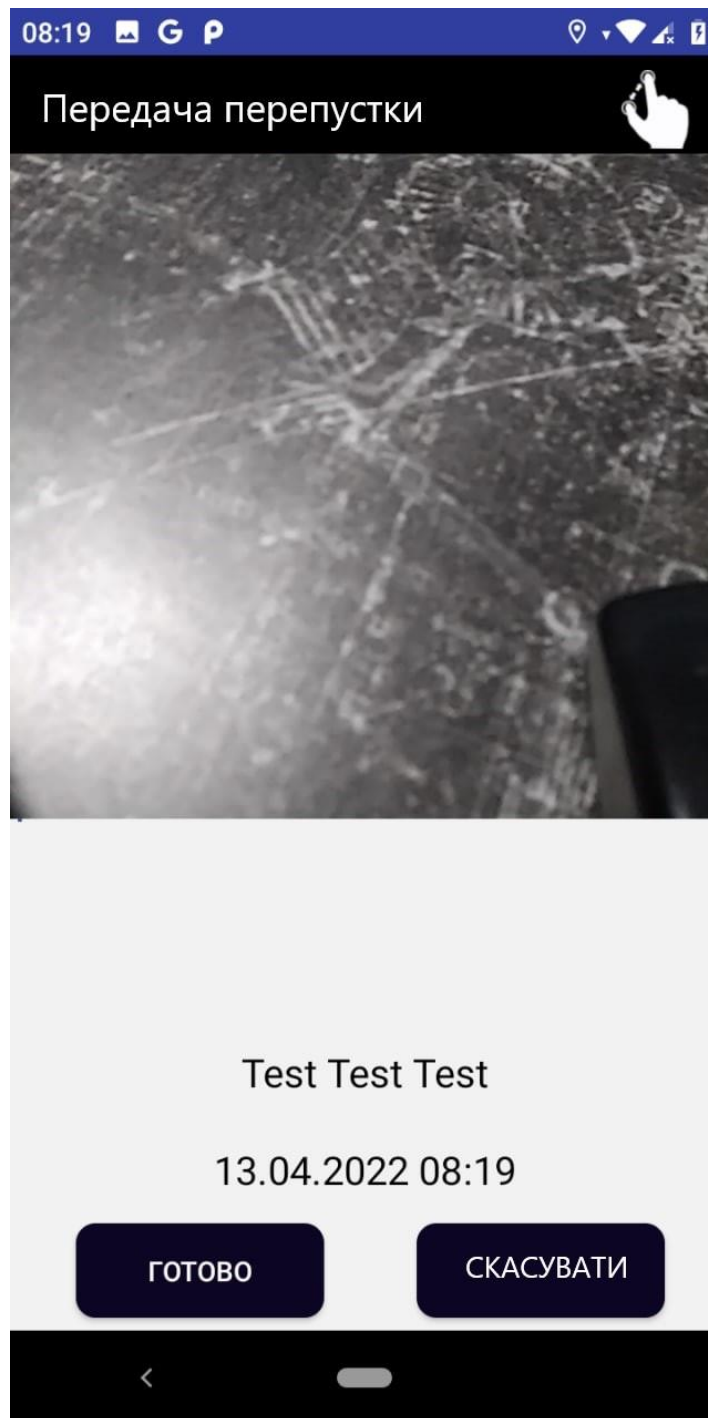


Рисунок 4.5 – Сторінка фіксації початку робочої зміни після сканування перепустки

Натискання на кнопку «Готово» запустить перевірку на коректність даних, за результатами перевірки фон сторінки буде кольорово забарвлено -

зеленим у разі успішності та червоним за наявності помилок відповідно. Приклад успішно пройденої перевірки наведено на рис. 4.6.

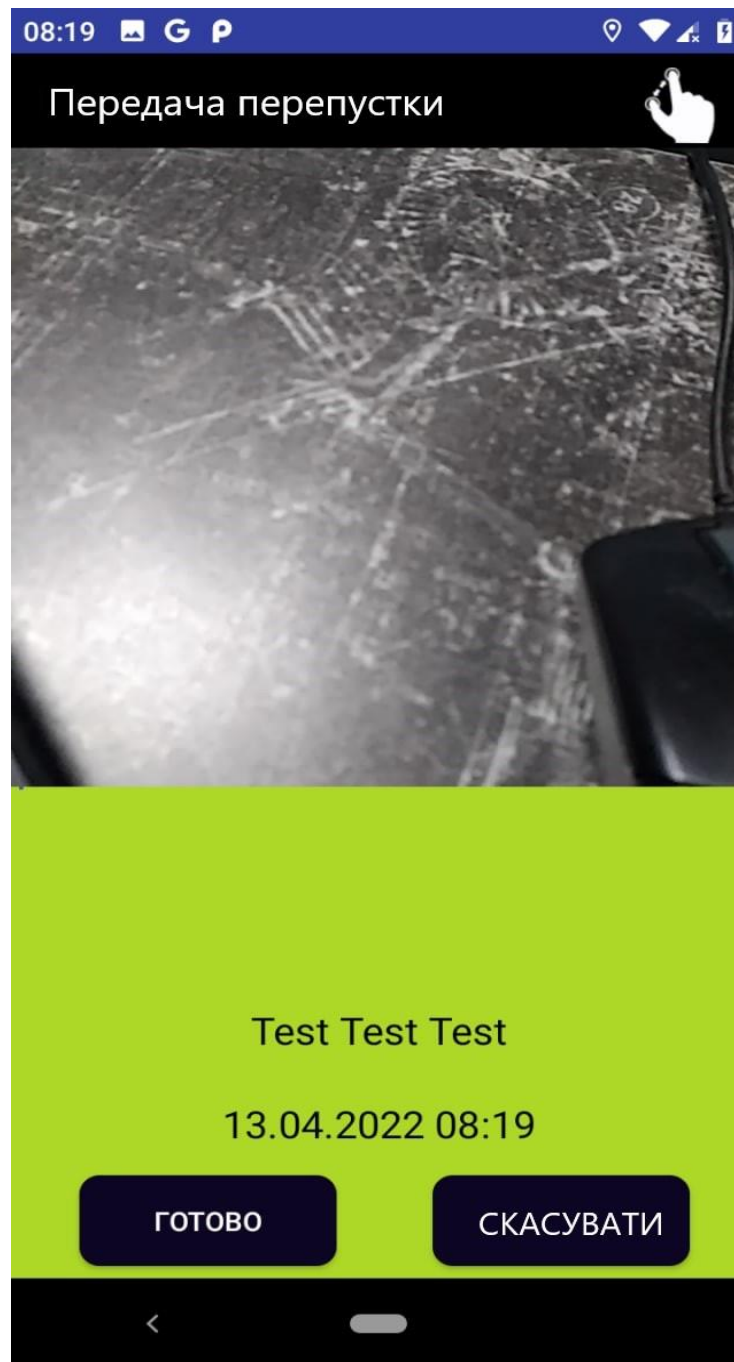


Рисунок 4.6 – Приклад успішної перевірки даних на коректність при фіксації початку робочої зміни

У разі наявності помилок під час перевірки на коректність даних при фіксації початку робочої зміни користувач побачить текстове повідомлення з

текстом помилки – приклад наведено на рис. 4.7. Після цього фон сторінки буде забарвлено у червоний колір, приклад наведено на рис. 4.8.

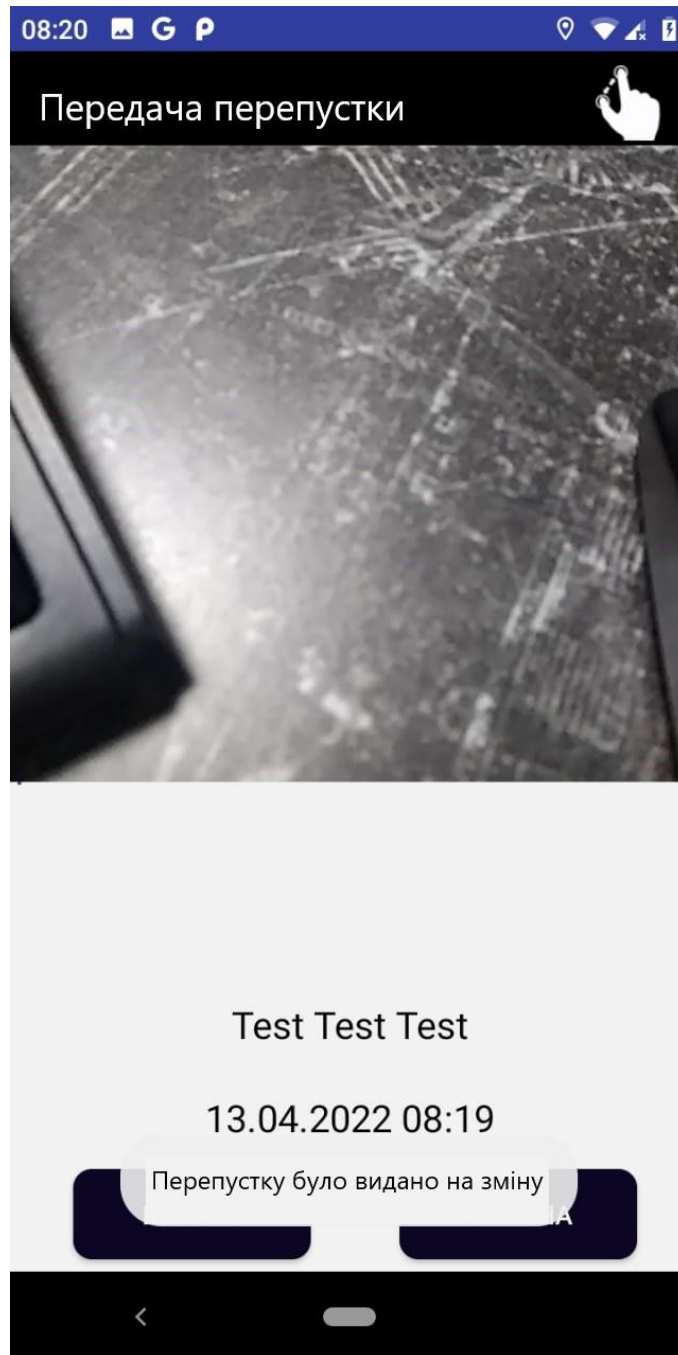


Рисунок 4.7 – Приклад повідомлення про помилку при фіксації початку
робочої зміни

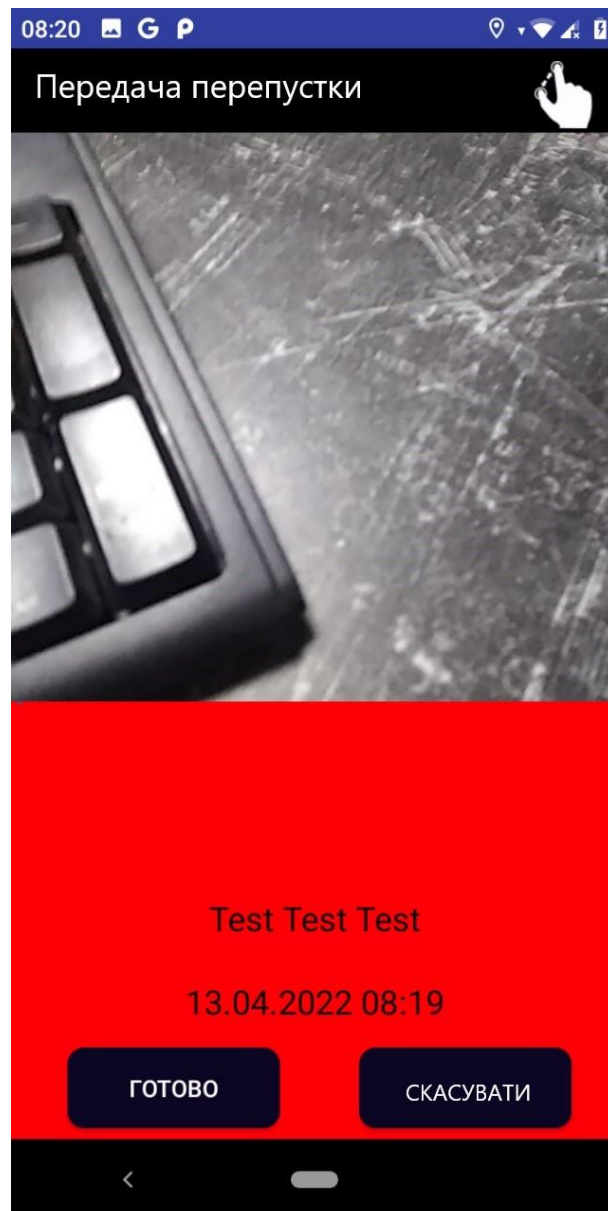


Рисунок 4.8 – Приклад кольорового забарвлення фону сторінки фіксації початку робочої зміни у разі наявності помилок у даних

У разі успішної фіксації початку робочої зміни та у разі натискання на кнопку «Скасувати» сторінку буде перезавантажено для сканування наступної перепустки. Після успішної фіксації початку робочої зміни з меню, що наведено на рис. 4.3, користувач може натисканням кнопки «Звіт» перейти до перегляду звіту, що складається з переліку працівників, чий початок робочої зміни зафіксовано. Приклад сторінки такого звіту наведено на рис. 4.9.

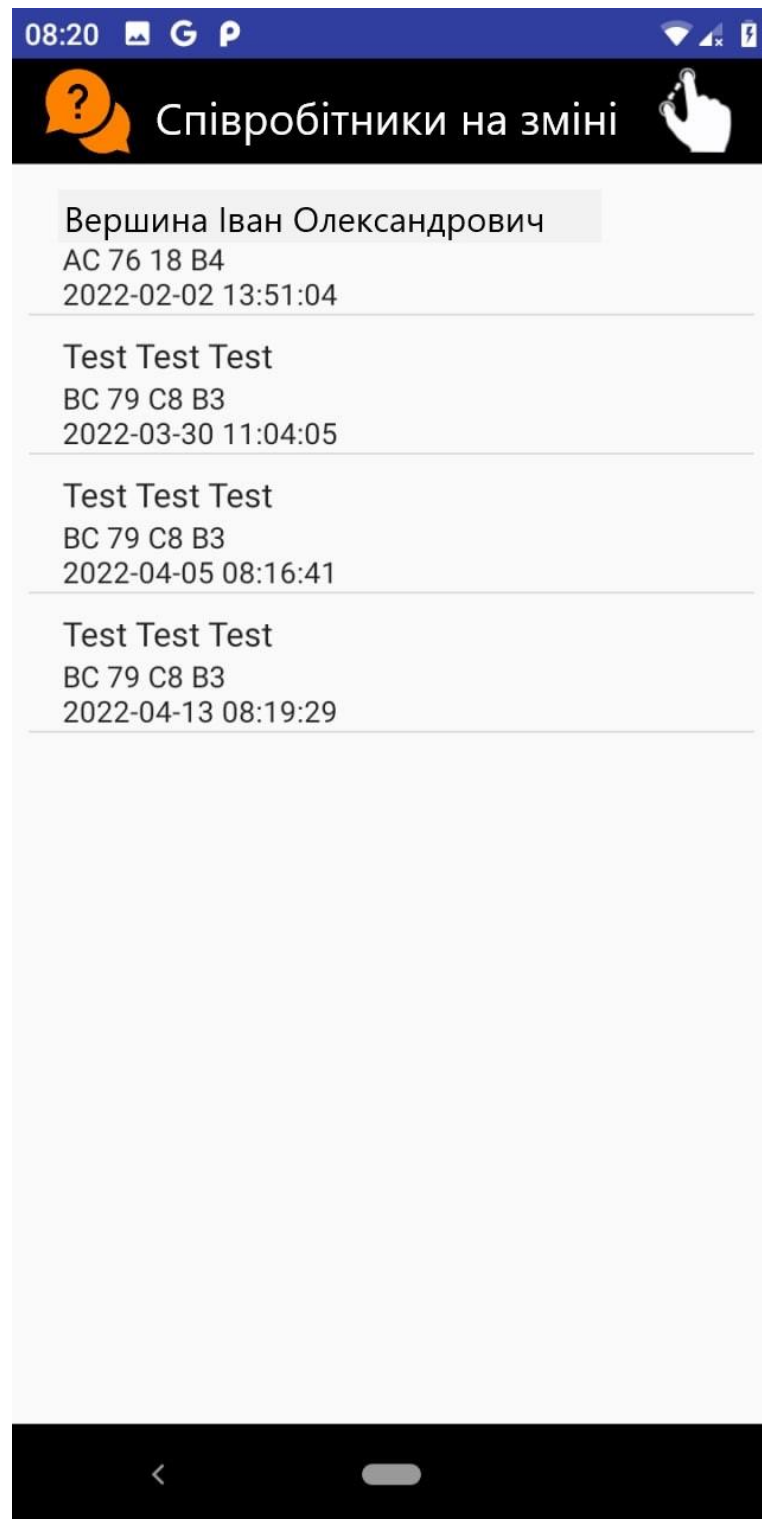


Рисунок 4.9 – Приклад сторінки звіту співробітників на зміні

Натисканням на відповідного співробітника з переліку користувач може перейти до сторінки «Карточка співробітника», приклад такої сторінки наведено на рис. 4.10.

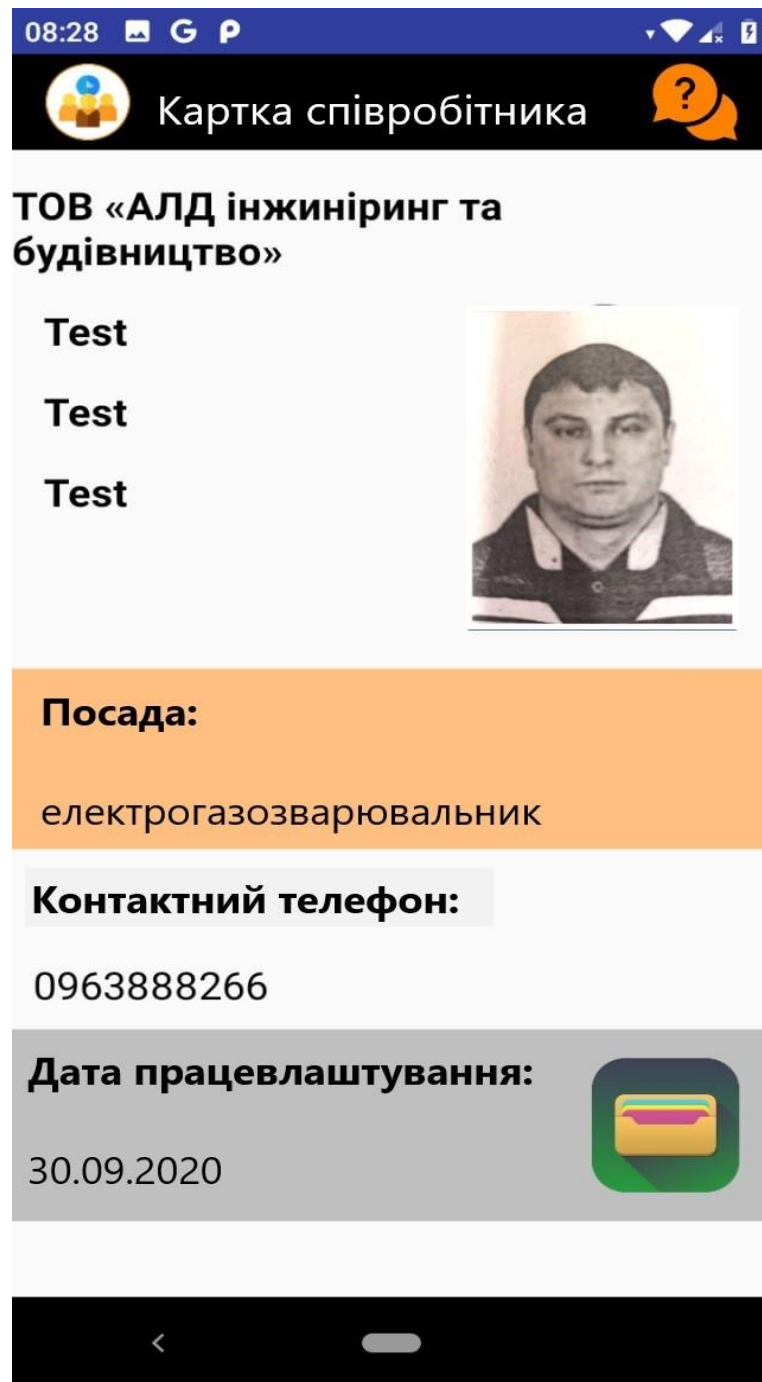


Рисунок 4.10 – Приклад сторінки - карточки співробітника

Натискання на кнопку із зображенням файлів відкриє сторінку «Документи співробітника», приклад такої сторінки наведено на рис. 4.11.

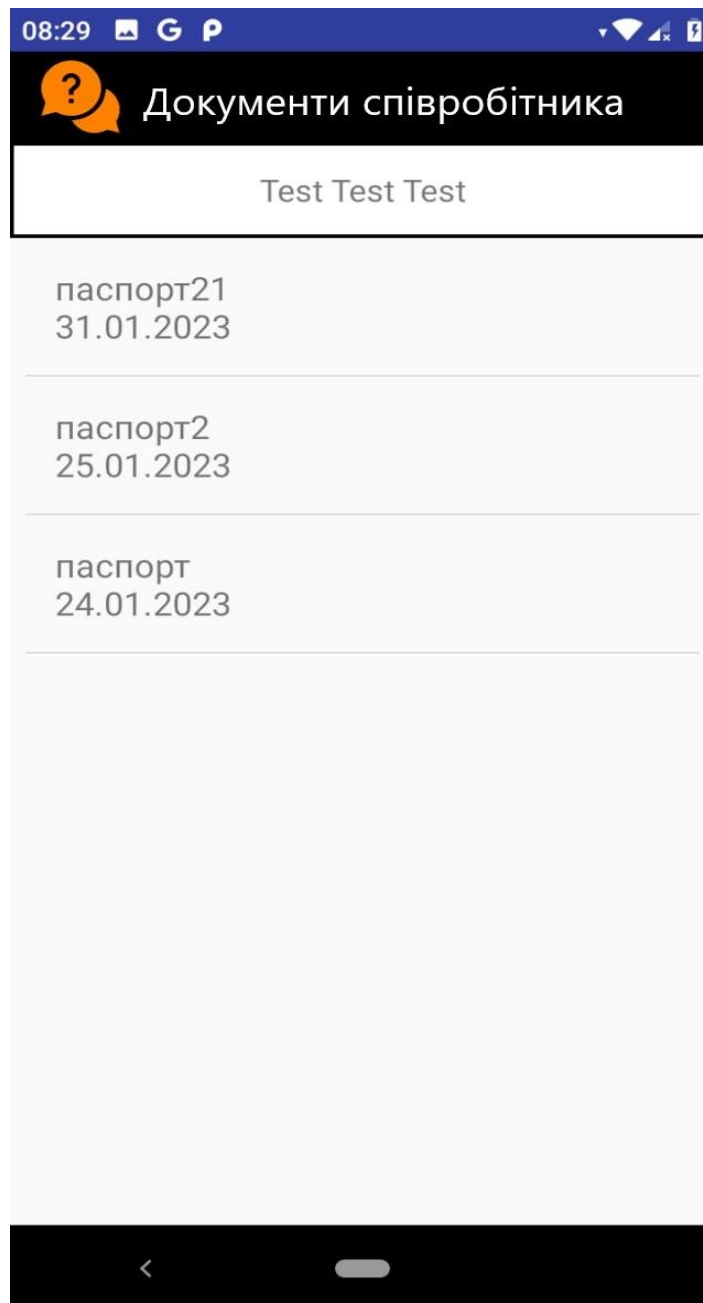


Рисунок 4.11 – Сторінка «Документи співробітника»

Натискання на обраний з переліку документ відкриє сторінку «Документ співробітника», приклад такої сторінки наведено на рис. 4.12.

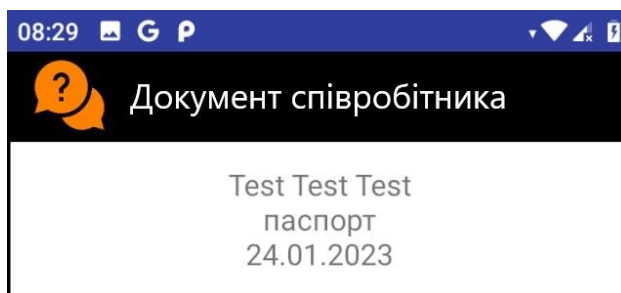


Рисунок 4.12 – Сторінка «Документ співробітника»

З меню сторінки «Авторизація», що наведено на рис. 4.3, користувач натисканням на кнопку «Повернення перепустки» переходить до фіксації завершення робочої зміни. Сканування перепустки відбувається за аналогією, як і при фіксації початку робочої зміни. Після зчитування перепустки користувач може бачити ППП власника перепустки, дату та час, що буде зафіксовано як завершення робочої зміни, та шкалу розміру премії для поточного працівника за день. Приклад такої сторінки наведено на рис. 4.13.

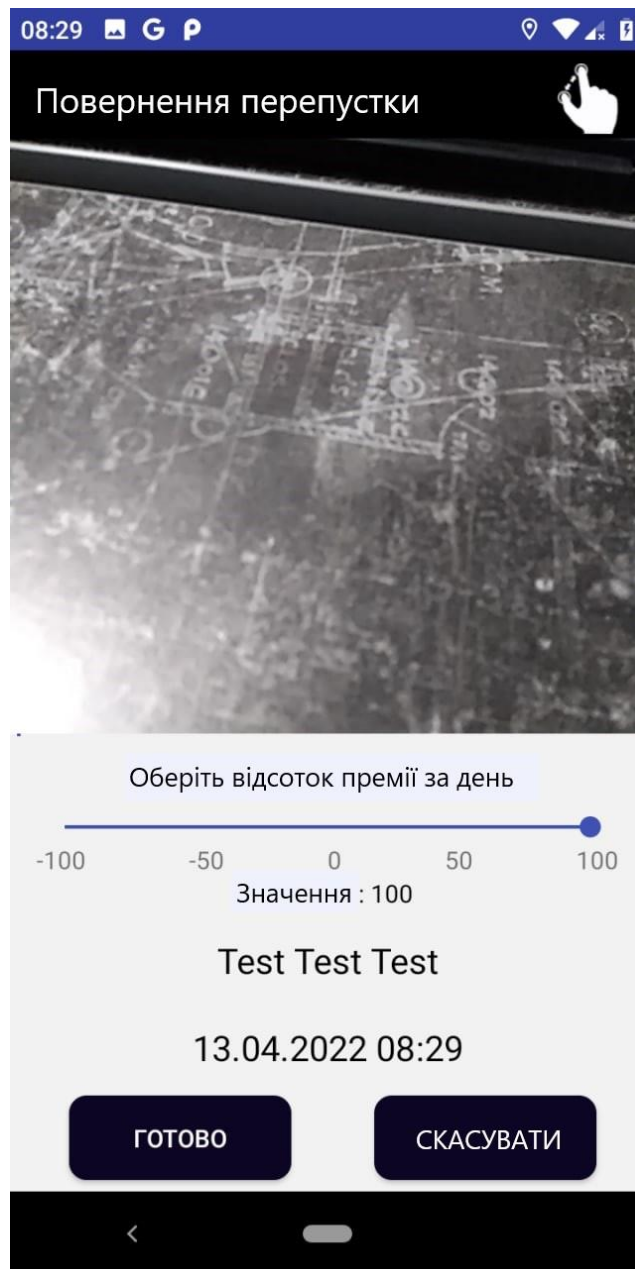


Рисунок 4.13 – Сторінка фіксації завершення робочої зміни після сканування перепустки

У разі наявності помилок під час перевірки на коректність даних при фіксаціях завершення робочої зміни користувач побачить текстове повідомлення з текстом помилки – приклад наведено на рис. 4.14. Після цього фон сторінки буде забарвлено у червоний колір, приклад наведено на рис. 4.15.

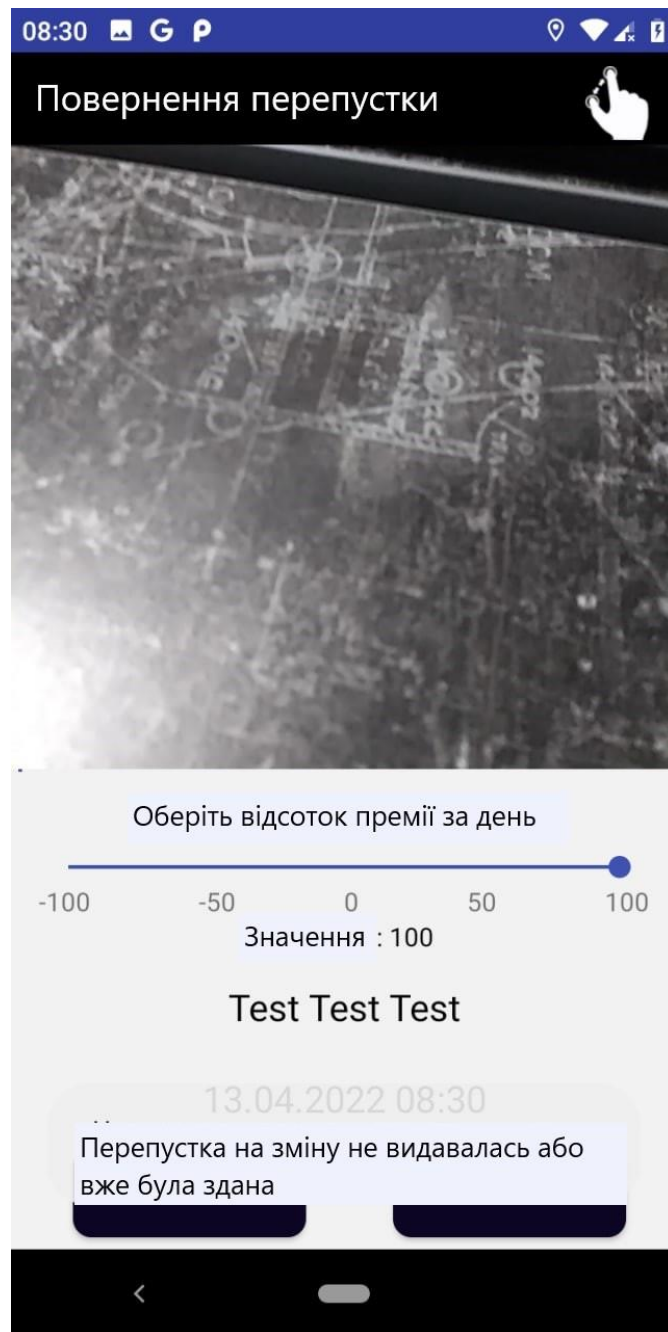


Рисунок 4.14 – Приклад повідомлення про помилку під час фіксації
завершення робочої зміни

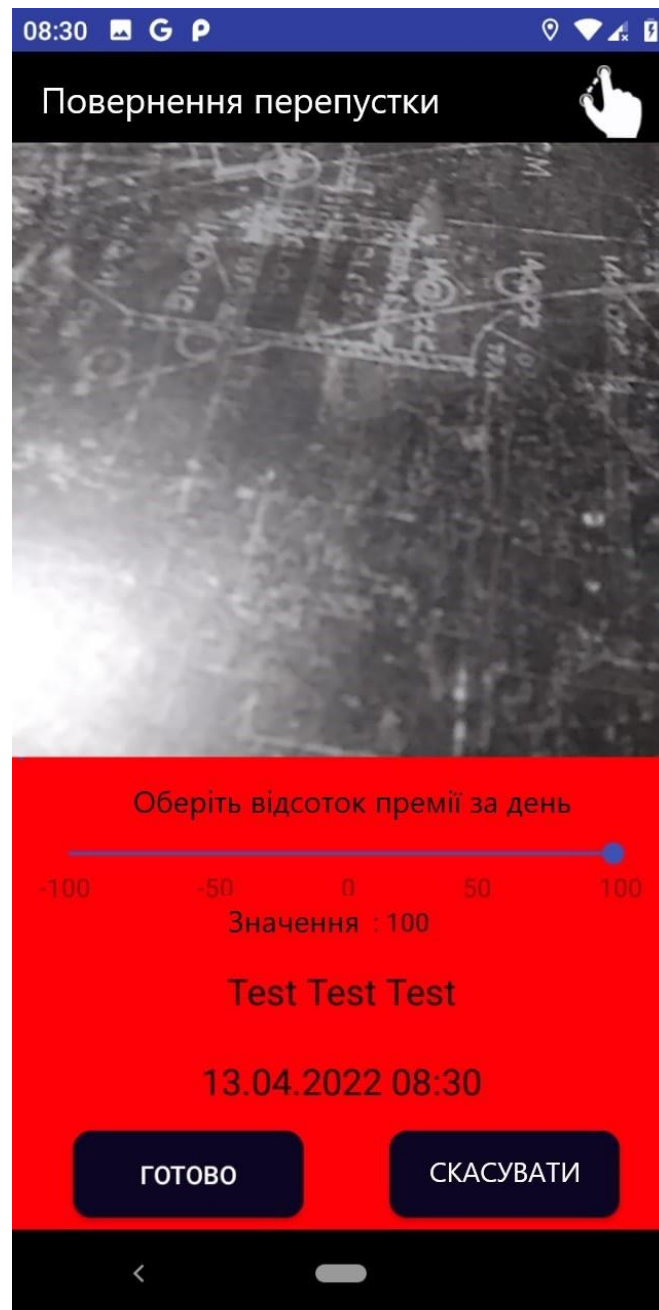


Рисунок 4.15 – Приклад сторінки фіксації завершення робочої зміни при наявності помилок у даних

Також зі сторінки «Авторизація» користувач може у ручному режимі передати денний звіт на сервер, для цього необхідно натиснути на кнопку у правому верхньому куту. Після цього користувач отримає попередження, приклад наведено на рис. 4.16, а після підтвердження дані буде надіслано.

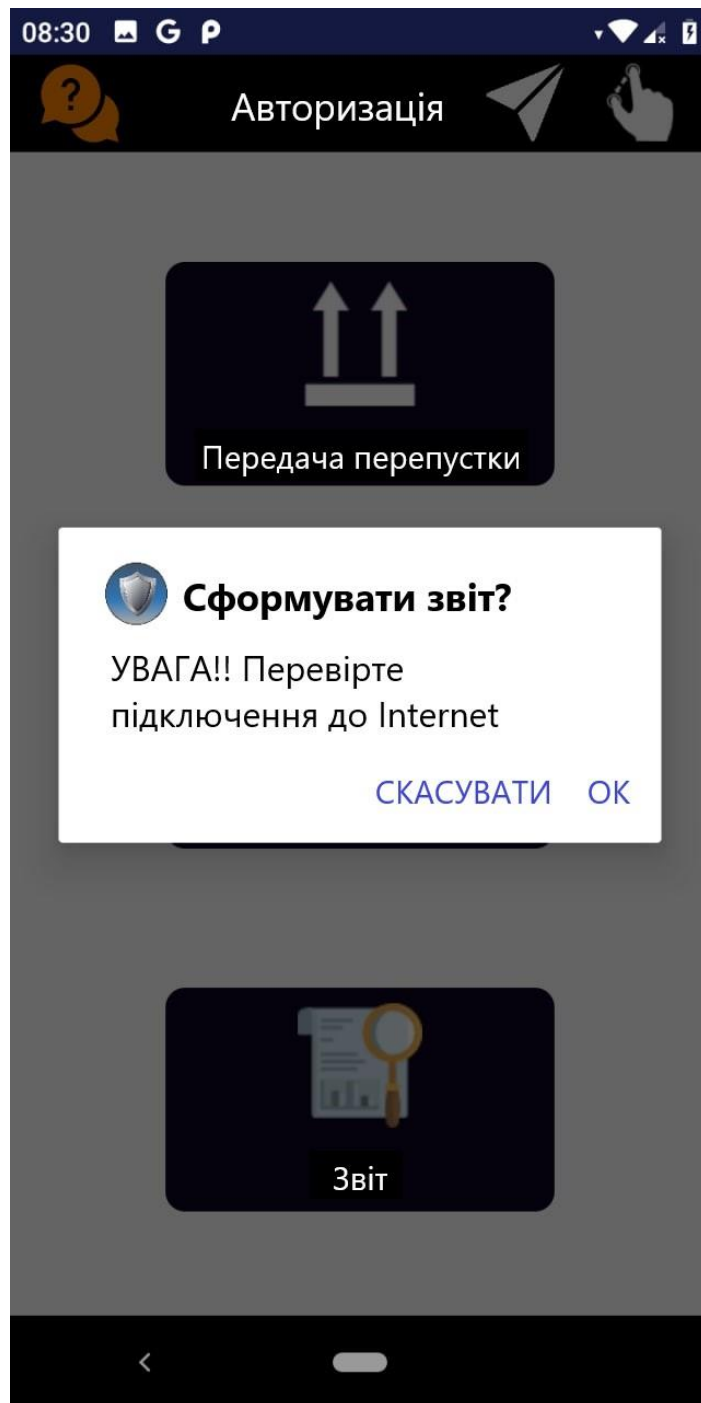


Рисунок 4.16 – Приклад попередження при відправці щоденного звіту на сервер у ручному режимі

Натискання на сторінці «Авторизація» кнопки із зображенням курсору у шапці відкриє сторінку передачі місячного звіту, приклад наведено на рис. 4.17.

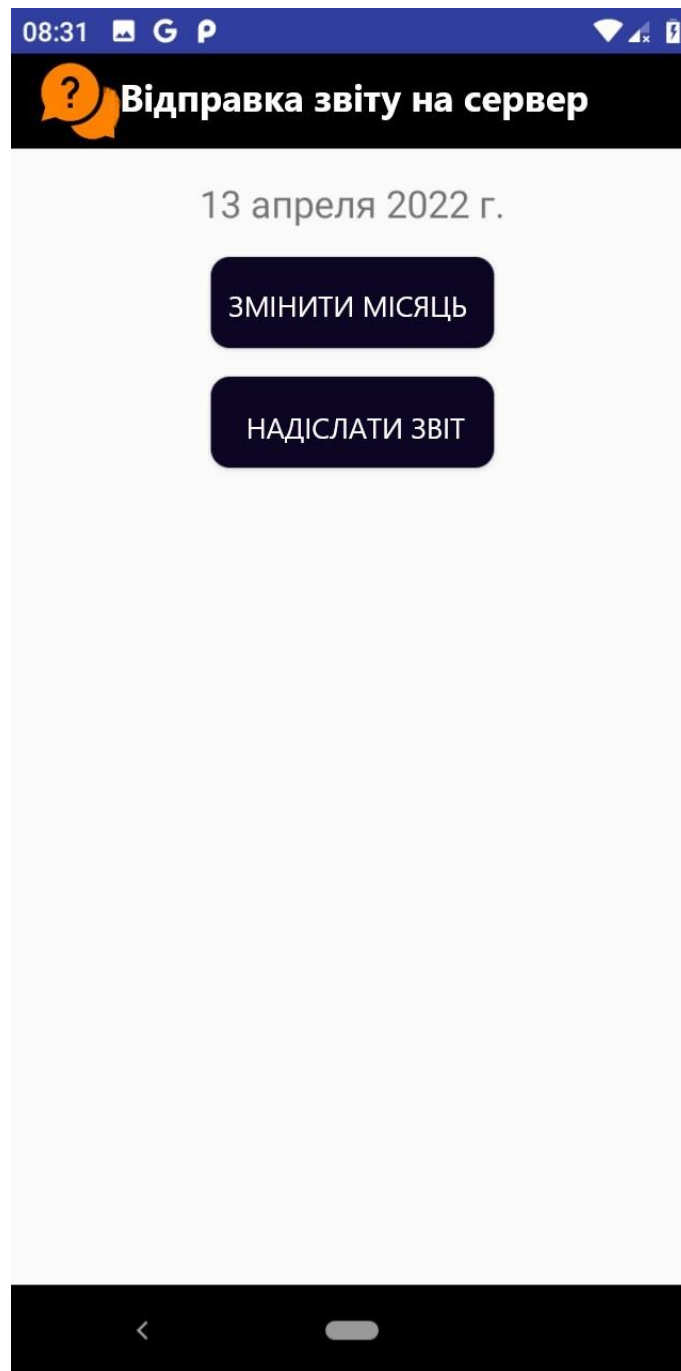


Рисунок 4.17 – Сторінка «Відправка звіту на сервер»

За замовчуванням текстове поле містить поточну дату, натискання на кнопку «Змінити місяць» відкриє елемент керування «Календар», приклад якого наведено на рис. 4.18. Мова спілкування таких елементів керування співпадає з основною мовою пристрою, де встановлено застосунок.

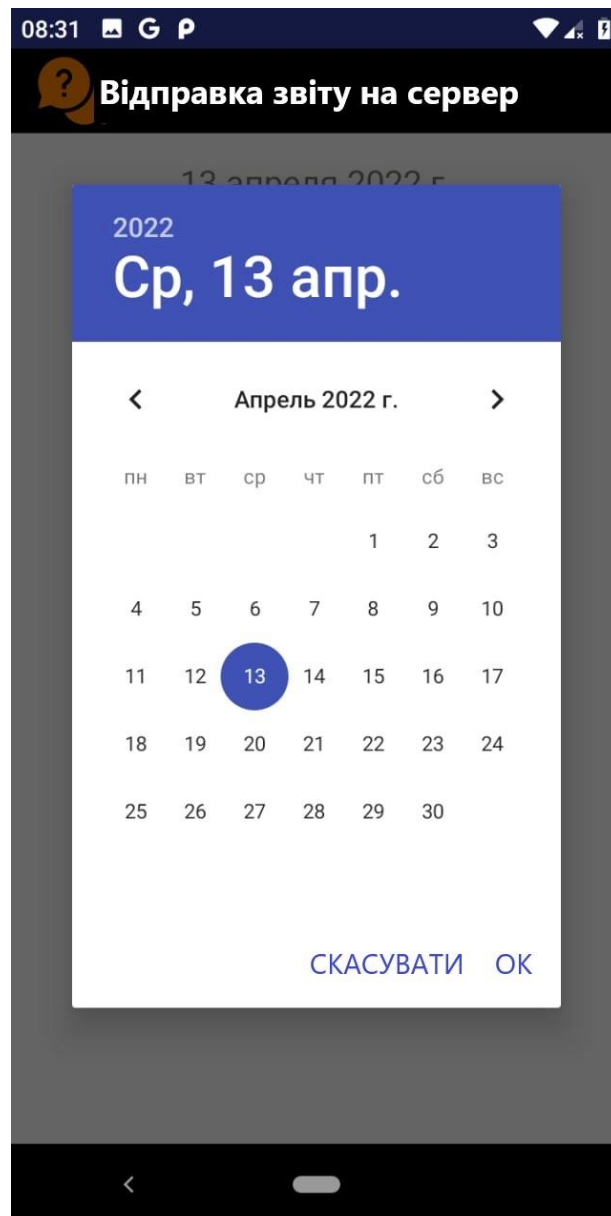


Рисунок 4.18 – Встановлення місяця при відправці місячного звіту на сервер

Натискання на кнопку «Надіслати звіт» сторінки, що наведено на рис. 4.16, призведе до формування та відправки місячного звіту на сервер відповідно.

Натискання на кнопку «База даних» сторінки основного меню користувача приведе до відкриття відповідної сторінки інтерфейсу, де наведено перелік усіх облікових записів користувачів, приклад сторінки наведено на рис. 4.19. Мова, якою будуть подані особисті дані працівників залежить від мови заповнення даних відділом кадрів.

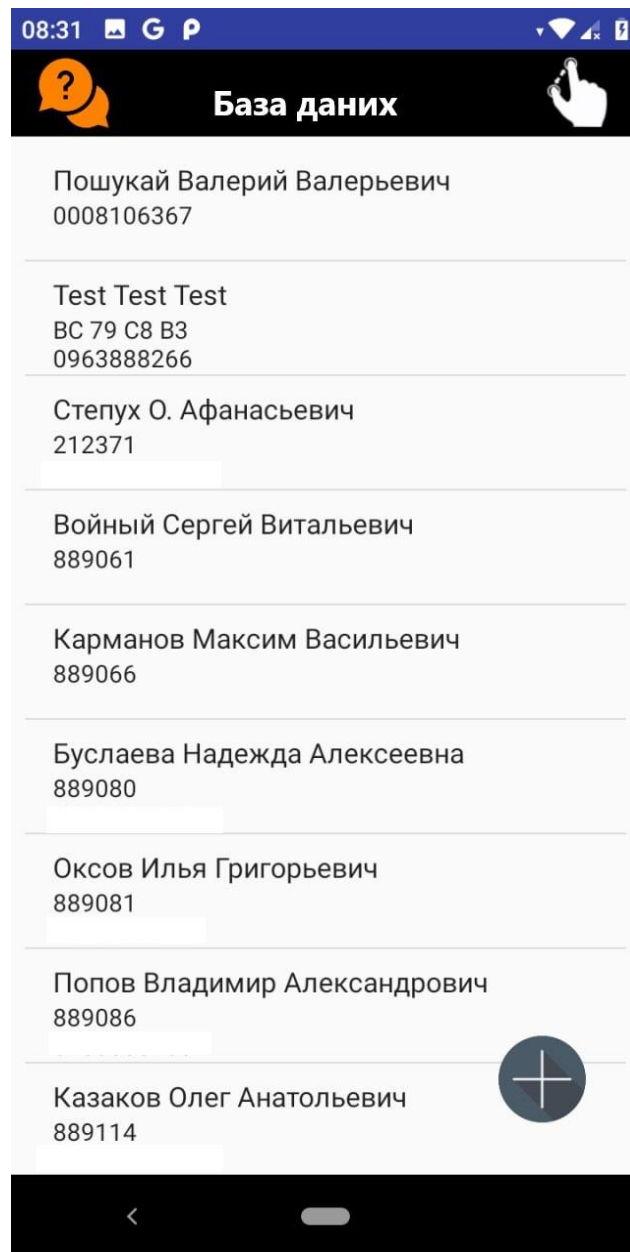


Рисунок 4.19 – Сторінка «База даних»

Натискання на кнопку у правому нижньому куті із зображенням «+» відкриє сторінку створення нового облікового запису, приклад наведено на рис. 4.20.

The screenshot shows a mobile application interface with a dark blue header. The header contains the time '08:32', signal and battery icons, a logo with three people, the title 'Додати співробітника' (Add employee), and a help icon. Below the header is a form with several input fields: 'Зчитайте номер перепустки...' (Read the license number...), 'Прізвище' (Surname), 'Ім'я' (Name), and 'По-батькові' (Patronymic). Below these fields are two dropdown menus: 'ТОВ «МІКС-ІНВЕСТ»' (JSC 'MIX-INVEST') and 'Підсобний робітник' (Assistant worker). At the bottom of the form are two large dark blue buttons: 'Перевірити' (Check) and 'Зберегти' (Save). The bottom of the screen shows a black navigation bar with a back arrow and a home indicator.

Рисунок 4.20 – Сторінка створення нового облікового запису

Заповнивши форму необхідними даними і натиснувши на кнопку «Перевірити», користувач може отримати результати перевірки коректності введених ним даних, приклад наведено на рис. 4.21.

08:32 G P

Додати співробітника

BC 79 C8 B3

Test

Test

Test

ТОВ «АЛД інжиніринг та будівни..»

Підсобний робітник

Перевірити

Користувач з даним табельним номером вже існує: BC 79 C8 B3 Test Test Test

Зберегти

Рисунок 4.21 – Результати перевірки при створенні нового облікового запису

Відповідно, користувач може створити обліковий запис за умови, якщо знайдено лише частковий збіг у даних, які стосуються ПП, номер перепустки має бути унікальним у системі.

5 ТЕСТУВАННЯ ТА ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ

5.1 Опис застосування

Найменування продукту – мобільний застосунок “Trackers”.

Програмний продукт призначений для фіксації робочого часу працівників підприємства. Область застосування – підприємства, що мають розгалужену структуру об’єктів, де виконуються роботи.

Дана розробка дає можливість користувачу створити обліковий запис, зафіксувати початок та завершення робочої зміни, переглянути дані облікового запису, дозвільні документи. Також користувач може передати щоденний та щомісячний звіти на сервер, залишити звернення до технічної підтримки.

5.2 Керівництво користувача

Для початку роботи з застосунком користувач має його запустити шляхом натискання на відповідну іконку.

Для створення облікового запису користувач має перейти до розділу «База даних» основного меню. Потрапивши на сторінку «База даних» користувач може переглянути перелік актуальних облікових записів. Натискання на кнопку «+» праворуч знизу приведе до відкриття сторінки створення облікового запису. Перед зчитуванням перепустки користувач має під’єднати портативний зчитувач до пристрою, якщо планується сканувати RFID-перепустку не терміналом, а мобільним телефоном. Поле номеру перепустки заповнюється автоматично за результатами зчитування, для зчитування перепустку необхідно піднести на відстань до 5 сантиметрів до зчитувача. Далі користувач має заповнити форму своїми даними ПП, обрати структурний підрозділ та спеціальність з випадаючого списку. Після заповнення форми користувач має натиснути на кнопку «Перевірка», результати перевірки з’являться на екрані. Користувач не може створити

обліковий запис з номером перепустки, що вже використовується. Користувач не може створити обліковий запис з даними ППП, що вже використовуються. Після успішної перевірки натискання на кнопку «Готово» призведе до збереження даних нового облікового запису.

Тільки після створення облікового запису користувач може фіксувати робочий час. Для фіксації початку робочої зміни користувачу необхідно перейти до розділу «Авторизація» -> «Передача перепустки». Якщо обрано спосіб фіксації – RFID (для мобільного телефону обов’язкове попереднє під’єднання портативного зчитувача), перепуску необхідно піднести до зчитувача на відстань до 5 сантиметрів, якщо використовується QR, перепустку потрібно помістити у область видимості вікна камери пристрою та дочекатися розпізнавання. У разі успішної авторизації користувач має побачити дані свого ППП та дату і час фіксації на екрані. Натискання на кнопку «Готово» призведе до фіксації початку робочої зміни за умові відсутності помилок, фон сторінки буде змінено на зелений. Користувач може отримати наступні повідомлення про помилки:

- «Перепустка була видана на зміну» – у поточному обліковому періоді вже відбулась фіксація початку робочої зміни за цією перепусткою;
- «Користувача не знайдено» – відсутній обліковий запис, що закріплений за даною перепусткою.

У разі наявності помилок під час фіксації користувач отримує спливаюче повідомлення з текстом помилки, фон сторінки стає червоним.

Для фіксації завершення робочої зміни користувачу необхідно перейти до розділу «Авторизація» -> «Повернення перепустки». Якщо обрано спосіб фіксації – RFID (для мобільного телефону обов’язкове попереднє під’єднання портативного зчитувача), перепуску необхідно піднести до зчитувача на відстань до 5 сантиметрів, якщо використовується QR, перепустку потрібно помістити у область видимості вікна камери пристрою та дочекатися розпізнавання. У разі успішної авторизації користувач має побачити дані ППП власника перепустки та дату і час фіксації

на екрані. Також користувачу доступна шкала нарахування відповідному співробітнику премії за день. Встановлення значення відбувається шляхом перетаскування повзунка. Натискання на кнопку «Готово» призведе до фіксації завершення робочої зміни за умові відсутності помилок, фон сторінки буде змінено на зелений. Користувач може отримати наступні повідомлення про помилки:

- «Перепустка на зміну не видавалась або вже була здана» – у поточному обліковому періоді за даною перепусткою відсутнє відкриття зміни, або зміну вже було закрито;
- «Користувача не знайдено» – відсутній обліковий запис, що закріплений за даною перепусткою.

У разі наявності помилок під час фіксації користувач отримує спливаюче повідомлення з текстом помилки, фон сторінки стає червоним.

Після фіксації початку робочої зміни можна переглянути перелік співробітників, які знаходяться на зміні у даний момент часу, для цього необхідно перейти до розділу «Авторизація» -> «Звіт». Вікно сторінки «Співробітники на зміні» дозволяє отримати дані ППП, номер перепустки, дату та час початку робочої зміни. Відповідно при фіксації завершення робочої зміни обліковий запис користувача зникає з цього переліку. Натискання на пункт переліку з шуканим обліковим записом призведе до відкриття картки облікового запису.

На сторінці «Картка співробітника» можна знайти наступну інформацію про власника перепустки:

- назва структурного підрозділу;
- ППП;
- посада;
- контактний номер телефону;
- дата працевлаштування.

Також тут можна переглянути актуальне фото співробітника, а натискання на кнопку з іконкою документів праворуч унизу призведе до переходу на сторінку «Документи співробітника».

Сторінка «Документи співробітника» містить дані ППП обраного співробітника та перелік назви документів і термінів їх дії, які закріплені за ним. Натискання на обрану назву документа призведе до переходу на сторінку «Документ працівника». Сторінка містить дані ППП працівника, назву та термін дії обраного документа, область перегляду самого документа, також можлива зміна масштабу шляхом діагональних стискаючих або розтискаючих свайпів по документу, можливий перегляд багатосторінкових документів.

Шапка застосунку містить іконки-кнопки. Натискання на кнопку з зображенням курсору на сторінці «Авторизація» призведе до ручної передачі щоденного звіту на сервер. Натискання на кнопку з зображенням папірового літака призведе до переходу на сторінку передачі щомісячного звіту. Сторінка «Відправка звіту на сервер» містить текстове поле з обраною датою кнопку «Змінити місяць», натискання на яку дозволить обрати місяць, кнопку «Відправити звіт», що формує на надсилає звіт за обраний місяць.

На будь-якій сторінці натискання на кнопку з іконкою знаку запитання у шапці призведе до переходу на сторінку технічної підтримки. Сторінка дозволяє сформулювати звернення до технічної підтримки, що складається з теми та тексту звернення, прикріплених або зроблених щойно зображень. Натискання на кнопку «Відправка» призведе до відправки на сервер раніше сформованого звернення.

5.3 Тестування

Прийнято рішення про мануальне тестування роботи всіх сторінок застосунку (функцій користувача) на мобільному телефоні та промисловому терміналі, які застосовуються на підприємстві. У випадку термінала

використано вбудований RFID-зчитувач, для телефону – портативний зчитувач. У обох випадках для сканування QR-коду використано вбудовану камеру пристрою.

У таблиці 5.1 наведено результати тестування програмного продукту. У шапці таблиці наведено пристрій, на якому проводились тести, та тип перепустки, якою проводилось зчитування.

Таблиця 5.1 – Результати тестування програмного продукту

	Термінал, RFID	Термінал, QR	Телефон, RFID	Телефон, QR
Створення нового облікового запису	+	+	+	+
Перегляд всіх облікових записів	+	+	+	+
Перегляд облікових записів користувачів на зміні	+	+	+	+
Фіксація початку робочої зміни	+	+	+	+
Фіксація завершення робочої зміни	+	+	+	+
Передача щоденного звіту на сервер	+	+	+	+
Передача щомісячного звіту на сервер	+	+	+	+
Перегляд картки співробітника	+	+	+	+
Перегляд переліку документів співробітника	+	+	+	+
Перегляд дозвільних документів співробітника	+	+	+	+

Для користувача вважається успішним результатом перевірки повний доступ до відповідних функцій, прогнозоване відтворення результатів дій користувача, також успішним результатом перевірки є обробка помилкових дій користувача.

За результатами тестування програмне забезпечення працює стабільно та коректно, всі тести, що були проведені, пройдені успішно.

У таблиці 5.2 наведено порівняння аналогів реалізацій системи фіксації робочого часу, а саме – СКУД, система трекінгу за допомогою «розумних годинників» та розроблена система «Trackers».

Таблиця 5.2 – Порівняння реалізацій системи фіксації робочого часу

Підстава для порівняння	СКУД	Система трекінгу за допомогою «розумних годинників»	«Trackers»
Відносна собівартість	висока	висока	середня
Мобільність системи	статична	мобільна	мобільна
Залежність від погодних умов	незалежна	залежна	незалежна
Достовірність отриманої інформації	умовно достовірна	умовно достовірна	достовірна
Залежність від висот місцевості	незалежна	дуже залежна	незалежна
Поріг навичок для впровадження	середній	високий	середній
Відповідність необхідних технічних засобів наявним	низька	низька	висока
Відповідність необхідних індивідуальних засобів для трекінгу наявним	середня	низька	висока

Згідно з таблицею 5.2 система фіксації робочого часу «Trackers» найкраще відповідає наявним на підприємстві умовам, наявним засобам індивідуальної фіксації, має високу мобільність та незалежна від погоди. Розробка є актуальною, оскільки найкраще відповідає потребам.

5.4 Дослідно-промислова експлуатація

Програмний засіб, який було розроблено в рамках проєкту, було застосовано на базі одного з підприємств. Фіксація робочого часу працівників проводилась щоденно протягом 6 останніх календарних місяців.

Протягом перших двох місяців працівниками відділу кадрів поступово були оцифровані дані про співробітників та їх дозвільні документи. Таким чином за кожним співробітником було закріплено перелік документів, які стали доступні для перегляду у застосунку. Ця функція широко

використовується для надання дозвільних документів на перевірку на об'єктах, де виконуються роботи.

Служба безпеки підприємства займається виготовленням корпоративних перепусток та видачою їх під час працевлаштування. Обліковий запис створюється відділом кадрів у загальному випадку. Також залишено можливість створення облікового запису у застосунку безпосередньо на об'єкті для виняткових ситуацій. Система фіксації використовується на різних структурних підрозділах компанії.

На початковому етапі траплялись випадки людського фактору – вихід на робоче місце без перепустки, нефіксація завершення робочої зміни. Випадки відсутності перепустки ліквідовано наказом за підприємством про недопущення до робіт без перепустки. Випадки невідповідального ставлення до фіксування робочої зміни – на першому місяці експлуатації формуванням засобами стороннього ПЗ щоденних звітів на керівників структурних підрозділів.

За умов виходу на стабільну роботу протягом двох місяців впроваджено погодинну оплату праці працівників за результатами фіксування робочого часу застосунком. Саме у цей період зафіксовано найстабільнішу роботу системи.

Проведено успішну промислово-дослідну експлуатацію застосунку.

ВИСНОВКИ

З метою автоматизації фіксування початку та завершення робочої зміни працівників підприємства є необхідною розробка мобільного застосунку.

Під час розробки застосунку для дипломного проєкту були проаналізовані технології створення схожих систем фіксації робочого часу, які функціонують на інших підприємствах.

Для проєктування були обрані мобільні пристрої під керуванням ОС Android, бо відповідали вимогам. В якості мови програмування було обрано Java, бо мова відповідала основним можливостям та вимогам, що висувалися до мови програмування.

Під час розробки програмного коду були вивчені та застосовані:

- обробники подій інтерфейсу;
- робота з окремими потоками подій;
- робота з динамічною пам'яттю;
- робота з правами відповідно до дозволів застосунку;
- запити до бази даних локальної;
- взаємодія з API сервера.

Створений проєкт відповідає меті своєї розробки, всім вимогам, що були пред'явлені до програмного засобу.

Повний код проєкту наведено у додатку А.

Програмний продукт пройшов всі тести: для тестування було обрано основні функції користувача для випадків роботи з мобільним телефоном та промисловим терміналом, для сканування RFID або QR перепусток.

Передбачено зручний інтерфейс користувача з інтуїтивно зрозумілим потоком подій.

Успішно проведено промислово-дослідне випробовування та впровадження на комерційних установах.

Новизна полягає у портативному нестатичному використанні RFID та QR технологій сканування для широко розповсюджених мобільних телефонів та вузькоспеціалізованих промислових терміналів.

За темою роботи є 1 наукова публікація, а саме тези доповіді на науково-практичній конференції «Сучасні проблеми і досягнення в галузі радіотехніки, телекомунікацій та інформаційних технологій», яка дистанційно проводилася у Національному університеті «Запорізька політехніка» у грудні 2022 року.

Шляхи майбутнього удосконалення:

- авторизація нормувальника;
- доступ до даних усіх пристроїв, що використовують застосунок;
- вдосконалення перегляду документів;
- фіксація місцезнаходження пристрою на момент фіксування робочого часу;
- розширення кола вирішуваних проблем.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Мартиненко, Г.Ю. Дослідження та програмна реалізація методів обліку робочого часу на підприємствах [Електронний ресурс] / Г.Ю. Мартиненко, Т.І. Каплієнко // Сучасні проблеми і досягнення в галузі радіотехніки, телекомунікацій та інформаційних технологій: XI Міжн. наук.-практ. конф., 12–14 грудня 2022 р.: тези доп. / Редкол.: С. В. Морщавка (відпов. ред.). Електрон. дані. – Запоріжжя: НУ «Запорізька політехніка», 2022. – подано до друку.
2. Techopedia Explains Data Collection System (DCS) [Electronic resource]. – Access mode: <https://www.techopedia.com/definition/11311/data-collection-system-dcs#techopedia-explains-data-collection-system-dcs>.
3. Планування та проектування систем збору даних [Електронний ресурс]. – Режим доступу: https://www.rita.dot.gov/bts/sites/rita.dot.gov/bts/files/subject_areas/statistical_policy_and_research/bts_statistical_standards_manual/html/chapter_02.html.
4. Surveys and Data Collection Systems [Electronic resource]. – Access mode: <https://www.cdc.gov/nchs/surveys.html>.
5. Create data forms using SharePoint Designer [Electronic resource]. – Access mode: <https://support.microsoft.com/en-us/office/create-data-forms-using-sharepoint-designer-5b5e3970-af22-45d5-a796-edfe7dda15f6?ui=en-us&rs=en-us&ad=us>.
6. Multiforms: Multi-Page Collection of Data from Users [Electronic resource]. – Access mode: <https://www.drupal.org/node/145576>.
7. Data Collection Software [Electronic resource]. – Access mode: <https://www.getapp.com/customer-management-software/data-collection/>.
8. Survey Data Collection [Electronic resource]. – Access mode: <https://www.norc.umd.edu/research/capabilities/pages/data-collection-and-management/survey-data-collection.aspx>.
9. Використання системи збору даних [Електронний ресурс]. – Режим доступу:

<https://surveys.nces.ed.gov/ipeds/ViewContent.aspx?contentId=16>.

10. Як збирати дані [Електронний ресурс]. – Режим доступу: <http://cvquality.acc.org/en/NCDR-Home/Data-Collection/How-to-Collect-Data.aspx>.

11. eRegistries: Electronic registries for maternal and child health [Electronic resource]. – Access mode: <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4721069/>.

12. Pace, W. Electronic Data Collection Options for Practice-Based Research Networks [Text] / W. Pace, E. Staton. – Ann Fam Med, 2005.

13. Управління даними для підвищення ефективності [Електронний ресурс]. – Режим доступу: <https://www.hrsa.gov/sites/default/files/quality/toolbox/508pdfs/managingdataperformanceimprovement.pdf>.

14. Collecting and Reporting Data for Performance Measurement: Moving Toward Alignment [Electronic resource]. – Access mode: <https://bok.ahima.org/PdfView?oid=70430>.

15. Quiz [Electronic resource]. – Access mode: <https://www.drupal.org/project/quiz>.

16. Інтернет-програма QuizBuilder, створена за допомогою Laravel [Electronic resource]. – Access mode: <http://webxity.com/portfolios/online-quizbuilder-web-app-built-with-laravel/>.

17. Технології радіоміток [Електронний ресурс]. – Режим доступу: <https://www.rfidjournal.com/>.

18. Приклади використання RFID систем в побуті та на виробництві [Електронний ресурс]. – Режим доступу: <https://web.archive.org/web/20140606214710/http://asupro.com/gps-gsm/system/>.

19. Викристання RFID в корпоративній сфері та промисловості [Електронний ресурс]. – Режим доступу: <https://youtu.be/APGPLH3FCQM>.

20. Історія QR-кодів та їх використання в комунікаціях [Електронний ресурс]. – Режим доступу: <https://bazilik.media/istoriia-qr-kodiv-ta-ikh-vykorystannia-v-komunikatsiiakh/>.

21. Об'єктно-орієнтоване програмування. Мова Java [Електронний ресурс]. – Режим доступу: https://dut.edu.ua/uploads/1_1216_25218115.pdf.
22. Smart [Electronic resource]. – Access mode: <https://all-ok.com.ua/xiaomi-mi-smart-band-4-black-global-version>.
23. Idterminal [Electronic resource]. – Access mode: <https://idterminal.com/ua/p17643706-turniket-tiso-centurion.html>.

ДОДАТОК А
Текст програми

TrackersActivity.java

```

package myhome.control;
/*підключення модулів та бібліотек
  android.Manifest;android.app.Activity;android.content.ContentValues;android.content.Context;
  android.content.Intent;android.content.IntentSender;android.content.pm.PackageManager;android.databases
  e.Cursor;android.database.sqlite.SQLiteDatabase;android.device.ScanDevice;android.graphics.PointF;and
  roid.location.Location;android.location.LocationListener;android.location.LocationManager;android.os.A
  syncTask;android.os.Bundle;android.os.Handler;androidx.annotation.Nullable;androidx.constraintlayout.
  widget.ConstraintLayout;androidx.core.app.ActivityCompat;androidx.appcompat.app.AppCompatActivity;
  androidx.appcompat.widget.Toolbar;android.text.Editable;android.text.TextWatcher;android.util.Log;a
  ndroid.view.KeyEvent;android.view.View;android.view.inputmethod.InputMethodManager;android.widg
  et.Button;android.widget.EditText;android.widget.TextView;android.widget.Toast;com.dlazarov66.qrcode
  readerview.QRCodeReaderView;java.io.ByteArrayOutputStream;java.io.IOException;java.io.InputStrea
  m;java.io.OutputStream;java.math.BigInteger;java.net.HttpURLConnection;java.net.MalformedURLException;
  java.net.URL;java.security.MessageDigest;java.security.NoSuchAlgorithmException;java.text.Sim
  pleDateFormat;java.util.Date;
  */

  public class TrackersActivity extends AppCompatActivity implements
  QRCodeReaderView.OnQRCodeReadListener {
      //екземпляри класів для роботи з бд
      private DataBaseHelper helper;
      private SQLiteDatabase db;
      //змінні для проміжних даних користувача
      String patronymic, name, surname, code_employee, imei, testString, track_smena;

      int i = 1, len1 = 1, j = 0, kol;
      int p1 = 1, p2 = 1;
      String analiz, teststring1, analiz1;
      //екземпляр класу для роботи з пристроєм
      ScanDevice sm;
      //змінна-ключ для роботи з сканером пристрою
      private final static String SCAN_ACTION = "scan.rcv.message";
      //змінна для зберігання проміжних даних ключа qr-коду
      private String barcodeStr;
      //елементи інтерфейсу для керування подіями застосунку
      EditText showScanResult;
      TextView name_employee, textView, tab1;
      Button btnSend;
      ConstraintLayout cl;
      private QRCodeReaderView qrCodeReaderView;

      //змінна для проміжного значення qr-коду
      private String qrDecoding;

      // permission request codes need to be < 256
      private static final int RC_HANDLE_CAMERA_PERM = 2;

      //-----
      private LocationManager locationManager;
      //-----

      Double latitude;
      Double longitude;

      //-----

```

//метод викликається при старті активіті, містить ініціалізацію елементів інтерфейсу та слухачі подій

```

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_trackers);

    latitude = 0.0;
    longitude = 0.0;
    //-----
    locationManager = (LocationManager) getSystemService(LOCATION_SERVICE);

    Button exit= (Button)findViewById(R.id.button);
    exit.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            Intent intent = new Intent(TrackersActivity.this, RegistrationActivity.class);
            startActivity(intent);
        }
    });

    helper = new DataBaseHelper(this);
    db = helper.getWritableDatabase();

    cl=(ConstraintLayout)findViewById(R.id.cl);
    name_employee = (TextView) findViewById(R.id.name_employee);
    btnSend = (Button) findViewById(R.id.save);
    textView = (TextView) findViewById(R.id.textView);
    tab1 = (TextView) findViewById(R.id.workshop);

    textView.setText("Відскануйте перепустку");
    showScanResult = (EditText) findViewById(R.id.tab1);

    showScanResult.setOnEditorActionListener(new TextView.OnEditorActionListener() {
        public boolean onEditorAction(TextView v, int actionId, KeyEvent event) {
            if (event != null&& (event.getKeyCode() == KeyEvent.KEYCODE_ENTER)) {
                InputMethodManager in = (InputMethodManager)
getSystemService(Context.INPUT_METHOD_SERVICE);

in.hideSoftInputFromWindow(showScanResult.getApplicationWindowToken(),InputMethodManager.HIDE_NOT_ALWAYS);
            }
            return false;
        }
    });

    showScanResult.addTextChangedListener(new TextWatcher() {
        public void afterTextChanged(Editable s) {
            String testString8 = showScanResult.getText().toString();
            Log.d("MyLog123456", "testString8after - " + testString8);
            if (testString8.length() < 11) {
            } else {
                Log.d("MyLog", "afterTextChangedInput");
            }
        }
    });

```

```

textView.setText(track_smena);

SimpleDateFormat dateFormat = new SimpleDateFormat("dd.MM.yyyy HH:mm");
strDate = dateFormat.format(new Date());
tab1.setText(strDate);

Button save = (Button) findViewById(R.id.save);
save.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {

        SQLiteDatabase db = helper.getWritableDatabase();

        Class3 cls3 = new Class3();
        cls3.output();

        if (analiz.equals("1")) {
            cl.setBackgroundResource(R.color.red);
            Handler handler = new Handler();
            handler.postDelayed(new Runnable() {
                public void run() {
                    cl.setBackgroundResource(R.color.fon);
                    Toast toast = Toast.makeText(TrackersActivity.this,
                        "Перепустку було видано на зміну",
                        Toast.LENGTH_LONG);
                    toast.show();
                }
            }, 3000);
        } else {
            if (track_smena.equals("Немає даних про перепустку")) {
                cl.setBackgroundResource(R.color.red);
                Handler handler = new Handler();
                handler.postDelayed(new Runnable() {
                    public void run() {
                        cl.setBackgroundResource(R.color.fon);
                        Toast toast = Toast.makeText(TrackersActivity.this,
                            "Немає даних про перепустку",
                            Toast.LENGTH_LONG);
                        toast.show();
                    }
                }, 3000);
            } else {

                if (latitude > 0.0 & longitude > 0.0) { //if here coordinates

                    ContentValues contentValues = new ContentValues();
                    contentValues.put(DataBaseHelper.COLUM_1, imei);
                    contentValues.put(DataBaseHelper.COLUM_2, track_smena);
                    contentValues.put(DataBaseHelper.COLUM_3, code_employee);
                    SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd
HH:mm:ss");

                    String strDate = sdf.format(new Date());

```



```

        teststring1 = testString.substring(0, 10);

        Class2 cls2 = new Class2();
        cls2.output();
    } else {
        teststring1 = "";
        analiz = "";
        track_smena = "Немає даних про перепустку";
    }

    }
}
});
Log.d("MyLog", "onCreate");

Button cancel= (Button)findViewById(R.id.Cancel);
cancel.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        Intent intent = new Intent(TrackersActivity.this, RegistrationActivity.class);
        startActivity(intent);
    }
});

start();

}

//метод роботи з qr-кодом, також преевірка дозволів застосунку
public void start(){
    int rc = ActivityCompat.checkSelfPermission(this, Manifest.permission.CAMERA);
    if (rc == PackageManager.PERMISSION_GRANTED) {
        qrDecoding = null;
        qrCodeReaderView = (QRCodeReaderView) findViewById(R.id.qrdecoderview);

        //встановлення слухача на читання коду
        qrCodeReaderView.setOnQRCodeReadListener(this);

        // функція для активації та деактивації читання
        qrCodeReaderView.setQRDecodingEnabled(true);

        // Функція інтервалу автофокусу
        qrCodeReaderView.setAutofocusInterval(1000L);

        // функція для активація ліхтаря
        qrCodeReaderView.setTorchEnabled(true);

        // функція активації фронтальної камери
        qrCodeReaderView.setFrontCamera();

        // функція повернення на основну камеру
        qrCodeReaderView.setBackCamera();
        //-----
    } else {
        requestCameraPermission();
    }
}
}

```

```

//метод перевірки дозволів для роботи з камерою пристрою
private void requestCameraPermission() {
    Log.w("MyLog", "Camera permission is not granted. Requesting permission");

    final String[] permissions = new String[]{Manifest.permission.CAMERA};

    if (!ActivityCompat.shouldShowRequestPermissionRationale(this,
        Manifest.permission.CAMERA)) {
        ActivityCompat.requestPermissions(this, permissions,
RC_HANDLE_CAMERA_PERM);
        return;
    }

    final Activity thisActivity = this;

    View.OnClickListener listener = new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            ActivityCompat.requestPermissions(thisActivity, permissions,
                RC_HANDLE_CAMERA_PERM);
        }
    };

}

//-----
// викликається, коли код декодовано
// "text" : розкодована строка
// "points" : точки контролю простору декодування
@Override
public void onQRCodeRead(String text, PointF[] points) {
    Log.d("MyLog", "onQRCodeRead");
    qrDecoding = text.trim();
    qrCodeReaderView.setQRDecodingEnabled(false);
    String testString8 = qrDecoding;
    Log.d("MyLog", "testString8 - " + testString8);
    if (testString8.length() < 47) {
    } else {
        Log.d("MyLog", "11111testString8 - " + testString8);
        String[] words = testString8.split(" // ");
        String md5input = words[0];
        md5input = md5input.toLowerCase();
        Log.d("MyLog", "md5input - " + md5input);
        testString = words[1];
        String md5mustbe = md5Custom(testString);
        Log.d("MyLog", "md5mustbe - " + md5mustbe);
        if (md5input.equals(md5mustbe)) {

            Log.d("MyLog", "testString - " + testString);
            teststring1 = "";
            len1 = testString.length();
            Log.d("MyLog", "len1 - " + len1);
            if (len1 > 5) {
                analiz = testString.substring(0, 5);
            }
        }
    }
}

```

```

        teststring1 = testString.substring(5, len1);
        Log.d("kop", "teststring1 = "+teststring1);

        Class2 cls2 = new Class2();
        cls2.output();
    } else {
        teststring1 = "";
        analiz = "";
        track_smena = "Немає даних про перепустку";
        qrCodeReaderView.setQRDecodingEnabled(true);
    }
} else {
    Toast toast = Toast.makeText(TrackersActivity.this,
        "Невірний формат коду",
        Toast.LENGTH_SHORT);
    toast.show();
    teststring1 = "";
    analiz = "";
    track_smena = "Немає даних про перепустку";
    qrCodeReaderView.setQRDecodingEnabled(true);
}
}
}

textView.setText(track_smena);

SimpleDateFormat dateFormat = new SimpleDateFormat("dd.MM.yyyy HH:mm");
strDate = dateFormat.format(new Date());
tab1.setText(strDate);

Button save = (Button) findViewById(R.id.save);
save.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {

        SQLiteDatabase db = helper.getWritableDatabase();

        Class3 cls3 = new Class3();
        cls3.output();

        if (analiz.equals("1")) {
            cl.setBackgroundResource(R.color.red);
            Handler handler = new Handler();
            handler.postDelayed(new Runnable() {
                public void run() {
                    cl.setBackgroundResource(R.color.fon);
                    Toast toast = Toast.makeText(TrackersActivity.this,
                        "Перепустку було видано на зміну",
                        Toast.LENGTH_LONG);
                    toast.show();
                }
            }, 1000);
        }
    }
});

```

```

qrCodeReaderView.setQRDecodingEnabled(true);

} else {
    if (track_smena.equals("Немає даних про перепустку")) {
        cl.setBackgroundResource(R.color.red);
        Handler handler = new Handler();
        handler.postDelayed(new Runnable() {
            public void run() {
                cl.setBackgroundResource(R.color.fon);
                Toast toast = Toast.makeText(TrackersActivity.this,
                    "Немає даних про перепустку",
                    Toast.LENGTH_LONG);
                toast.show();
            }
        }, 1000);
        qrCodeReaderView.setQRDecodingEnabled(true);
    } else {

        //here need check to coordinates

        if(latitude > 0.0 & longitude > 0.0){

            ContentValues contentValues = new ContentValues();
            contentValues.put(DataBaseHelper.COLUM_1, imei);
            contentValues.put(DataBaseHelper.COLUM_2, track_smena);
            contentValues.put(DataBaseHelper.COLUM_3, code_employee);
            SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss");
            String strDate = sdf.format(new Date());
            contentValues.put(DataBaseHelper.COLUM_4, strDate);
            contentValues.put(DataBaseHelper.COLUM_10, latitude);
            contentValues.put(DataBaseHelper.COLUM_11, longitude);
            db.insert(DataBaseHelper.TABLE_NAME_SAVE, null, contentValues);
            cl.setBackgroundResource(R.color.green);
            Handler handler = new Handler();
            handler.postDelayed(new Runnable() {
                public void run() {
                    qrDecoding = null;
                    cl.setBackgroundResource(R.color.fon);
                    qrCodeReaderView.setQRDecodingEnabled(true);
                    textView.setText("Відскануйте перепустку");
                    tab1.setText("");
                }
            }, 1000);

        } else {
            cl.setBackgroundResource(R.color.red);
            Handler handler = new Handler();
            handler.postDelayed(new Runnable() {
                public void run() {
                    cl.setBackgroundResource(R.color.fon);
                    Toast toast = Toast.makeText(TrackersActivity.this,
                        "Неможливо визначити місцезнаходження",
                        Toast.LENGTH_LONG);
                    toast.show();
                }
            }

```

```

        }, 3000);
    }

    }

    });

    qrCodeReaderView.setQRDecodingEnabled(true);

}

@Override
protected void onResume() {
    super.onResume();
    Log.d("MyLog", "onResume");
    qrCodeReaderView.startCamera();

    //-----
    if (ActivityCompat.checkSelfPermission(this,
Manifest.permission.ACCESS_FINE_LOCATION) != PackageManager.PERMISSION_GRANTED &&
ActivityCompat.checkSelfPermission(this, Manifest.permission.ACCESS_COARSE_LOCATION) !=
PackageManager.PERMISSION_GRANTED) {
        Log.d("MyLog", "here");
    }
    locationManager.requestLocationUpdates(LocationManager.GPS_PROVIDER,
        1000 * 1, 1, locationListener);
    locationManager.requestLocationUpdates(
        LocationManager.NETWORK_PROVIDER, 1000 * 1, 1,
        locationListener);
    checkEnabled();

}

//метод обробки призупинення роботи
@Override
protected void onPause() {
    super.onPause();
    Log.d("MyLog", "onPause");
    qrCodeReaderView.stopCamera();
    //-----
    locationManager.removeUpdates(locationListener);
    //-----
}

//-----

private LocationListener locationListener = new LocationListener() {

    @Override
    public void onLocationChanged(Location location) {
        showLocation(location);
    }
}

```

```

@Override
public void onProviderDisabled(String provider) {
    checkEnabled();
}

@Override
public void onProviderEnabled(String provider) {
    checkEnabled();
    if (ActivityCompat.checkSelfPermission(TrackersActivity.this,
Manifest.permission.ACCESS_FINE_LOCATION) != PackageManager.PERMISSION_GRANTED &&
ActivityCompat.checkSelfPermission(TrackersActivity.this,
Manifest.permission.ACCESS_COARSE_LOCATION) !=
PackageManager.PERMISSION_GRANTED) {
        return;
    }
    showLocation(locationManager.getLastKnownLocation(provider));
}

};

private void showLocation(Location location) {
    String str="";
    if (location == null) {
        Log.d("MyLog", "location-null");
        return;
    }
    if (location.getProvider().equals(LocationManager.GPS_PROVIDER)) {
        str = formatLocation(location);
    } else if (location.getProvider().equals(
        LocationManager.NETWORK_PROVIDER)) {
        str = formatLocation(location);
    }
}

private String formatLocation(Location location) {
    if (location == null){
        Log.d("MyLog", "location-null");
        return "";
    }
    latitude = location.getLatitude();
    longitude = location.getLongitude();
    cl.setBackgroundResource(R.color.red);

    cl.setBackgroundResource(R.color.fon);

    return String.format(
        "Coordinates: lat = %1$.8f, lon = %2$.8f",
        location.getLatitude(), location.getLongitude());
}

//for button - go to gps-settings
public void onClickLocationSettings(View view) {
    startActivity(new Intent(
        android.provider.Settings.ACTION_LOCATION_SOURCE_SETTINGS));
}

```

```

};

//-----

//метод мд5 шифрування
public static String md5Custom(String st) {
    MessageDigest messageDigest = null;
    byte[] digest = new byte[0];

    try {
        messageDigest = MessageDigest.getInstance("MD5");
        messageDigest.reset();
        messageDigest.update(st.getBytes());
        digest = messageDigest.digest();
    } catch (NoSuchAlgorithmException e) {

        e.printStackTrace();
    }

    BigInteger bigInt = new BigInteger(1, digest);
    String md5Hex = bigInt.toString(16);

    while( md5Hex.length() < 32 ){
        md5Hex = "0" + md5Hex;
    }

    return md5Hex;
}

//перевірка наявності шуканого облікового запису
public class Class2 {
    public void output() {

        Cursor cursor = db.query("passport", null, DataBaseHelper.COLUM_1 + "=?", new
String[]{teststring1}, null,
        null, null, null);
        if (cursor.getCount() == 0) {
            track_smena = "Немає даних про перепуску";
        } else {

            cursor.moveToFirst();
            while (!cursor.isAfterLast()) {

                imei = cursor.getString(1);
                track_smena = cursor.getString(2)+' '+cursor.getString(3)+' '+cursor.getString(4);
                code_employee = cursor.getString(4);
                SimpleDateFormat dateFormat = new SimpleDateFormat("yyyy-MM-dd
hh:mm:ss");

                cursor.moveToNext();
            }
            cursor.close();
        }
    }
}

```

```

    }
}

//робота з бд на предмет робочих змін
public class Class3 {
    public void output() {

        String analiz2="";
        SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd");
        String strDate = sdf.format(new Date());
        Cursor cursor = db.query("otchet", null, " patronumic Like ?", new String[]
{"%" + strDate + "%"}, null,
        null, null, null);

        cursor.moveToFirst();
        while (!cursor.isAfterLast()) {
            analiz2=(cursor.getString(1));
            Log.d("kop", "analiz = "+analiz);
            analiz1=(cursor.getString(5));
            Log.d("kop", "analiz1 = "+analiz1);
            if(teststring1.equals(analiz2) && analiz1==null){
                analiz ="1";
            }
            Log.d("kop", "analizz = "+analiz);

            cursor.moveToNext();
        }
        if(!analiz.equals("1")){
            analiz ="0";
        }
        Log.d("kop", "analizAFTER = "+analiz);
        cursor.close();
    }
}

//відправка щоденного звіту на сервер
class SendData extends AsyncTask<Void, Void, Void> {

    String resultString = null;

    @Override
    protected void onPreExecute() {
        super.onPreExecute();
    }

    @Override
    protected Void doInBackground(Void... params) {
        try {

            Cursor cursor = db.query("otchet", null, null, null,
            null, null, null);
            cursor.moveToFirst();

            while (!cursor.isAfterLast()) {
                imei = cursor.getString(1);
                code_employee = cursor.getString(2);

```

```

String date_up = cursor.getString(4);
String date_down= cursor.getString(3);

String myURL = "https://saas04.alldholding.com/wp-
security/mobile_api/input.php";

String parammetrs =
"imei="+imei+"&code_employee="+code_employee+"&date_up="+date_up+"&date_down="+date_dow
n;

byte[] data = null;
InputStream is = null;

try {
    URL url = new URL(myURL);
    HttpURLConnection conn = (HttpURLConnection) url.openConnection();
    conn.setReadTimeout(10000);
    conn.setConnectTimeout(15000);
    conn.setRequestMethod("POST");
    conn.setRequestProperty("Connection", "Keep-Alive");
    conn.setRequestProperty("Content-Type", "application/x-www-form-
urlencoded");
    conn.setRequestProperty("Content-Length", "" +
Integer.toString(parammetrs.getBytes().length));
    conn.setDoOutput(true);
    conn.setDoInput(true);

    // конвертація строки в UTF-8
    data = parammetrs.getBytes("UTF-8");

    OutputStream os = conn.getOutputStream();

    // передача даних на сервер
    os.write(data);
    os.flush();
    os.close();
    data = null;
    conn.connect();
    int responseCode= conn.getResponseCode();

    // відповідь
    ByteArrayOutputStream baos = new ByteArrayOutputStream();

    if (responseCode == 200) {
        Toast.makeText(getApplicationContext(), "Дані успішно передано",
Toast.LENGTH_SHORT).show();

        // все ОК (відповідь 200)
        is = conn.getInputStream();

```

```

byte[] buffer = new byte[8192]; // розмір буфера

// читання відповіді
int bytesRead;

while ((bytesRead = is.read(buffer)) != -1) {
    baos.write(buffer, 0, bytesRead);
}

data = baos.toByteArray();
resultString = new String(data, "UTF-8"); // "OK"

    } else {

    }
    conn.disconnect();

} catch (MalformedURLException e) {

} catch (IOException e) {

} catch (Exception e) {

}

        cursor.moveToNext();
    } cursor.close();
}
catch (Exception e) {
    e.printStackTrace();
}
return null;
}

@Override
protected void onPostExecute(Void result) {
    super.onPostExecute(result);

    Toast toast = Toast.makeText(getApplicationContext(), "Дані передано",
Toast.LENGTH_SHORT);

    }
}
}

```

```

Trackers_backActivity.java
package myhome.control;

```

```

/*підключення модулів та бібліотек
    android.Manifest;android.app.Activity;android.content.ContentValues;android.content.Context;
    android.content.Intent;android.content.pm.PackageManager;android.database.Cursor;android.database.sql
    ite.SQLiteDatabase;android.device.ScanDevice;android.graphics.PointF;android.location.Location;androi
    d.location.LocationListener;android.location.LocationManager;android.os.AsyncTask;android.os.Build;a
    ndroid.os.Bundle;android.os.Handler;androidx.annotation.RequiresApi;androidx.appcompat.widget.App
    CompatSeekBar;androidx.constraintlayout.widget.ConstraintLayout;androidx.core.app.ActivityCompat;a
    ndroidx.appcompat.app.AppCompatActivity;android.text.Editable;android.text.TextWatcher;android.util.
    Log;android.view.KeyEvent;android.view.View;android.view.inputmethod.InputMethodManager;android
    .widget.Button;android.widget.EditText;android.widget.SeekBar;android.widget.TextView;android.widg
    et.Toast;com.dlazarov66.qrcodereaderview.QRCodeReaderView;java.io.ByteArrayOutputStream;java.io.I
    OException;java.io.InputStream;java.io.OutputStream;java.net.HttpURLConnection;java.net.Malformed
    URLException;java.net.URL;java.text.SimpleDateFormat;java.util.Date;java.util.HashMap;java.util.Map;
    */

```

```

    public class Trackers_backActivity extends AppCompatActivity implements
    QRCodeReaderView.OnQRCodeReadListener {
        //екземпляри класу для роботи з бд
        private DataBaseHelper helper;
        private SQLiteDatabase db;
        //змінні для проміжних даних користувача
        String name,code_employee,imei,testString,track_smena;
        int i=1,len1=1;
        String analiz,teststring1;
        //екземпляр класу для роботи з пристроєм
        ScanDevice sm;

        //ключ-змінна для роботи з сканером
        private final static String SCAN_ACTION = "scan.rcv.message";
        private String barcodeStr;
        //елементи інтерфейсу
        EditText showScanResult;
        TextView name_employee,textView,tab1;
        Button btnSend;
        byte[] barcode;
        String strDate;
        ConstraintLayout cl;
        private String qrDecoding;
        private QRCodeReaderView qrCodeReaderView;

        // permission request codes need to be < 256
        private static final int RC_HANDLE_CAMERA_PERM = 2;

        private LocationManager locationManager;
        //-----

        Double latitude;
        Double longitude;

        //сеекбар для регулювання премії
        SeekBar seekBar;
        TextView tvValueSeek;
        Integer value = null;
        Integer result = null; //here result mark seekbar`s for users
        ConstraintLayout cl_scale_big;

```

public Map<Integer,Integer> mMapValue = new HashMap<Integer, Integer>(); //словник
значень для шкали оцінок

```
@RequiresApi(api = Build.VERSION_CODES.O)
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_trackers);
    cl=(ConstraintLayout)findViewById(R.id.cl);
    cl_scale_big=(ConstraintLayout)findViewById(R.id.cl_scale_big);

    // заповнення словаря для шкали оцінок
    int j=0; //+5
    int k=-100; //+10
    for (int i=0; i<21; i++){
        mMapValue.put(j, k);
        j = j + 5;
        k = k + 10;
    }

    seekBar = findViewById(R.id.seek);
    tvValueSeek = findViewById(R.id.tvValueSeek);
    int i = seekBar.getProgress();
    value = calc(i); //here round to 0-5-10
    Log.d("OOO","value1="+value);
    value = mMapValue.get(value);
    Log.d("OOO","value2="+value);
    result=value;
    tvValueSeek.setText("Значення : " + result);

    seekBar.setOnSeekBarChangeListener(new SeekBar.OnSeekBarChangeListener() {
        @Override public void onProgressChanged(SeekBar seekBar, int i, boolean b) {
            value = calc(i); //here round to 0-5-10
            value = mMapValue.get(value);
            Log.d("OOO","value2="+value);
            result=value;
            tvValueSeek.setText("Значення : " + result); //округлення до десятків
        }

        @Override public void onStartTrackingTouch(SeekBar seekBar) {

        }

        @Override public void onStopTrackingTouch(SeekBar seekBar) {

        }
    });

    latitude = 0.0;
    longitude = 0.0;
    //-----
    locationManager = (LocationManager) getSystemService(LOCATION_SERVICE);
```

```
qrDecoding = null;
qrCodeReaderView = (QRCodeReaderView) findViewById(R.id.qrdecoderview);
```

```
Button exit= (Button)findViewById(R.id.button);
exit.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        Intent intent = new Intent(Trackers_backActivity.this, RegistrationActivity.class);
        startActivity(intent);
    }
});
```

```
helper = new DataBaseHelper(this);
db = helper.getWritableDatabase();
```

```
name_employee = (TextView) findViewById(R.id.name_employee);
btnSend = (Button) findViewById(R.id.save);
textView = (TextView) findViewById(R.id.textView);
tab1 = (TextView) findViewById(R.id.workshop);
```

```
textView.setText("Відскануйте перепустку");
```

```
showScanResult = (EditText) findViewById(R.id.tab1);
```

```
showScanResult.setOnEditorActionListener(new TextView.OnEditorActionListener() {
    public boolean onEditorAction(TextView v, int actionId, KeyEvent event) {
        if (event != null && (event.getKeyCode() == KeyEvent.KEYCODE_ENTER)) {
            InputMethodManager in = (InputMethodManager)
getSystemService(Context.INPUT_METHOD_SERVICE);
```

```
in.hideSoftInputFromWindow(showScanResult.getApplicationWindowToken(), InputMethodManager.HIDE_NOT_ALWAYS);
        }
        return false;
    }
});
```

```
showScanResult.addTextChangedListener(new TextWatcher() {
    public void afterTextChanged(Editable s) {
        String testString8 = showScanResult.getText().toString();
        Log.d("MyLog", "testString8after - " + testString8);
        if (testString8.length() < 11) { //here trouble
        } else {
            textView.setText(track_smena);
            cl_scale_big.setVisibility(View.VISIBLE);

            SimpleDateFormat dateFormat = new SimpleDateFormat("dd.MM.yyyy HH:mm");
            strDate = dateFormat.format(new Date());
            tab1.setText(strDate);
```

```
Button save = (Button) findViewById(R.id.save);
save.setOnClickListener(new View.OnClickListener() {
```

```

@Override
public void onClick(View v) {
    SQLiteDatabase db = helper.getWritableDatabase();
    ContentValues contentValues = new ContentValues();

    Cursor myIMAI = db.query("otchet", null, " IMEI= ? AND code_employee
is null ", new String[]{teststring1}, null,
    null, null, null);
    if (myIMAI.getCount() == 0) {
        cl.setBackgroundResource(R.color.red);
        Handler handler = new Handler();
        handler.postDelayed(new Runnable() {
            public void run() {
                cl.setBackgroundResource(R.color.fon);
                Toast toast = Toast.makeText(Trackers_backActivity.this,
                    "Перепустка на зміну не видавалась або вже була здана",
                    Toast.LENGTH_LONG);
                toast.show();
            }
        } else {

            if (latitude > 0.0 & longitude > 0.0) { //if here coordinates

                SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd
HH:mm:ss");

                String strDate = sdf.format(new Date());
                contentValues.put(DataBaseHelper.COLUMN_5, strDate);
                contentValues.put(DataBaseHelper.COLUMN_12, latitude);
                contentValues.put(DataBaseHelper.COLUMN_13, longitude);
                Log.d("OOO", "result before insert = "+result);
                contentValues.put(DataBaseHelper.COLUMN_14, result);//here mark
working

                db.update(DataBaseHelper.TABLE_NAME_SAVE, contentValues,
                DataBaseHelper.COLUMN_1 + " = ? AND " + "code_employee is null ", new String[]{imei});
                cl.setBackgroundResource(R.color.green);
                Handler handler = new Handler();
                handler.postDelayed(new Runnable() {
                    public void run() {

                        Intent intent = new Intent(Trackers_backActivity.this,
                        Trackers_backActivity.class);

                        startActivity(intent);
                    }
                }, 3000);

            }else {
                cl.setBackgroundResource(R.color.red);
                Handler handler = new Handler();
                handler.postDelayed(new Runnable() {
                    public void run() {
                        cl.setBackgroundResource(R.color.fon);
                        Toast toast = Toast.makeText(Trackers_backActivity.this,
                            "Неможливо визначити місцезоташування",
                            Toast.LENGTH_LONG);

```

```

        toast.show();
    }
    }, 3000);
    }
    }
    }
    });
}

public void beforeTextChanged(CharSequence s, int start, int count, int after) {

}

// при зміні тексту виконується фільтрація
public void onTextChanged(CharSequence s, int start, int before, int count) {
    String testString8 = showScanResult.getText().toString();
    if (testString8.length() < 11) { //here trouble
    } else {

        testString = testString8;

        teststring1 = "";
        len1 = testString.length();
        if (len1 > 10) {
            analiz = testString.substring(0, 5);
            teststring1 = testString.substring(0, 10);
            Class2 cls2 = new Class2();
            cls2.output();
        } else {
            teststring1 = "";
            analiz = "";
            track_smena = "Немає даних про перепустку";

        }

    }
}

});

Button cancel= (Button)findViewById(R.id.Cansel);
cancel.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        Intent intent = new Intent(Trackers_backActivity.this, RegistrationActivity.class);
        startActivity(intent);
    }
});

start();

Log.d("MyLog", "OnCreate");

}

//підрахунок одиниць для шкали
public Integer calc_ed(Integer value){

```

```

Integer result=null;
if(value<10){
    if(value<3){
        result=0;
    }else{
        if(value<7){
            result=5;
        }else{
            result=10;
        }
    }
}
return result;
}

//тут приводяться значення за правилом 3-5, 2-0, 6-5, 7-10
public Integer calc(Integer value){
    Integer result=null;
    if(value>97){
        result=100;
    }else{
        Integer ed = value % 10;
        result = value - ed + calc_ed(ed);
    }
    return result;
}

public void start(){
    int rc = ActivityCompat.checkSelfPermission(this, Manifest.permission.CAMERA);
    if (rc == PackageManager.PERMISSION_GRANTED) {
        qrDecoding = null;
        qrCodeReaderView = (QRCodeReaderView) findViewById(R.id.qrdecoderview);

        //встановлення слухача на читання коду
        qrCodeReaderView.setOnQRCodeReadListener(this);

        // функція для активації та деактивації читання
        qrCodeReaderView.setQRDecodingEnabled(true);

        // Функція інтервалу автофокусу
        qrCodeReaderView.setAutofocusInterval(1000L);

        // функція для активація ліхтаря
        qrCodeReaderView.setTorchEnabled(true);

        // функція активації фронтальної камери
        qrCodeReaderView.setFrontCamera();

        // функція повернення на основну камеру
        qrCodeReaderView.setBackCamera();
        //-----
    }else {
        requestCameraPermission();
    }
}
}

```

```

//метод перевіряє наявність дозволів на роботу з камерою пристрою
private void requestCameraPermission() {
    Log.w("MyLog", "Camera permission is not granted. Requesting permission");

    final String[] permissions = new String[]{Manifest.permission.CAMERA};

    if (!ActivityCompat.shouldShowRequestPermissionRationale(this,
        Manifest.permission.CAMERA)) {
        ActivityCompat.requestPermissions(this, permissions,
RC_HANDLE_CAMERA_PERM);
        return;
    }

    final Activity thisActivity = this;

    View.OnClickListener listener = new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            ActivityCompat.requestPermissions(thisActivity, permissions,
                RC_HANDLE_CAMERA_PERM);
        }
    };
}

//-----
// викликається, коли код декодовано
// "text" : розкодована строка
// "points" : точки контролю простору декодування
@Override
public void onQRCodeRead(String text, PointF[] points) {
    Log.d("MyLog", "onQRCodeRead");
    qrDecoding = text.trim();
    qrCodeReaderView.setQRDecodingEnabled(false);
    String testString8 = qrDecoding;
    if (testString8.length() < 47) {
    } else {
        String[] words = testString8.split(" // ");
        String md5input = words[0];
        md5input = md5input.toLowerCase();
        Log.d("MyLog", "md5input - " + md5input);
        testString = words[1];
        String md5mustbe = TrackersActivity.md5Custom(testString);
        Log.d("MyLog", "md5mustbe - " + md5mustbe);
        if (md5input.equals(md5mustbe)) {

            teststring1 = "";
            len1 = testString.length();
            if (len1 > 5) {
                analiz = testString.substring(0, 5);
                teststring1 = testString.substring(5, len1);
                Class2 cls2 = new Class2();
                cls2.output();
            } else {
                teststring1 = "";
                analiz = "";
            }
        }
    }
}

```

```

        track_smena = "Немає даних про перепустку";
        qrCodeReaderView.setQRDecodingEnabled(true);
    }
} else {
    Toast toast = Toast.makeText(Trackers_backActivity.this,
        "Невірний формат коду",
        Toast.LENGTH_SHORT);
    toast.show();
    teststring1 = "";
    analiz = "";
    track_smena = "Немає даних про перепустку";
    qrCodeReaderView.setQRDecodingEnabled(true);
}
}

textView.setText(track_smena);
cl_scale_big.setVisibility(View.VISIBLE);

SimpleDateFormat dateFormat = new SimpleDateFormat("dd.MM.yyyy HH:mm");
strDate = dateFormat.format(new Date());
tab1.setText(strDate);

Button save = (Button) findViewById(R.id.save);
save.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        SQLiteDatabase db = helper.getWritableDatabase();
        ContentValues contentValues = new ContentValues();

        Cursor myIMAI = db.query("otchet", null, " IMEI= ? AND  code_employee is null
", new String[]{teststring1}, null,
        null, null, null);
        if (myIMAI.getCount() == 0) {
            cl.setBackgroundResource(R.color.red);
            Handler handler = new Handler();
            handler.postDelayed(new Runnable() {
                public void run() {
                    cl.setBackgroundResource(R.color.fon);
                    Toast toast = Toast.makeText(Trackers_backActivity.this,
                        "Перепустка на зміну не видавалась або вже була здана",
                        Toast.LENGTH_LONG);
                    toast.show();
                }
            }, 3000);

            qrCodeReaderView.setQRDecodingEnabled(true);

        } else {

            if (latitude > 0.0 & longitude > 0.0) { //if here coordinates
                SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss");
                String strDate = sdf.format(new Date());
                contentValues.put(DataBaseHelper.COLUM_5, strDate);
                contentValues.put(DataBaseHelper.COLUM_12, latitude);
                contentValues.put(DataBaseHelper.COLUM_13, longitude);
            }
        }
    }
});

```

```

        Log.d("OOO", "result before insert = "+result);
        contentValues.put(DataBaseHelper.COLUMN_14, result);//here mark working
        db.update(DataBaseHelper.TABLE_NAME_SAVE, contentValues,
        DataBaseHelper.COLUMN_1 + " = ? AND " + "code_employee is null ", new String[]{imei});
        cl.setBackgroundResource(R.color.green);
        Handler handler = new Handler();
        handler.postDelayed(new Runnable() {
            public void run() {

                Intent intent = new Intent(Trackers_backActivity.this,
                Trackers_backActivity.class);

                startActivity(intent);
            }
        }, 3000);

        qrCodeReaderView.setQRDecodingEnabled(true);
    }else {
        cl.setBackgroundResource(R.color.red);
        Handler handler = new Handler();
        handler.postDelayed(new Runnable() {
            public void run() {
                cl.setBackgroundResource(R.color.fon);
                Toast toast = Toast.makeText(Trackers_backActivity.this,
                "Неможливо встановити місцезнаходження",
                Toast.LENGTH_LONG);
                toast.show();
            }
        }, 3000);
    }
}
});

qrCodeReaderView.setQRDecodingEnabled(true);
}

@Override
protected void onResume() {
    super.onResume();
    Log.d("MyLog", "onResume");
    qrCodeReaderView.startCamera();

    //-----
    if (ActivityCompat.checkSelfPermission(this,
    Manifest.permission.ACCESS_FINE_LOCATION) != PackageManager.PERMISSION_GRANTED &&
    ActivityCompat.checkSelfPermission(this, Manifest.permission.ACCESS_COARSE_LOCATION) !=
    PackageManager.PERMISSION_GRANTED) {
        Log.d("MyLog", "here");
        return;
    }
    locationManager.requestLocationUpdates(LocationManager.GPS_PROVIDER,
    1000 * 1, 1, locationListener);
    locationManager.requestLocationUpdates(
    LocationManager.NETWORK_PROVIDER, 1000 * 1, 1,
    locationListener);

```

```

        checkEnabled();
    }

    @Override
    protected void onPause() {
        super.onPause();
        Log.d("MyLog", "onPause");
        qrCodeReaderView.stopCamera();
        //-----
        locationManager.removeUpdates(locationListener);
    }

    //-----

    private LocationListener locationListener = new LocationListener() {

        @Override
        public void onLocationChanged(Location location) {
            showLocation(location);
        }

        @Override
        public void onProviderDisabled(String provider) {
            checkEnabled();
        }

        @Override
        public void onProviderEnabled(String provider) {
            checkEnabled();
            if (ActivityCompat.checkSelfPermission(Trackers_backActivity.this,
Manifest.permission.ACCESS_FINE_LOCATION) != PackageManager.PERMISSION_GRANTED &&
ActivityCompat.checkSelfPermission(Trackers_backActivity.this,
Manifest.permission.ACCESS_COARSE_LOCATION) !=
PackageManager.PERMISSION_GRANTED) {
                return;
            }
            showLocation(locationManager.getLastKnownLocation(provider));
        }

        @Override
        public void onStatusChanged(String provider, int status, Bundle extras) {
            /*if (provider.equals(LocationManager.GPS_PROVIDER)) {
                tvStatusGPS.setText("Status: " + String.valueOf(status));
            } else if (provider.equals(LocationManager.NETWORK_PROVIDER)) {
                tvStatusNet.setText("Status: " + String.valueOf(status));
            }*/
        }
    };

    private void showLocation(Location location) {
        String str="";
        if (location == null) {
            Log.d("MyLog", "location-null");
            return;
        }
        if (location.getProvider().equals(LocationManager.GPS_PROVIDER)) {

```

```

        str = formatLocation(location);
    } else if (location.getProvider().equals(
        LocationManager.NETWORK_PROVIDER)) {
        str = formatLocation(location);
    }
}
//робота з локацією
private String formatLocation(Location location) {
    if (location == null){
        Log.d("MyLog", "location-null");
        return "";
    }
    latitude = location.getLatitude();
    longitude = location.getLongitude();
    cl.setBackgroundResource(R.color.red);

    cl.setBackgroundResource(R.color.fon);

    Log.d("MyLog", String.format(
        "Coordinates: lat = %1$.8f, lon = %2$.8f",
        location.getLatitude(), location.getLongitude()));
    return String.format(
        "Coordinates: lat = %1$.8f, lon = %2$.8f",
        location.getLatitude(), location.getLongitude());
}

private void checkEnabled() {
}

//for button - go to gps-settings
public void onClickLocationSettings(View view) {
    startActivity(new Intent(
        android.provider.Settings.ACTION_LOCATION_SOURCE_SETTINGS));
};

//-----

//робота з бд на рахунок перевірки наявності облікового запису
public class Class2 {
    public void output() {

        Cursor cursor = db.query("passport", null, DataBaseHelper.COLUM_1 + "=?", new
String[]{teststring1}, null,
        null, null, null);
        if (cursor.getCount() == 0) {
            track_smena = "Немає даних про перепустку";
        } else {

            cursor.moveToFirst();
            while (!cursor.isAfterLast()) {
                imei = cursor.getString(1);
                track_smena = cursor.getString(2)+' '+cursor.getString(3)+' '+cursor.getString(4);
                code_employee = cursor.getString(4);
            }
        }
    }
}

```

```

SimpleDateFormat dateFormat = new SimpleDateFormat("yyyy-MM-dd
hh:mm:ss");

        cursor.moveToNext();
    }
    cursor.close();
}

}

}

//надсилання даних зміни на сервер
class SendData extends AsyncTask<Void, Void, Void> {

    String resultString = null;

    @Override
    protected void onPreExecute() {
        super.onPreExecute();
    }

    @Override
    protected Void doInBackground(Void... params) {

        Cursor cursor = db.query("otchet", null, null, null,
            null, null, null);
        cursor.moveToFirst();

        while (!cursor.isAfterLast()) {
            imei = cursor.getString(1);
            code_employee = cursor.getString(2);
            String date_up = cursor.getString(4);
            String date_down= cursor.getString(3);

            String myURL = "https://saas04.alldholding.com/wp-
security/mobile_api/input.php";

            String parammetrs =
"imei="+imei+"&code_employee="+code_employee+"&date_up="+date_up+"&date_down="+date_dow
n;

            byte[] data = null;
            InputStream is = null;

            try {
                URL url = new URL(myURL);
                HttpURLConnection conn = (HttpURLConnection) url.openConnection();
                conn.setReadTimeout(10000);
                conn.setConnectTimeout(15000);
                conn.setRequestMethod("POST");
                conn.setRequestProperty("Connection", "Keep-Alive");
                conn.setRequestProperty("Content-Type", "application/x-www-form-
urlencoded");

```

```

        conn.setRequestProperty("Content-Length",
Integer.toString(parammetrs.getBytes().length));
        conn.setDoOutput(true);
        conn.setDoInput(true);

        data = parammetrs.getBytes("UTF-8");

        OutputStream os = conn.getOutputStream();

        os.write(data);
        os.flush();
        os.close();
        data = null;
        conn.connect();
        int responseCode= conn.getResponseCode();

        ByteArrayOutputStream baos = new ByteArrayOutputStream();

        if (responseCode == 200) {
            Toast.makeText(getApplicationContext(), "Дані успішно передано",
Toast.LENGTH_SHORT).show();
            is = conn.getInputStream();

            byte[] buffer = new byte[8192];

            int bytesRead;

            while ((bytesRead = is.read(buffer)) != -1) {
                baos.write(buffer, 0, bytesRead);
            }

            data = baos.toByteArray();
            resultString = new String(data, "UTF-8");

        } else {

        }
        conn.disconnect();

    } catch (MalformedURLException e) {

    } catch (IOException e) {

    } catch (Exception e) {

    }

    cursor.moveToNext();
} cursor.close();

```

```

    }
    catch (Exception e) {
        e.printStackTrace();
    }
    return null;
}

@Override
protected void onPostExecute(Void result) {
    super.onPostExecute(result);

    Toast toast = Toast.makeText(getApplicationContext(), "Дані передано!",
    Toast.LENGTH_SHORT);

    }
}
}

```

```

BDActivity.java
package myhome.control;

/*підключення модулів та бібліотек
android.content.ContentValues;android.content.Context;android.content.Intent;android.content.
SharedPreferences;android.database.Cursor;android.database.sqlite.SQLiteDatabase;android.net.Connecti
vityManager;android.net.NetworkInfo;android.os.AsyncTask;android.os.Bundle;androidx.appcompat.app
.AppCompatActivity;androidx.appcompat.widget.Toolbar;android.util.Log;android.view.View;android.w
idget.ListView;android.widget.SimpleAdapter;android.widget.TextView;android.widget.Toast;org.json.J
SONArray;org.json.JSONException;org.json.JSONObject;java.io.BufferedReader;java.io.IOException;ja
va.io.InputStreamReader;java.net.HttpURLConnection;java.net.URL;java.util.ArrayList;java.util.HashM
ap;java.util.Timer;java.util.TimerTask;
*/

public class BDActivity extends AppCompatActivity {

    //елемент інтерфейсу-перелік
    ListView listView;
    TextView bd;
    //екземпляри класів для роботи з бд
    private DataBaseHelper helper;
    private SQLiteDatabase db;
    //змінні для промідних даних
    String Name = "";
    String IMEI = "";
    ArrayList<HashMap<String, String>> arrayList = new ArrayList<>();
    HashMap<String, String> map;
    //адаптер преліку
    SimpleAdapter adapter;

    //змінні для запуску задач
    private Timer mTimer;
    private MyTimerTask mMyTimerTask;

    //змінна глобальних даних користувача
    SharedPreferences mSettings;

```

```

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_bd);

    mSettings = getSharedPreferences(MainActivity.APP_PREFERENCES,
Context.MODE_PRIVATE);

    helper = new DataBaseHelper(this);
    db = helper.getWritableDatabase();

    listView = (ListView) findViewById(R.id.listView);
    Restart res = new Restart();
    res.bd();
    adapter = new SimpleAdapter(this, arrayList, android.R.layout.simple_list_item_2,
        new String[]{ "IMEI", "Name"},
        new int[]{ android.R.id.text1, android.R.id.text2});
    listView.setAdapter(adapter);

    if (mTimer != null) {
        mTimer.cancel();
    }

    // re-schedule timer here
    // otherwise, IllegalStateException of
    // "TimerTask is scheduled already"
    // will be thrown
    mTimer = new Timer();
    mMyTimerTask = new MyTimerTask();

    mTimer.scheduleAtFixedRate(mMyTimerTask, 1000, 86400000);

}

//перевірка на наявність доступу до мережі в окремому потоці
public void task(View v) {

    // запускаем проверку в новом потоке
    new Thread(new Runnable() {
        @Override
        public void run(){
            if (checkInternet()) {

                getJSON("https://saas04.alldholding.com/wp-
security/mobile_api/trackers_sis_02.php");
            } else {
                Log.d("my_tag", "Немає підключення");
            }
        }
    }).start();
}

// головний метод для перевірки підключення

```

```

public boolean checkInternet() {

    ConnectivityManager cm =
    (ConnectivityManager) getSystemService(Context.CONNECTIVITY_SERVICE);

    NetworkInfo activeNetwork = cm.getActiveNetworkInfo();
    // перевірка
    if (activeNetwork != null && activeNetwork.isConnected()) {
        try {
            // текст доступності зовнішнього ресурсу
            URL url = new URL("http://www.google.com/");
            HttpURLConnection urlc = (HttpURLConnection)url.openConnection();
            urlc.setRequestProperty("User-Agent", "test");
            urlc.setRequestProperty("Connection", "close");
            urlc.setConnectTimeout(1000); // Timeout в секундах
            urlc.connect();
            // статус ресурса OK
            if (urlc.getResponseCode() == 200) {
                return true;
            }
            // інакше перевірка невдала
            return false;

        } catch (IOException e) {
            Log.d("my_tag", "Помилка перевірки підключення до мережі", e);

            return false;
        }
    }

    return false; }

//метод актуалізації облікових даних з сервером
private void getJSON(final String urlWebService) {
    class GetJSON extends AsyncTask<Void, Void, String> {
        @Override
        protected void onPreExecute() {
            super.onPreExecute();
        }

        @Override
        protected void onPostExecute(String s) {
            super.onPostExecute(s);
            Toast.makeText(getApplicationContext(), "Дані успішно завантажено",
Toast.LENGTH_SHORT).show();
            adapter.notifyDataSetChanged();
            Log.d("MM", s);
            try {

                db.delete(DataBaseHelper.TABLE_NAME, null, null);
                loadIntoListView(s);
            } catch (JSONException e) {
                Log.d("MM", e.toString());
                e.printStackTrace();
            }
        }
    }
}

```

```

@Override
protected String doInBackground(Void... voids) {
    try {
        URL url = new URL(urlWebService);
        HttpURLConnection con = (HttpURLConnection) url.openConnection();
        StringBuilder sb = new StringBuilder();
        BufferedReader bufferedReader = new BufferedReader(new
InputStreamReader(con.getInputStream()));
        String json;
        while ((json = bufferedReader.readLine()) != null) {
            sb.append(json + "\n");
        }

        return sb.toString().trim();
    } catch (Exception e) {
        Log.d("MM", e.toString());
        return null;
    }
}

GetJSON getJSON = new GetJSON();
getJSON.execute();
}

//метод завантаження даних до візуального переліку
private void loadIntoListView(String json) throws JSONException {
    JSONArray jsonArray = new JSONArray(json);
    for (int i = 0; i < jsonArray.length(); i++) {
        JSONObject obj = jsonArray.getJSONObject(i);
        String IMEI = obj.getString("IMEI");
        String surname = obj.getString("surname");
        String name = obj.getString("name");
        String patronymic = obj.getString("patronymic");
        String code_employee = obj.getString("code_employee");
        String phone = obj.getString("phone");

        SQLiteDatabase db = helper.getWritableDatabase();
        ContentValues contentValues = new ContentValues();
        contentValues.put(DataBaseHelper.COLUM_1, IMEI);

        contentValues.put(DataBaseHelper.COLUM_2, surname);
        contentValues.put(DataBaseHelper.COLUM_3, name);
        contentValues.put(DataBaseHelper.COLUM_4, patronymic);
        contentValues.put(DataBaseHelper.COLUM_5, code_employee);
        contentValues.put(DataBaseHelper.COLUM_6, phone);

        db.insert(DataBaseHelper.TABLE_NAME, null, contentValues);

    }
    Restart res = new Restart();
    res.bd();
}

```

```

    }

    //перехід до створення нового облікового запису
    public void btAddPeople(View view) {
        Intent intent = new Intent(BDActivity.this, AddActivity.class);
        startActivity(intent);
    }

    //перехід до тех.підтримки
    public void btnSupport(View view) {

        Intent intent = new Intent(BDActivity.this, SupportActivity.class);
        startActivity(intent);
    }

    //робота з локальною бд
    public class Restart {
        public void bd() {

            Cursor cursor = db.query("passport", null, null, null,
                null, null, null);
            if (cursor.getCount()==0) {
                Toast toast = Toast.makeText(BDActivity.this,
                    Toast.LENGTH_LONG);
                toast.show();}
            else{
                arrayList.clear();
                cursor.moveToFirst();
                while (!cursor.isAfterLast()) {

                    Name = cursor.getString(2)+' '+cursor.getString(3)+' '+cursor.getString(4);

                    IMEI = cursor.getString(1)+"\n"+ cursor.getString(6);
                    cursor.moveToNext();
                    map = new HashMap<>();

                    map.put("IMEI",Name);
                    map.put("Name",IMEI);
                    arrayList.add(map);

                }

                cursor.close();}
        }
    }

    class MyTimerTask extends TimerTask {

        @Override
        public void run() {

```

```

        if (checkInternet()) {
            getJSON("https://saas04.alldholding.com/wp-
security/mobile_api/trackers_sis_02.php");
        } else {
            Log.d("my_tag", "Немає підключення");
        }
    }
}
}
}
}

```

RegistrationActivity.java
package myhome.control;

```

/*підключення модулів та бібліотек
android.content.ContentValues;android.content.Context;android.content.Intent;android.content.
SharedPreferences;android.database.Cursor;android.database.sqlite.SQLiteDatabase;android.net.Connecti
vityManager;android.net.NetworkInfo;android.os.AsyncTask;android.os.Bundle;androidx.fragment.app.F
ragmentManager;androidx.appcompat.app.AppCompatActivity;androidx.appcompat.widget.Toolbar;andr
oid.util.Log;android.view.View;android.widget.Toast;java.io.ByteArrayOutputStream;java.io.IOExceptio
n;java.io.InputStream;java.io.OutputStream;java.net.HttpURLConnection;java.net.MalformedURLException;
ion;java.net.URL;java.text.SimpleDateFormat;java.util.Date;java.util.Timer;java.util.TimerTask;
*/

```

```

public class RegistrationActivity extends AppCompatActivity {
    //екземпляри класів для роботи з бд
    private DataBaseHelper helper;
    private SQLiteDatabase db;
    //змінні для проміжних даних користувача
    String imei;
    String date_begin;
    String date_end;
    String id1;
    Double lat_input, long_input, lat_output, long_output;
    Integer mark;

    //запуск задач у певний час
    private Timer mTimer;
    private MyTimerTask mMyTimerTask;
    //глобальні відомості про користувача
    SharedPreferences mSettings;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_registration);
        helper = new DataBaseHelper(this);
        db = helper.getWritableDatabase();

        mSettings = getSharedPreferences(MainActivity.APP_PREFERENCES,
Context.MODE_PRIVATE);

        if (mTimer != null) {

```

```

        mTimer.cancel();
    }

    // re-schedule timer here
    // otherwise, IllegalStateException of
    // "TimerTask is scheduled already"
    // will be thrown
    mTimer = new Timer();
    mMyTimerTask = new MyTimerTask();
    mTimer.scheduleAtFixedRate(mMyTimerTask, 1000, 86400000);

}

    //перехід до фіксації початку робочої зміни
public void up(View v) {
    Intent intent = new Intent(this, TrackersActivity.class);
    startActivity(intent);
}

    //перехід до фіксації завершення робочої зміни
public void down(View v) {
    Intent intent = new Intent(this, Trackers_backActivity.class);
    startActivity(intent);
}

//преглед співробітників на зміні
public void report(View v) {
    Intent intent = new Intent(this, ReportActivity.class);
    startActivity(intent);
}

//формування місячного звіту
public void btnSendAll(View v) {
    Intent intent = new Intent(this, SendAllReportActivity.class);
    startActivity(intent);
}

//ініціація перевірки наявності доступу до мережі через фрагмент
public void btnHelloClick(View v){
    FragmentManager fm = getSupportFragmentManager();
    myDialogFragment dialog = new myDialogFragment();
    dialog.show(fm, "my_tag"); }

public void okClicked() {
// запуск перевірки у новому потоці
    new Thread(new Runnable() {
        @Override
        public void run(){
            if (checkInternet()) {

                try {
                    new SendData().execute();
                } catch (Exception e) {
                }
            } else {

                try {
                    onNoExecute();

```

```

        } catch (Exception e) {
        }
    }
}
}).start();
}
// головний метод для перевірки підключення
public boolean checkInternet() {

    ConnectivityManager cm
    (ConnectivityManager) getSystemService(Context.CONNECTIVITY_SERVICE);

    NetworkInfo activeNetwork = cm.getActiveNetworkInfo();
    if (activeNetwork != null && activeNetwork.isConnected()) {
        try {
            URL url = new URL("http://www.google.com/");
            HttpURLConnection urlc = (HttpURLConnection)url.openConnection();
            urlc.setRequestProperty("User-Agent", "test");
            urlc.setRequestProperty("Connection", "close");
            urlc.setConnectTimeout(1000); // Timeout в секундах
            urlc.connect();
            // статус OK
            if (urlc.getResponseCode() == 200) {
                return true;
            }
            return false;

        } catch (IOException e) {
            Log.d("my_tag", "Помилка перевірки підключення до мережі", e);

            return false;
        }
    }

    return false;
}

public void cancelClicked() {

}

//до тех.підтримки
public void btnSupport(View view) {

    Intent intent = new Intent(RegistrationActivity.this, SupportActivity.class);
    startActivity(intent);
}

//щоденний звіт
class SendData extends AsyncTask<Void, Void, Void> {

    String resultString = null;

    @Override

```

```

protected void onPreExecute() {
    super.onPreExecute();
}

@Override
protected void doInBackground(Void... params) {
    try {
        String[] selectionArgs = null;
        SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd");
        String strDate = sdf.format(new Date()); //now
        //-----
        Date mydate = new Date(System.currentTimeMillis() - (1000 * 60 * 60 * 24));
        SimpleDateFormat dateFormat = new SimpleDateFormat("yyyy-MM-dd");
        String yestr = dateFormat.format(mydate);
        //-----
        selectionArgs = new String[] { "%" + strDate + "%", "%" + strDate + "%", "%" + yestr + "%",
        "%" + yestr + "%" };
        Log.d("MyLog", "now=" + strDate + "yesterd=" + yestr);
        Cursor cursor = db.query("otchet", null, " patronumic Like ? OR code_employee
        Like ? OR patronumic Like ? OR code_employee Like ?", selectionArgs, null,
        null, null, null);
        cursor.moveToFirst();

        while (!cursor.isAfterLast()) {
            id1 = cursor.getString(0);
            imei = cursor.getString(1);
            date_begin = cursor.getString(4);
            date_end = cursor.getString(5);
            lat_input = cursor.getDouble(8);
            long_input = cursor.getDouble(9);
            lat_output = cursor.getDouble(10);
            long_output = cursor.getDouble(11);
            mark = cursor.getInt(12);

            String myURL = "https://saas04.alldholding.com/wp-
            security/mobile_api/input_trackers_mark_02.php";
            String parammetrs = "imei=" + imei
            + "&date_begin=" + date_begin
            + "&date_end=" + date_end
            + "&lat_input=" + lat_input
            + "&long_input=" + long_input
            + "&lat_output=" + lat_output
            + "&long_output=" + long_output
            + "&mark=" + mark;
            Log.d("MyLog", "parammetrs=" + parammetrs);
            Log.d("MyLog", "myURL=" + myURL + "*");

            byte[] data = null;
            InputStream is = null;

            try {
                URL url = new URL(myURL);
                HttpURLConnection conn = (HttpURLConnection) url.openConnection();
                conn.setReadTimeout(10000);
                conn.setConnectTimeout(15000);
            }
        }
    }
}

```

```

conn.setRequestMethod("POST");
conn.setRequestProperty("Connection", "Keep-Alive");
conn.setRequestProperty("Content-Type", "application/x-www-form-
urlencoded");
conn.setRequestProperty("Content-Length", "" +
Integer.toString(parammetrs.getBytes().length));
conn.setDoOutput(true);
conn.setDoInput(true);

data = parammetrs.getBytes("UTF-8");
OutputStream os = conn.getOutputStream();
os.write(data);
os.flush();
os.close();
data = null;
conn.connect();
int responseCode= conn.getResponseCode();

ByteArrayOutputStream baos = new ByteArrayOutputStream();
if (responseCode == 200) {
    Log.d("MyLog", "hereOK");

    Toast.makeText(getApplicationContext(), "Дані успішно передано",
Toast.LENGTH_SHORT).show();
    is = conn.getInputStream();
    byte[] buffer = new byte[8192];

    SQLiteDatabase db = helper.getWritableDatabase();
    ContentValues contentValues = new ContentValues();
    contentValues.put(DataBaseHelper.COLUMN_6, "1");
    db.update(DataBaseHelper.TABLE_NAME_SAVE, contentValues,null, null);
    int bytesRead;

    while ((bytesRead = is.read(buffer)) != -1) {
        baos.write(buffer, 0, bytesRead);
    }

    data = baos.toByteArray();
    resultString = new String(data, "UTF-8");

} else {

}
conn.disconnect();

} catch (MalformedURLException e) {

} catch (IOException e) {

} catch (Exception e) {

}

```

```

        cursor.moveToNext();
    } cursor.close();
}
catch (Exception e) {
    e.printStackTrace();
}
return null;
}

@Override
protected void onPostExecute(Void result) {
    super.onPostExecute(result);
    Toast.makeText(RegistrationActivity.this, "Звіт успішно передано",
Toast.LENGTH_SHORT).show();

}
}

public void onNoExecute() {

    FragmentManager fm = getSupportFragmentManager();
    myNoInternetData dialog = new myNoInternetData();
    dialog.show(fm, "my_tag");

}

class MyTimerTask extends TimerTask {

    @Override
    public void run() {

        if (checkInternet()) {

            try {
                new SendData().execute();
            } catch (Exception e) {
            }
        } else {

            try {
                onNoExecute();
            } catch (Exception e) {
            }

        }

    }
}
}

```

```
}
```

```
ReportActivity.java
package myhome.control;

/*підключення модулів та бібліотек
  android.content.Intent;android.database.Cursor;android.database.sqlite.SQLiteDatabase;android
  .os.Bundle;androidx.appcompat.app.AppCompatActivity;android.view.View;android.widget.Adapter Vie
  w;android.widget.ImageView;android.widget.ListView;android.widget.SimpleAdapter;android.widget.Te
  xtView;android.widget.Toast;java.text.SimpleDateFormat;java.util.ArrayList;java.util.Date;java.util.Hash
  Map;
  */

public class ReportActivity extends AppCompatActivity {
//елемент інтерфейсу-перелік
  ListView listView;
  TextView bd;
  //екземпляри класів для роботи з бд
  private DataBaseHelper helper;
  private SQLiteDatabase db;
  //змінні для прожних даних користувача
  String Name = "";
  String IMEI = "";
  ArrayList<HashMap<String, String>> arrayList = new ArrayList<>();
  HashMap<String, String> map;
  //адаптер переліку
  SimpleAdapter adapter;

  @Override
  protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_report);

    ImageView exit= (ImageView) findViewById(R.id.button);
    exit.setOnClickListener(new View.OnClickListener() {
      @Override
      public void onClick(View v) {
        Intent intent = new Intent(ReportActivity.this, MainActivity.class);
        startActivity(intent);
      }
    });

    helper = new DataBaseHelper(this);
    db = helper.getWritableDatabase();

    listView = (ListView) findViewById(R.id.listView);
    Restart res = new Restart();
    res.bd();
    adapter = new SimpleAdapter(this, arrayList, android.R.layout.simple_list_item_2,
      new String[]{ "IMEI","Name"},
      new int[]{android.R.id.text1, android.R.id.text2});
    listView.setAdapter(adapter);
    listView.setOnItemClickListener(new AdapterView.OnItemClickListener{
```



```

tvInfo = (TextView) findViewById(R.id.tvInfo);

permission_adding = false;

etTabNumber.addTextChangedListener(new TextWatcher() {
    @Override
    public void beforeTextChanged(CharSequence s, int start, int count, int after) {

    }

    @Override
    public void onTextChanged(CharSequence s, int start, int before, int count) {
        intance="";
    }

    @Override
    public void afterTextChanged(Editable s) {

    }
});

etTabNumber.setOnFocusChangeListener(new View.OnFocusChangeListener() {
    @Override
    public void onFocusChange(View v, boolean hasFocus) {
        if (v == etTabNumber && hasFocus == false) {
            String result = etTabNumber.getText().toString().trim().replaceAll("[^0-9]", "");
            etTabNumber.setText(result);
        }
    }
});

etSurname.addTextChangedListener(new TextWatcher() {
    @Override
    public void beforeTextChanged(CharSequence s, int start, int count, int after) {

    }

    @Override
    public void onTextChanged(CharSequence s, int start, int before, int count) {
        intance="";
    }

    @Override
    public void afterTextChanged(Editable s) {

    }
});

etName.addTextChangedListener(new TextWatcher() {
    @Override
    public void beforeTextChanged(CharSequence s, int start, int count, int after) {

    }

    @Override
    public void onTextChanged(CharSequence s, int start, int before, int count) {
        intance="";
    }
}

```

```

        @Override
        public void afterTextChanged(Editable s) {

        }
    });
    etPatronymic.addTextChangedListener(new TextWatcher() {
        @Override
        public void beforeTextChanged(CharSequence s, int start, int count, int after) {

        }

        @Override
        public void onTextChanged(CharSequence s, int start, int before, int count) {
            intance="";
        }

        @Override
        public void afterTextChanged(Editable s) {

        }
    });

    tvInfo.setText("");
    nameCompany="";
    namePosition="";

    helper = new DataBaseHelper(this);
    db = helper.getWritableDatabase();

    task();

    Iterator<Company> i1 = arrayList.iterator();
    while (i1.hasNext())
    {
        Company s = i1.next();
        if (s == null )
        {
            i1.remove();
        }
    }
}

adapterCompany = new ArrayAdapter<Company>(this.context,
android.R.layout.simple_spinner_item, arrayList);

adapterCompany.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item);

spCompany.setAdapter(adapterCompany);
spCompany.setOnItemClickListener(new AdapterView.OnItemClickListener() {
    @Override
    public void onItemClick(AdapterView<?> parent, View view,
        int position, long id) {
        // позиція обраного елементу
        Log.d("MyLog", "AddActivity @ name() " + arrayList.get(position).name_company);
        nameCompany = arrayList.get(position).name_company;
    }
}

```

```

        @Override
        public void onNothingSelected(AdapterView<?> arg0) {
        }
    });
    spCompany.setSelection(0);

    task_position();

    Iterator<Position> i2 = arrayListPosition.iterator();
    while (i2.hasNext())
    {
        Position s = i2.next();
        if (s == null )
        {
            i2.remove();
        }
    }
}

    adapterPosition = new ArrayAdapter<Position>(this.context,
    android.R.layout.simple_spinner_item, arrayListPosition);

    adapterPosition.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item);

    spPosition.setAdapter(adapterPosition);
    spPosition.setOnItemClickListener(new AdapterView.OnItemClickListener() {
        @Override
        public void onItemClick(AdapterView<?> parent, View view,
            int position, long id) {
            // позиція обраного елементу
            Log.d("MyLog", "AddActivity @ " +
            arrayListPosition.get(position).name_position);
            namePosition = arrayListPosition.get(position).name_position;
        }
        @Override
        public void onNothingSelected(AdapterView<?> arg0) {
        }
    });
    spPosition.setSelection(0);

}

//запуск відправки нового користувача
public void okClicked() {
// запуск у окремому потоці
    new Thread(new Runnable() {
        @Override
        public void run(){
            if (checkInternet()) {

                try {
                    new SendData().execute();

```

```

        } catch (Exception e) {
        }
    } else {

        try {
            onNoExecute();
        } catch (Exception e) {
        }
    }
}
}).start();
}

public void onNoExecute() {

    FragmentManager fm = getSupportFragmentManager();
    myNoInternetDataAdd dialog = new myNoInternetDataAdd();
    dialog.show(fm, "my_tag");

}

//запуск отримання підрозділів
public void task(){
    new Thread(new Runnable() {
        @Override
        public void run(){
            if (checkInternet()) {
                arrayList.clear();
                getJSON("https://saas04.alldholding.com/wp-
security/mobile_api/trackers_sis_company1.php");
            } else {
                arrayList.clear();
                Log.d("my_tag", "Немає підключення");
            }
        }
    }).start();
}

//запуск отримання посад
public void task_position(){
    new Thread(new Runnable() {
        @Override
        public void run(){
            if (checkInternet()) {
                arrayListPosition.clear();
                getJSONPosition("https://saas04.alldholding.com/wp-
security/mobile_api/trackers_sis_position.php");
            } else {
                arrayListPosition.clear();
                Log.d("my_tag", "Немає підключення");
            }
        }
    }).start();
}

```

```

    }

    // головний метод для перевірки підключення до мережі
    public boolean checkInternet() {

        ConnectivityManager cm
        (ConnectivityManager) getSystemService(Context.CONNECTIVITY_SERVICE);

        NetworkInfo activeNetwork = cm.getActiveNetworkInfo();

        if (activeNetwork != null && activeNetwork.isConnected()) {
            try {
                URL url = new URL("http://www.google.com/");
                HttpURLConnection urlc = (HttpURLConnection)url.openConnection();
                urlc.setRequestProperty("User-Agent", "test");
                urlc.setRequestProperty("Connection", "close");
                urlc.setConnectTimeout(1000); // Timeout в секундах
                urlc.connect();
                // статус OK
                if (urlc.getResponseCode() == 200) {
                    return true;
                }

                return false;

            } catch (IOException e) {
                Log.d("my_tag", "Помилка перевірки підключення до мережі", e);

                return false;
            }
        }

        return false; }

    //завантаження актуальних даних підрозділів
    private void getJSON(final String urlWebService) {
        class GetJSON extends AsyncTask<Void, Void, String> {
            @Override
            protected void onPreExecute() {
                super.onPreExecute();
            }

            @Override
            protected void onPostExecute(String s) {
                super.onPostExecute(s);
                Toast.makeText(getApplicationContext(), "Дані успішно завантажено",
                Toast.LENGTH_SHORT).show();
                try {

                    db.delete(DataBaseHelper.TABLE_COMPANY, null, null);
                    loadIntoLocalDB(s);
                } catch (JSONException e) {
                    e.printStackTrace();
                }
            }
        }
    }

```

```

    }

    @Override
    protected String doInBackground(Void... voids) {
        try {
            URL url = new URL(urlWebService);
            HttpURLConnection con = (HttpURLConnection) url.openConnection();
            StringBuilder sb = new StringBuilder();
            BufferedReader bufferedReader = new BufferedReader(new
InputStreamReader(con.getInputStream()));
            String json;
            while ((json = bufferedReader.readLine()) != null) {
                sb.append(json + "\n");
            }
            return sb.toString().trim();
        } catch (Exception e) {
            return null;
        }
    }
}
GetJSON getJSON = new GetJSON();
getJSON.execute();
}

//завантаження актуальних даних посад
private void getJSONPosition(final String urlWebService) {
    class GetJSONPosition extends AsyncTask<Void, Void, String> {
        @Override
        protected void onPreExecute() {
            super.onPreExecute();
        }

        @Override
        protected void onPostExecute(String s) {
            super.onPostExecute(s);
            Toast.makeText(getApplicationContext(), "Дані успішно завантажено",
Toast.LENGTH_SHORT).show();
            try {

                db.delete(DataBaseHelper.TABLE_POSITION, null, null);
                loadPosIntoLocalDB(s);
            } catch (JSONException e) {
                e.printStackTrace();
            }
        }
    }
}

    @Override
    protected String doInBackground(Void... voids) {
        try {
            URL url = new URL(urlWebService);
            HttpURLConnection con = (HttpURLConnection) url.openConnection();
            StringBuilder sb = new StringBuilder();

```

```

        BufferedReader bufferedReader = new BufferedReader(new
InputStreamReader(con.getInputStream()));
        String json;
        while ((json = bufferedReader.readLine()) != null) {
            sb.append(json + "\n");
        }
        return sb.toString().trim();
    } catch (Exception e) {
        return null;
    }
}
}
GetJSONPosition getJSONPosition = new GetJSONPosition();
getJSONPosition.execute();
}

```

//робота з локальною бд з підрозділами

```

private void loadIntoLocalDB(String json) throws JSONException {
    JSONArray jsonArray = new JSONArray(json);
    DataBaseHelper dataBaseHelper = new DataBaseHelper(AddActivity.this);
    //dataBaseHelper.
    for (int i = 0; i < jsonArray.length(); i++) {
        JSONObject obj = jsonArray.getJSONObject(i);
        String id = obj.getString("id");
        String name = obj.getString("name");
        String code = obj.getString("code");

        SQLiteDatabase db = helper.getWritableDatabase();
        ContentValues contentValues = new ContentValues();
        contentValues.put(DataBaseHelper.COLUM_ID, id);
        contentValues.put(DataBaseHelper.COLUM_3, name);
        contentValues.put(DataBaseHelper.COLUM_7, code);

        db.insert(DataBaseHelper.TABLE_COMPANY, null, contentValues);
    }
}

```

```

Restart res = new Restart();
res.bd();

```

```

}

```

//робота з локальною бд з посадами

```

private void loadPosIntoLocalDB(String json) throws JSONException {
    JSONArray jsonArray = new JSONArray(json);
    DataBaseHelper dataBaseHelper = new DataBaseHelper(AddActivity.this);
    //dataBaseHelper.
    for (int i = 0; i < jsonArray.length(); i++) {
        JSONObject obj = jsonArray.getJSONObject(i);
        String id = obj.getString("id");
        String name_position = obj.getString("name_position");
        String code_position = obj.getString("code_position");

        SQLiteDatabase db = helper.getWritableDatabase();
        ContentValues contentValues = new ContentValues();
    }
}

```

```

        contentValues.put(DataBaseHelper.COLUM_ID, id);
        contentValues.put(DataBaseHelper.COLUM_8, name_position);
        contentValues.put(DataBaseHelper.COLUM_9, code_position);

        db.insert(DataBaseHelper.TABLE_POSITION, null, contentValues);

    }

    RestartPos restartPos = new RestartPos();
    restartPos.bd();

}

public void IntertnetNoOkClicked() {

}

    //натискання на додавання облік.запису
    public void addPeople(View view) {
        String string = validate();
        if(string.equals("")){
            Log.d("kop", "intance = "+intance);
            if(intance.equals("")){
                Toast toast = Toast.makeText(AddActivity.this,
                    "Перевірте дані перед додаванням",
                    Toast.LENGTH_SHORT);
                toast.show();
                tvInfo.setText("");
            }else {
                if (intance.contains("1")) {
                    Toast toast = Toast.makeText(AddActivity.this,
                        "Такий користувач вже існує",
                        Toast.LENGTH_SHORT);
                    toast.show();
                } else {
                    Company company = new Company();
                    int position = spCompany.getSelectedItemPosition();
                    company = arrayList.get(position);
                    userAdd.code_company = company.code;
                    Position position1 = new Position();
                    int pos = spPosition.getSelectedItemPosition();
                    position1 = arrayListPosition.get(pos);
                    userAdd.code_position = position1.code_position;

                    userAdd.imei = etTabNumber.getText().toString().trim();

                    userAdd.surname = etSurname.getText().toString().trim();
                    if (!etName.getText().toString().trim().equals("")) {
                        userAdd.name_people = etName.getText().toString().trim();
                    } else {
                        userAdd.name_people = "";
                    }
                    if (!etPatronymic.getText().toString().trim().equals("")) {
                        userAdd.patronymic = etPatronymic.getText().toString().trim();
                    } else {
                        userAdd.patronymic = "";
                    }
                }
            }
        }
    }

```

```

        }
        onClicked();
    }
}
}else{
    Toast toast = Toast.makeText(AddActivity.this,
        string,
        Toast.LENGTH_SHORT);
    toast.show();
}
intance="";
}

```

//валідація даних

```

public String validate(){
    if(etTabNumber.getText().toString().trim().equals("")){
        return "Зчитайте номер перепуски";
    }
    if(etSurname.getText().toString().trim().equals("")){
        return "Введіть прізвище";
    }
    Company company = new Company();
    int position = spCompany.getSelectedItemPosition();
    if(arrayList.size() != 0) {
        company = arrayList.get(position);
        Log.d("MyLog", "name - " + company.name_company + "code - " + company.code);
    }else{
        return "Підрозділи не завантажено";
    }
    return "";
}

```

//тех.підтримка

```

public void Support(View view) {
    Intent intent = new Intent(AddActivity.this, SupportActivity.class);
    startActivity(intent);
}

```

//вихід з активіті

```

public void home(View view) {
    finish();
}

```

//робота з бд-всі користувачі локальної бд

```

public ArrayList<CheckUserFromLocalDB> getAllPeopleLocalDB(){

    Cursor cursor = null;
    ArrayList<CheckUserFromLocalDB> array = new ArrayList<>();
    try {

        String sql = "SELECT * FROM passport";
        cursor = db.rawQuery(sql, new String[]{
        });
        cursor.moveToFirst();

        while (!cursor.isAfterLast()) {

```

```

        array.add(this.createPeopleFromDB(cursor));
        cursor.moveToNext();
    }

    }catch (Exception e){
        Log.d("kop", "AddActivity @ getAllPeopleLocalDB() " + e.getMessage());
    }finally {
        if (cursor != null) {
            cursor.close();
        }
    }
    return array;
}

//створення структури одиниці юзера
private CheckUserFromLocalDB createPeopleFromDB(Cursor cursor){
    CheckUserFromLocalDB people = new CheckUserFromLocalDB();
    people.surname = cursor.getString(cursor.getColumnIndex("surname"));
    people.name_people = cursor.getString(cursor.getColumnIndex("name"));
    people.patronymic = cursor.getString(cursor.getColumnIndex("patronymic"));
    people.imei = cursor.getString(cursor.getColumnIndex("imei"));
    return people;
}

//перевірка даних
public void checkPeople(View view) {

    String text = etTabNumber.getText().toString().trim();
    if (text.equals(adding_password)) {
        permission_adding = true;
    }
    if (permission_adding) {
        intance = "2";
        buffer = "Перевірка успішна";
        tvInfo.setText(buffer);
    }else{
        String string = validate();
        if (string.equals("")) {
            tvInfo.setText("");
            buffer = "";
            intance = "";
        }

        ArrayList<CheckUserFromLocalDB> array = getAllPeopleLocalDB();
        String surname = etSurname.getText().toString().trim();
        String name = etName.getText().toString().trim();
        String patronymic = etPatronymic.getText().toString().trim();
        String imei = etTabNumber.getText().toString().trim();
        for (int i = 0; i < array.size(); i++) {
            CheckUserFromLocalDB people = array.get(i);
            if (people.imei.equals(imei)) {
                buffer += "\nКористувач з такою перепусткою вже існує: " + people.print();
                intance += "1";
            } else {
                if (people.surname.equals(surname)) {
                    if (people.name_people.equals(name)) {
                        if (people.patronymic.equals(patronymic)) {

```

```

        buffer += "\nТакий користувач вже існує: " + people.print();

        intance += "1";

    } else {
        buffer += "\nСпівпадання за прізвищем та ім'ям: " + people.print();

        intance += "2";
    }
} else {
    buffer += "\nСпівпадання за прізвищем: " + people.print();

    intance += "3";
}
} else {
    if (people.surname.contains(surname)) {
        buffer += "\nМожливе співпадання за прізвищем: " + people.print();

        intance += "4";
    } else {
        intance += "0";
    }
}
}
}
if (buffer.equals("")) {
    buffer = "Перевірка успішна";
}
tvInfo.setText(buffer);
} else {
    Toast toast = Toast.makeText(AddActivity.this,
        string,
        Toast.LENGTH_SHORT);
    toast.show();
    tvInfo.setText("");
}

}

}

//робота з бд - підрозділи
public class Restart {
    public void bd() {

        Cursor cursor = db.query("company", null, null, null,
            null, null, null);
        if (cursor.getCount()==0) {
            arrayList.clear();
            Toast toast = Toast.makeText(AddActivity.this,
                "Дані по підрозділам не завантажено",
                Toast.LENGTH_SHORT);
            toast.show();
        }
        else{
            arrayList.clear();

```

```

        cursor.moveToFirst();
        while (!cursor.isAfterLast()) {

            Company company = new Company();
            company.id = cursor.getLong(0);
            company.name_company = cursor.getString(1);
            company.code = cursor.getString(2);

            cursor.moveToNext();

            arrayList.add(company);

        }

        for(int i=0; i<arrayList.size();i++){
            Log.d("MyLog", "arrayList - "+ arrayList.get(i).name_company);
        }
        adapterCompany.notifyDataSetChanged();
        spCompany.setSelection(5);
        cursor.close();
    }
}

//робота з бд - посади
public class RestartPos {
    public void bd() {

        Cursor cursor = db.query("position", null, null, null,
            null, null, null);
        if (cursor.getCount()==0) {
            arrayListPosition.clear();
            Toast toast = Toast.makeText(AddActivity.this,
                "Дані по посадам не завантажено",
                Toast.LENGTH_SHORT);
            toast.show();
        }
        else{
            arrayListPosition.clear();
            cursor.moveToFirst();
            while (!cursor.isAfterLast()) {

                Position position = new Position();
                position.id = cursor.getLong(0);
                position.name_position = cursor.getString(1);
                position.code_position = cursor.getString(2);

                cursor.moveToNext();

                arrayListPosition.add(position);

            }

            for(int i=0; i<arrayListPosition.size();i++){

```

```

        Log.d("MyLog", "arrayListPosition - "+ arrayListPosition.get(i).name_position);
    }
    adapterPosition.notifyDataSetChanged();
    spPosition.setSelection(0);
    cursor.close();
}
}
}

```

```

//вiдправка даних облiкового запису на сервер
class SendData extends AsyncTask<Void, Void, Void> {

```

```

    String resultString = null;

```

```

    boolean flag = false;

```

```

    @Override
    protected void onPreExecute() {
        super.onPreExecute();
    }

```

```

    @Override
    protected Void doInBackground(Void... params) {
        try {

```

```

            String myURL = "https://saas04.alldholding.com/wp-
security/mobile_api/input_trackers_responsible_rfid1.php";
            String parammetrs ="surname="+userAdd.surname
                +"&name="+userAdd.name_people
                +"&patronymic="+userAdd.patronymic
                +"&code="+userAdd.code_company
                +"&position="+userAdd.code_position
                +"&imei="+userAdd.imei;

```

```

            Log.d("MyLog", "parammetrs="+parammetrs);

```

```

            byte[] data = null;
            InputStream is = null;

```

```

            try {
                URL url = new URL(myURL);
                HttpURLConnection conn = (HttpURLConnection) url.openConnection();
                conn.setReadTimeout(10000);
                conn.setConnectTimeout(15000);
                conn.setRequestMethod("POST");
                conn.setRequestProperty("Connection", "Keep-Alive");
                conn.setRequestProperty("Content-Type", "application/x-www-form-
urlencoded");
                conn.setRequestProperty("Content-Length", ""
                    + Integer.toString(parammetrs.getBytes().length));
                conn.setDoOutput(true);

```

```

        conn.setDoInput(true);

        data = parammetrs.getBytes("UTF-8");
        OutputStream os = conn.getOutputStream();
        os.write(data);
        os.flush();
        os.close();
        data = null;
        conn.connect();
        int responseCode= conn.getResponseCode();

        ByteArrayOutputStream baos = new ByteArrayOutputStream();
        if (responseCode == 200) {
            Log.d("MyLog", "hereOK");
            flag = true;

            is = conn.getInputStream();
            byte[] buffer = new byte[8192];

            int bytesRead;

            while ((bytesRead = is.read(buffer)) != -1) {
                baos.write(buffer, 0, bytesRead);
            }

            data = baos.toByteArray();
            resultString = new String(data, "UTF-8");
        } else {

        }
        conn.disconnect();

    } catch (MalformedURLException e) {

    } catch (IOException e) {

    } catch (Exception e) {

    }

    }
    catch (Exception e) {
        e.printStackTrace();
    }
    return null;
}

@Override
protected void onPostExecute(Void result) {
    super.onPostExecute(result);
    if(flag){
        etTabNumber.setText("");
        etName.setText("");
        etSurname.setText("");
    }
}

```

```

        etPatronymic.setText("");
        tvInfo.setText("");
        Toast.makeText(getApplicationContext(), "Дані успішно завантажено",
Toast.LENGTH_SHORT).show();
    }

    }
}

```

SendAllReportActivity.java
package myhome.control;

```

/*підключення модулів та бібліотек
androidx.appcompat.app.AppCompatActivity;androidx.fragment.app.FragmentManager;androi
d.app.DatePickerDialog;android.app.TimePickerDialog;android.content.ContentValues;android.content.C
ontext;android.content.Intent;android.database.Cursor;android.database.sqlite.SQLiteDatabase;android.ne
t.ConnectivityManager;android.net.NetworkInfo;android.os.AsyncTask;android.os.Bundle;android.text.fo
rmat.DateFormat;android.text.format.DateUtils;android.util.Log;android.view.View;android.widget.Date
Picker;android.widget.TextView;android.widget.Toast;java.io.ByteArrayOutputStream;java.io.IOExcepti
on;java.io.InputStream;java.io.OutputStream;java.net.HttpURLConnection;java.net.MalformedURLExceptionExce
ption;java.net.URL;java.text.SimpleDateFormat;java.util.Calendar;java.util.Date;
*/

```

```
public class SendAllReportActivity extends AppCompatActivity {
```

```

//екземпляри класів для роботи з бд
private DataBaseHelper helper;
private SQLiteDatabase db;
//змінні для проміжних даних користувача
String imei;
String date_begin;
String date_end;
String id1;
Double lat_input, long_input, lat_output, long_output;
Integer mark;

```

```

//елементи інтерфейсу
TextView currentDate1;
Calendar date1=Calendar.getInstance();

```

```

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_send_all_report);

```

```

    helper = new DataBaseHelper(this);
    db = helper.getWritableDatabase();

```

```

    currentDate1 = findViewById(R.id.currentDate1);

```

```

    setInitialDate1();

```

```

    }

    // відображення вікна вибору дати
    public void setDate1(View v) {
        new DatePickerDialog(SendAllReportActivity.this, d1,
            date1.get(Calendar.YEAR),
            date1.get(Calendar.MONTH),
            date1.get(Calendar.DAY_OF_MONTH))
            .show();
    }

    // встановлення початкових дат
    private void setInitialDate1() {
        currentDate1.setText(DateUtils.formatDateTime(this,
            date1.getTimeInMillis(),
            DateUtils.FORMAT_SHOW_DATE | DateUtils.FORMAT_SHOW_YEAR));
    }

    // встановлення обробника вибору дати
    DatePickerDialog.OnDateSetListener d1=new DatePickerDialog.OnDateSetListener() {
        @Override
        public void onDateSet(DatePicker view, int year, int monthOfYear, int dayOfMonth) {
            date1.set(Calendar.YEAR, year);
            date1.set(Calendar.MONTH, monthOfYear);
            date1.set(Calendar.DAY_OF_MONTH, dayOfMonth);
            setInitialDate1();
        }
    };

    //тех.підтримка
    public void btnSupport(View view) {
        Intent intent = new Intent(SendAllReportActivity.this, SupportActivity.class);
        startActivity(intent);
    }

    //попередження користувачу
    public void sendAll(View view){
        FragmentManager fm = getSupportFragmentManager();
        myDialogFragment1 dialog = new myDialogFragment1();
        dialog.show(fm, "my_tag");
    }

    public void okClicked() {
        // запуск перевірки на наявність мережі в окремому потоці
        new Thread(new Runnable() {
            @Override
            public void run(){
                if (checkInternet()) {

                    Log.d("MyLog", "here 1");
                    try {
                        new SendData1().execute();
                    } catch (Exception e) {
                        Log.d("MyLog", "Exception"+e);
                    }
                }
            }
        }).start();
    }

```

```

    }
    } else {

        try {
            onNoExecute();
        } catch (Exception e) {
        }
    }
}
}).start();
}

```

```

public void onNoExecute() {

```

```

    FragmentManager fm = getSupportFragmentManager();
    myNoInternetData1 dialog = new myNoInternetData1();
    dialog.show(fm, "my_tag");

```

```

}

```

```

// головний метод для перевірки підключення до мережі
public boolean checkInternet() {

```

```

    ConnectivityManager cm
    (ConnectivityManager) getSystemService(Context.CONNECTIVITY_SERVICE);

```

```

=

```

```

    NetworkInfo activeNetwork = cm.getActiveNetworkInfo();
    if (activeNetwork != null && activeNetwork.isConnected()) {
        try {
            URL url = new URL("http://www.google.com/");
            HttpURLConnection urlc = (HttpURLConnection)url.openConnection();
            urlc.setRequestProperty("User-Agent", "test");
            urlc.setRequestProperty("Connection", "close");
            urlc.setConnectTimeout(1000); // Timeout в секундах
            urlc.connect();
            // статус OK
            if (urlc.getResponseCode() == 200) {
                return true;
            }
            return false;

```

```

        } catch (IOException e) {
            Log.d("my_tag", "Помилка перевірки доступу до мережі", e);

```

```

            return false;
        }
    }
}

```

```

return false;
}

```

```

//передача даних звіту на сервер

```

```

class SendData1 extends AsyncTask<Void, Void, Void> {

```

```

String resultString = null;

@Override
protected void onPreExecute() {
    super.onPreExecute();
}

@Override
protected Void doInBackground(Void... params) {
    try {
        String[] selectionArgs = null;
        String strDate = DateFormat.format("yyyy-MM", date1).toString();//now
        Log.d("MyLog", "1strDate"+strDate);
        selectionArgs = new String[] { "%"+strDate+"%", "%"+strDate+"%"};
        Log.d("MyLog", "1now="+strDate);
        Cursor cursor = db.query("otchet", null, " patronumic Like ? OR code_employee
Like ?", selectionArgs, null,
            null, null, null);
        cursor.moveToFirst();

        while (!cursor.isAfterLast()) {
            id1 = cursor.getString(0);
            imei = cursor.getString(1);
            date_begin = cursor.getString(4);
            date_end = cursor.getString(5);
            lat_input = cursor.getDouble(8);
            long_input = cursor.getDouble(9);
            lat_output = cursor.getDouble(10);
            long_output = cursor.getDouble(11);
            mark = cursor.getInt(12);

            String myURL = "https://saas04.alldholding.com/wp-
security/mobile_api/input_trackers_mark_send_zp_02.php";
            String parammetrs ="imei="+imei
            +"&date_begin="+date_begin
            +"&date_end="+date_end
            +"&lat_input="+lat_input
            +"&long_input="+long_input
            +"&lat_output="+lat_output
            +"&long_output="+long_output
            +"&mark="+mark;
            Log.d("MyLog", "1parammetrs="+parammetrs);

            byte[] data = null;
            InputStream is = null;

            try {
                URL url = new URL(myURL);
                HttpURLConnection conn = (HttpURLConnection) url.openConnection();
                conn.setReadTimeout(10000);
                conn.setConnectTimeout(15000);
                conn.setRequestMethod("POST");
                conn.setRequestProperty("Connection", "Keep-Alive");
            }
        }
    }
}

```

```

conn.setRequestProperty("Content-Type","application/x-www-form-
urlencoded");
conn.setRequestProperty("Content-Length", "" +
Integer.toString(parammets.getBytes().length));
conn.setDoOutput(true);
conn.setDoInput(true);

data = parammets.getBytes("UTF-8");
OutputStream os = conn.getOutputStream();
os.write(data);
os.flush();
os.close();
data = null;
conn.connect();
int responseCode= conn.getResponseCode();

ByteArrayOutputStream baos = new ByteArrayOutputStream();
if (responseCode == 200) {
    Log.d("MyLog", "hereOK");

    Toast.makeText(getApplicationContext(), "Дані успішно надіслані",
Toast.LENGTH_SHORT).show();
    is = conn.getInputStream();
    byte[] buffer = new byte[8192];

    SQLiteDatabase db = helper.getWritableDatabase();
    ContentValues contentValues = new ContentValues();
    contentValues.put(DataBaseHelper.COLUM_6, "1");
    db.update(DataBaseHelper.TABLE_NAME_SAVE, contentValues,null, null);
    int bytesRead;

    while ((bytesRead = is.read(buffer)) != -1) {
        baos.write(buffer, 0, bytesRead);
    }

    data = baos.toByteArray();
    resultString = new String(data, "UTF-8");

    } else {

    }
    conn.disconnect();

    } catch (MalformedURLException e) {

    } catch (IOException e) {

    } catch (Exception e) {

    }

    cursor.moveToNext();

```

```

        } cursor.close();
    }
    catch (Exception e) {
        e.printStackTrace();
    }
    return null;
}

@Override
protected void onPostExecute(Void result) {
    super.onPostExecute(result);
    Toast.makeText(SendAllReportActivity.this, "Звіт успішно передано",
Toast.LENGTH_SHORT).show();

}
}
}

```

ActivityPerson.java

package myhome.control;

/*підключення модулів та бібліотек

```

    androidx.appcompat.app.AppCompatActivity;android.app.ProgressDialog;android.content.Intent;android.graphics.Bitmap;android.graphics.BitmapFactory;android.os.AsyncTask;android.os.Bundle;android.provider.ContactsContract;android.util.Log;android.view.View;android.widget.ImageView;android.widget.TextView;android.widget.Toast;com.github.barteksc.pdfviewer.PDFView;com.google.gson.Gson;com.google.gson.GsonBuilder;java.io.BufferedReader;java.io.IOException;java.io.InputStream;java.net.HttpURLConnection;java.net.MalformedURLException;java.net.URL;javax.net.ssl.HttpURLConnection;okhttp3.MediaType;okhttp3.OkHttpClient;okhttp3.RequestBody;okhttp3.logging.HttpLoggingInterceptor;retrofit2.Call;retrofit2.Callback;retrofit2.Response;retrofit2.Retrofit;retrofit2.converter.gson.GsonConverterFactory;
*/

```

```

public class ActivityPerson extends AppCompatActivity {

```

```

    //змінні для проміжних даних користувача

```

```

    String imei="",fio="",surname="",name="",patronymic="";

```

```

    //елементи інтерфейсу

```

```

    TextView tvSurname, tvName, tvPatronymic,etCompany, etPosition, etPhone, etDate;

```

```

    ImageView ivPhoto;

```

```

    private ProgressDialog progressDialogSending;

```

```

    //екземпляри класів для роботи з сервером

```

```

    private Retrofit retrofit;

```

```

    private ServerApi serverApi;

```

```

    //виклики запитів

```

```

    private Call<String> callDocumentURL;

```

```

    private Call<String> callCompany;

```

```

    private Call<String> callPosition;

```

```

    private Call<String> callPhone;

```

```

    private Call<String> callDate;

```

```

    //елемент перегляду файлів

```

```

    PDFView pdfView;

```

```

    //додаткові змінні

```

```

    String company="";

```

```

    String position="";

```

```

    String phone="";

```

```

String date="";

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_person);
    tvSurname=(TextView)findViewById(R.id.etSurname);
    tvName=(TextView)findViewById(R.id.etName);
    tvPatronymic=(TextView)findViewById(R.id.etPatronymic);
    pdfView=findViewById(R.id.ivPhoto);
    etCompany =(TextView)findViewById(R.id.etCompany);
    etPosition=(TextView)findViewById(R.id.etPosition);
    etPhone=(TextView)findViewById(R.id.etPhone);
    etDate=(TextView)findViewById(R.id.etDate);

    imei = getIntent().getStringExtra("imei");
    Log.d("MyLog111","*" + imei + "*");
    fio = getIntent().getStringExtra("fio");
    if(fio!=null){
        Log.d("MyLog111","*" + fio + "*");

        String[] words = fio.split(" ");
        surname = words[0];
        name = words[1];
        patronymic = words[2];

        tvSurname.setText(surname);
        tvName.setText(name);
        tvPatronymic.setText(patronymic);
        Log.d("MyLog111","*" + surname + "*");
        Log.d("MyLog111","*" + name + "*");
        Log.d("MyLog111","*" + patronymic + "*");
    }

    //-----
    Gson gson = new GsonBuilder()
        .setLenient()
        .create();
    HttpLoggingInterceptor logging = new HttpLoggingInterceptor();
    logging.setLevel(HttpLoggingInterceptor.Level.BODY);
    OkHttpClient.Builder httpClient = new OkHttpClient.Builder();
    httpClient.addInterceptor(logging);
    retrofit = new Retrofit.Builder()
        .baseUrl("https://saas04.alldholding.com/wp-security/mobile_api/")
        .addConverterFactory(GsonConverterFactory.create(gson))
        .client(Extras.getClient(ActivityPerson.this))
        .build();
    serverApi = retrofit.create(ServerApi.class);

    getDocumentURL();
    getCompany();
    getPosition();
    getPhone();
    getDate();
}

```

```

//отримання посилання на фото
public void getDocumentURL(){

    new ActivityPerson.RetrievePDFfromUrl().execute(imageUrl);

    RequestBody action_rb = RequestBody.create(MediaType.parse("multipart/form-data"),
"get_photo_path");
    RequestBody imei_rb = RequestBody.create(MediaType.parse("multipart/form-data"),
String.valueOf(imei));
    callDocumentURL = serverApi.getPhotoURL(action_rb,imei_rb);
    callDocumentURL.enqueue(new Callback<String>() {
        @Override
        public void onResponse(Call<String> call, Response<String> response) {
            if (response.isSuccessful()){
                imageUrl = response.body();
                Log.d("MyLog getPhotoURL ", "imageUrl=" + imageUrl);
                new RetrievePDFfromUrl().execute(imageUrl);
            }else{
                Log.d("MyLog getPhotoURL ", "fail = " + response.errorBody());
            }
        }

        @Override
        public void onFailure(Call<String> call, Throwable t) {
            Toast.makeText(ActivityPerson.this, getString(R.string.errorNetwork),
Toast.LENGTH_SHORT).show();
        }
    });

}

//отримання даних підрозділу
public void getCompany(){

    RequestBody action_rb = RequestBody.create(MediaType.parse("multipart/form-data"),
"get_company");
    RequestBody imei_rb = RequestBody.create(MediaType.parse("multipart/form-data"),
String.valueOf(imei));
    callCompany = serverApi.getCompany(action_rb,imei_rb);
    callCompany.enqueue(new Callback<String>() {
        @Override
        public void onResponse(Call<String> call, Response<String> response) {
            if (response.isSuccessful()){
                company = response.body();
                Log.d("MyLog getCompany", "company=" + company);
                etCompany.setText(String.valueOf(company));
            }else{
                Log.d("MyLog getCompany", "fail = " + response.errorBody());
            }
        }

        @Override
        public void onFailure(Call<String> call, Throwable t) {
            Toast.makeText(ActivityPerson.this, getString(R.string.errorNetwork),
Toast.LENGTH_SHORT).show();
        }
    });

}

```

```

    }
    });

}

//отримання посади
public void getPosition(){

    RequestBody action_rb = RequestBody.create(MediaType.parse("multipart/form-data"),
"get_position");
    RequestBody imei_rb = RequestBody.create(MediaType.parse("multipart/form-data"),
String.valueOf(imei));
    callPosition = serverApi.getPosition(action_rb,imei_rb);
    callPosition.enqueue(new Callback<String>() {
        @Override
        public void onResponse(Call<String> call, Response<String> response) {
            if (response.isSuccessful()){
                position = response.body();
                Log.d("MyLog getPosition", "position=" + position);
                etPosition.setText(String.valueOf(position));
            }else{
                Log.d("MyLog getPosition", "fail = " + response.errorBody());
            }
        }

        @Override
        public void onFailure(Call<String> call, Throwable t) {
            Toast.makeText(ActivityResult.this, getString(R.string.errorNetwork),
Toast.LENGTH_SHORT).show();
        }
    });

}

//номер телефону отримуємо
public void getPhone(){

    RequestBody action_rb = RequestBody.create(MediaType.parse("multipart/form-data"),
"get_phone");
    RequestBody imei_rb = RequestBody.create(MediaType.parse("multipart/form-data"),
String.valueOf(imei));
    callPhone = serverApi.getPhone(action_rb,imei_rb);
    callPhone.enqueue(new Callback<String>() {
        @Override
        public void onResponse(Call<String> call, Response<String> response) {
            if (response.isSuccessful()){
                phone = response.body();
                Log.d("MyLog getPhone", "phone =" + phone);
                etPhone.setText(String.valueOf(phone));
            }else{
                Log.d("MyLog getPhone", "fail = " + response.errorBody());
            }
        }

        @Override

```

```

        public void onFailure(Call<String> call, Throwable t) {
            Toast.makeText(ActivityPerson.this, getString(R.string.errorNetwork),
Toast.LENGTH_SHORT).show();
        }
    });

}

    //дата працевлаштування
    public void getDate(){

        RequestBody action_rb = RequestBody.create(MediaType.parse("multipart/form-data"),
"get_employment_date");
        RequestBody imei_rb = RequestBody.create(MediaType.parse("multipart/form-data"),
String.valueOf(imei));
        callDate = serverApi.getDate(action_rb, imei_rb);
        callDate.enqueue(new Callback<String>() {
            @Override
            public void onResponse(Call<String> call, Response<String> response) {
                if (response.isSuccessful()){
                    date = response.body();
                    Log.d("MyLog getDate", "date=" + date);
                    etDate.setText(String.valueOf(date));
                }else{
                    Log.d("MyLog getDate", "fail = " + response.errorBody());
                }
            }
        })

        @Override
        public void onFailure(Call<String> call, Throwable t) {
            Toast.makeText(ActivityPerson.this, getString(R.string.errorNetwork),
Toast.LENGTH_SHORT).show();
        }
    });

}

//в окремому потоці відображаємо фото за посиланням
// create an async task class for loading pdf file from URL.
class RetrivePDFfromUrl extends AsyncTask<String, Void, InputStream> {
    @Override
    protected InputStream doInBackground(String... strings) {
        // we are using inputstream
        // for getting out PDF.
        InputStream inputStream = null;
        try {
            URL url = new URL(strings[0]);
            // below is the step where we are
            // creating our connection.
            HttpURLConnection urlConnection = null;
            try {
                urlConnection = (HttpsURLConnection) url.openConnection();
            } catch (IOException e) {
                e.printStackTrace();
            }
        }
    }
}

```

```

    }
    if (urlConnection.getResponseCode() == 200) {
        // response is success.
        // we are getting input stream from url
        // and storing it in our variable.
        inputStream = new BufferedInputStream(urlConnection.getInputStream());
    }

    } catch (IOException e) {
        // this is the method
        // to handle errors.
        e.printStackTrace();
        return null;
    }
    return inputStream;
}

@Override
protected void onPostExecute(InputStream inputStream) {
    // after the execution of our async
    // task we are loading our pdf in our pdf view.
    pdfView.fromStream(inputStream).load();
}
}

//перехід на основну сторінку
public void cancel(View view){
    Intent homeIntent = new Intent(Intent.ACTION_MAIN);
    homeIntent.addCategory( Intent.CATEGORY_HOME );
    homeIntent.setFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP);
    startActivity(homeIntent);
}

//тех.підтримка
public void btnSupport(View view) {

    Intent intent = new Intent(ActivityPerson.this, SupportActivity.class);
    startActivity(intent);
}

//перехід до перегляду документів користувача
public void document(View view) {
    Intent intent = new Intent(ActivityPerson.this, ActivityDocumentList.class);
    intent.putExtra("imei", imei);
    intent.putExtra("fio", fio);
    startActivity(intent);
}
}

```

ActivityDocumentList.java
package myhome.control;

/*підключення модулів та бібліотек
androidx.appcompat.app.AppCompatActivity;android.app.ProgressDialog;android.content.Intent;android.net.Uri;android.os.Bundle;android.util.Log;android.view.View;android.widget.AdapterView;android.widget.LinearLayout;android.widget.ListView;android.widget.SimpleAdapter;android.widget.Text

```
View;android.widget.Toast;com.google.gson.Gson;com.google.gson.GsonBuilder;com.google.gson.Json
Object;org.json.JSONException;org.json.JSONObject;java.io.File;java.io.IOException;java.util.ArrayList
;java.util.List;okhttp3.MediaType;okhttp3.MultipartBody;okhttp3.OkHttpClient;okhttp3.RequestBody;ok
http3.logging.HttpLoggingInterceptor;retrofit2.Call;retrofit2.Callback;retrofit2.Response;retrofit2.Retrofi
t;retrofit2.converter.gson.GsonConverterFactory;
```

```
*/
```

```
public class ActivityDocumentList extends AppCompatActivity {

    //прелоадер
    private ProgressDialog progressDialogSending;
        //екземпляри класів для роботи з сервером
    private Retrofit retrofit;
    private ServerApi serverApi;
        //виклик запиту
    private Call<ArrayList<Document>> callDocumentList;
        //прелік документів
    ArrayList<Document> documentArrayList = new ArrayList();
        //змінні для проміжних даних
    public String imei="";
    public static String fio = "";
        //візуальні елементи інтерфейсу
    TextView tvFio, tvEmptyList;
    ListView lvDocument;
        //адаптер переліку
    DocumentAdapter adapter;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_document_list);
        imei = getIntent().getStringExtra("imei");
        fio = getIntent().getStringExtra("fio");
        tvFio = (TextView) findViewById(R.id.fio);
        tvFio.setText(fio);
        lvDocument = (ListView) findViewById(R.id.listDocument);
        tvEmptyList = (TextView) findViewById(R.id.tvEmptyList);
        tvEmptyList.setVisibility(View.GONE);

        Gson gson = new GsonBuilder()
            .setLenient()
            .create();
        HttpLoggingInterceptor logging = new HttpLoggingInterceptor();
        logging.setLevel(HttpLoggingInterceptor.Level.BODY);
        OkHttpClient.Builder httpClient = new OkHttpClient.Builder();
        httpClient.addInterceptor(logging);
        retrofit = new Retrofit.Builder()
            .baseUrl("https://saas04.alldholding.com/wp-security/mobile_api/")
            .addConverterFactory(GsonConverterFactory.create(gson))
            //client(okHttpClient)
            .client(Extras.getClient(ActivityDocumentList.this))
            .build();
        serverApi = retrofit.create(ServerApi.class);
        getDocumentList();
    }
}
```

```

//тех.підтримка
public void btnSupport(View view) {

    Intent intent = new Intent(ActivityDocumentList.this, SupportActivity.class);
    startActivity(intent);
}

//отримання прееліку документів користувача
public void getDocumentList(){
    progressDialogSending = new ProgressDialog(ActivityDocumentList.this);
    progressDialogSending.show();
    progressDialogSending.setCancelable(false);

    RequestBody action_rb = RequestBody.create(MediaType.parse("multipart/form-data"),
"get_document_list");
    RequestBody imei_rb = RequestBody.create(MediaType.parse("multipart/form-data"),
imei);

    callDocumentList = serverApi.getDocumentList(action_rb,imei_rb);
    callDocumentList.enqueue(new Callback<ArrayList<Document>>() {
        @Override
        public void onResponse(Call<ArrayList<Document>> call,
Response<ArrayList<Document>> response) {
            progressDialogSending.dismiss();
            if (response.isSuccessful()){
                documentArrayList = response.body();

                adapter = new DocumentAdapter(ActivityDocumentList.this,
R.layout.document_name_list_item, documentArrayList);
                lvDocument.setAdapter(adapter);

                if (documentArrayList.size()==0){
                    tvEmptyList.setVisibility(View.VISIBLE);
                }
                progressDialogSending.dismiss();
            }else{

                Log.d("MyLog", "fail = " + response.errorBody());
                progressDialogSending.dismiss();
            }
        }

        @Override
        public void onFailure(Call<ArrayList<Document>> call, Throwable t) {
            progressDialogSending.dismiss();
            Toast.makeText(ActivityDocumentList.this, getString(R.string.errorNetwork),
Toast.LENGTH_SHORT).show();
        }
    });
}
}

```

ActivityDocument.java
package myhome.control;

```

/*підключення модулів та бібліотек
    androidx.appcompat.app.AppCompatActivity;android.app.ProgressDialog;android.content.Cont
ext;android.content.Intent;android.content.SharedPreferences;android.os.AsyncTask;android.os.Bundle;a
ndroid.util.Log;android.view.View;android.webkit.WebChromeClient;android.webkit.WebView;android.
webkit.WebViewClient;android.widget.LinearLayout;android.widget.ProgressBar;android.widget.TextVi
ew;android.widget.Toast;com.github.barteksc.pdfviewer.PDFView;com.google.gson.Gson;com.google.gs
on.GsonBuilder;java.io.BufferedInputStream;java.io.IOException;java.io.InputStream;java.net.HttpURL
Connection;java.net.URL;java.util.ArrayList;javax.net.ssl.HttpURLConnection;okhttp3.MediaType;okht
tp3.OkHttpClient;okhttp3.RequestBody;okhttp3.logging.HttpLoggingInterceptor;retrofit2.Call;retrofit2.C
allback;retrofit2.Response;retrofit2.Retrofit;retrofit2.converter.gson.GsonConverterFactory;
*/

```

```

public class ActivityDocument extends AppCompatActivity {
//прелоадер
    private ProgressDialog progressDialogSending;
        //екземпляри класів для роботи з сервером
    private Retrofit retrofit;
    private ServerApi serverApi;
        //виклик запиту
    private Call<String> callDocumentURL;
        //проміжні змінні
    int id_name_document;
    String name_document,fio;
        //елемент інтерфейсу для відображення документа
    PDFView pdfView;
    String pdfurl="";
    TextView tvFioDocument;

    SharedPreferences mSettings;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_document);
        id_name_document= getIntent().getIntExtra("id_name_document",0);
        name_document= getIntent().getStringExtra("name_document");
        fio= getIntent().getStringExtra("fio");
        tvFioDocument = (TextView) findViewById(R.id.fio_document);
        if (name_document != null & !name_document.isEmpty()){
            if (fio != null & !fio.isEmpty()){
                tvFioDocument.setText(fio+"\n"+name_document);
            }
        }
        Log.d("MyLog", " fio="+fio);

        Gson gson = new GsonBuilder()
            .setLenient()
            .create();
        HttpLoggingInterceptor logging = new HttpLoggingInterceptor();
        logging.setLevel(HttpLoggingInterceptor.Level.BODY);
        OkHttpClient.Builder httpClient = new OkHttpClient.Builder();
        httpClient.addInterceptor(logging);
        retrofit = new Retrofit.Builder()
            .baseUrl("https://saas04.alldholding.com/wp-security/mobile_api/")
            .addConverterFactory(GsonConverterFactory.create(gson))
            .client(okHttpClient)

```

```

        .client(Extras.getClient(ActivityDocument.this))
        .build();
serverApi = retrofit.create(ServerApi.class);

getDocumentURL();
pdfView = findViewById(R.id.idPDFView);

}

//отримання посилання на документ
public void getDocumentURL(){
    progressDialogSending = new ProgressDialog(ActivityDocument.this);

    progressDialogSending.show();
    progressDialogSending.setCancelable(false);

    RequestBody action_rb = RequestBody.create(MediaType.parse("multipart/form-data"),
"get_file_path");
    RequestBody id_name_document_rb =
RequestBody.create(MediaType.parse("multipart/form-data"), String.valueOf(id_name_document));
    mSettings = getSharedPreferences(MainActivity.APP_PREFERENCES,
Context.MODE_PRIVATE);

    RequestBody imei_rb = RequestBody.create(MediaType.parse("multipart/form-data"),
mSettings.getString(MainActivity.APP_PREFERENCES_RESPONSIBLE,""));
    callDocumentURL =
serverApi.getDocumentURL(action_rb,id_name_document_rb,imei_rb);
    callDocumentURL.enqueue(new Callback<String>() {
        @Override
        public void onResponse(Call<String> call, Response<String> response) {
            progressDialogSending.dismiss();
            if (response.isSuccessful()){
                pdfurl = response.body();
                Log.d("HERE", "pdfurl=" + pdfurl);
                new RetrivePDFfromUrl().execute(pdfurl);
                progressDialogSending.dismiss();
            }else{
                Log.d("MyLog", "fail = " + response.errorBody());
                progressDialogSending.dismiss();
            }
        }
    })

    @Override
    public void onFailure(Call<String> call, Throwable t) {
        progressDialogSending.dismiss();
        Toast.makeText(ActivityDocument.this, getString(R.string.errorNetwork),
Toast.LENGTH_SHORT).show();
    }
});
}

```

```

//відтворення документу за посиланням
// create an async task class for loading pdf file from URL.
class RetrivePDFfromUrl extends AsyncTask<String, Void, InputStream> {
    @Override
    protected InputStream doInBackground(String... strings) {
        // we are using inputstream
        // for getting out PDF.
        InputStream inputStream = null;
        try {
            URL url = new URL(strings[0]);
            // below is the step where we are
            // creating our connection.
            HttpURLConnection urlConnection = null;
            try {
                urlConnection = (HttpsURLConnection) url.openConnection();
            } catch (IOException e) {
                e.printStackTrace();
            }
            if (urlConnection.getResponseCode() == 200) {
                // response is success.
                // we are getting input stream from url
                // and storing it in our variable.
                inputStream = new BufferedInputStream(urlConnection.getInputStream());
            }

            } catch (IOException e) {
                // this is the method
                // to handle errors.
                e.printStackTrace();
                return null;
            }
            return inputStream;
        }

        @Override
        protected void onPostExecute(InputStream inputStream) {
            // after the execution of our async
            // task we are loading our pdf in our pdf view.
            pdfView.fromStream(inputStream).load();
        }
    }
}

//тех.підтримка
public void btnSupport(View view) {

    Intent intent = new Intent(ActivityDocument.this, SupportActivity.class);
    startActivity(intent);
}

}

```

```

SupportActivity.java
package myhome.control;

```

```

/*підключення модулів та бібліотек

```

```

    android.Manifest;android.app.ProgressDialog;android.content.ContentResolver;android.content.
    Context;android.content.DialogInterface;android.content.Intent;android.content.SharedPreferences;androi
    d.content.pm.PackageManager;android.database.Cursor;android.net.Uri;android.os.Build;android.os.Envi
    ronment;android.provider.MediaStore;androidx.annotation.NonNull;androidx.annotation.Nullable;androi
    dx.core.app.ActivityCompat;androidx.core.content.FileProvider;androidx.appcompat.app.AppCompatActivity;
    android.os.Bundle;android.util.Log;android.view.ContextMenu;android.view.Menu;android.view.M
    enuItem;android.view.View;android.webkit.MimeTypeMap;android.widget.Button;android.widget.EditT
    ext;android.widget.ImageView;android.widget.LinearLayout;android.widget.TextView;android.widget.T
    oast;com.google.gson.Gson;com.google.gson.GsonBuilder;com.google.gson.JsonObject;org.json.JSONE
    xception;org.json.JSONObject;java.io.File;java.io.IOException;java.text.SimpleDateFormat;java.util.Arr
    ayList;java.util.Date;java.util.List;okhttp3.MediaType;okhttp3.MultipartBody;okhttp3.OkHttpClient;okht
    tp3.RequestBody;okhttp3.logging.HttpLoggingInterceptor;retrofit2.Call;retrofit2.Callback;retrofit2.Respo
    nse;retrofit2.Retrofit;retrofit2.converter.gson.GsonConverterFactory;

```

```

    */

```

```

    public class SupportActivity extends AppCompatActivity {
        //-----

```

```

        public final static int REQUEST_CODE_PERMISSION_WRITE_EXTERNAL_STORAGE
= 201;

```

```

        //прелоадер
        private ProgressDialog progressDialogSending;
        //ключі
        private final String CAMERA = "камера";
        private final String GALLERY = "галерея";
        private final int CAMERA_METHOD = 1;
        private final int GALLERY_METHOD = 2;
        private final int DELETE_PHOTO = 101;
        private final int ACTIVITY_ALERT = 777;
        private String mCurrentPhotoPath = "";
        private Uri photoURI;
        //візуальні елементи інтерфейсу
        private TextView tvTitelMail, tvTitleSupport;
        private EditText etMail, etTopic;
        private ImageView btnAddPhoto, btnAddPicture;
        private Button btnSend;
        private View vSelectedPhotoInContextMenu;
        private LinearLayout llSupport;
        //структури преліків фото
        private ArrayList<ImageView> arrayViewPhotos;
        private ArrayList<File> arrayFilePhotos;

        //робота з сервером
        private Retrofit retrofit;
        private ServerApi serverApi;
        private Call<JsonObject> callSupport;

        //-----
        //глобальні налаштування користувача
        SharedPreferences mSettings;

        @Override
        protected void onCreate(Bundle savedInstanceState) {
            super.onCreate(savedInstanceState);
            setContentView(R.layout.activity_support);

```

```

        // перевірка дозволів
        if (Manifest.permission.WRITE_EXTERNAL_STORAGE)
            PackageManager.PERMISSION_GRANTED) {

            ActivityCompat.requestPermissions(SupportActivity.this,
                new String[] {
                    Manifest.permission.WRITE_EXTERNAL_STORAGE,
                    Manifest.permission.READ_EXTERNAL_STORAGE},
                SupportActivity.REQUEST_CODE_PERMISSION_WRITE_EXTERNAL_STORAGE);
        }

        // масив для зберігання View фото
        arrayViewPhotos = new ArrayList<>();
        // масив для зберігання File фото
        arrayFilePhotos = new ArrayList<>();
        Gson gson = new GsonBuilder()
            .setLenient()
            .create();
        HttpLoggingInterceptor logging = new HttpLoggingInterceptor();
        logging.setLevel(HttpLoggingInterceptor.Level.BODY);
        OkHttpClient.Builder httpClient = new OkHttpClient.Builder();
        httpClient.addInterceptor(logging);

        retrofit = new Retrofit.Builder()
            .baseUrl("https://saas04.alldholding.com/wp-security/mobile_api/")
            .addConverterFactory(GsonConverterFactory.create(gson))
            //.client(okHttpClient)
            .client(Extras.getClient(SupportActivity.this))
            .build();
        serverApi = retrofit.create(ServerApi.class);

        LinkView();

        String userName = "";
        if (!userName.equals("")) {
            tvTitleSupport.setText("Вітаємо " + userName + "! \n Ви звернулися до
тех.підтримки.");
        } else {
            tvTitleSupport.setText("Вітаємо! \n Ви звернулися до тех.підтримки.");
        }
        // Натискання на кнопку додавання нового фото через камеру
        btnAddPhoto.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                dispatchTakePictureIntent();
            }
        });

        // Натискання на кнопку додавання нового фото через галерею
        btnAddPicture.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {

```

```

// ACTION_OPEN_DOCUMENT is the intent to choose a file via the system's file
// browser.
Intent intent = new Intent(Intent.ACTION_GET_CONTENT);

// Filter to only show results that can be "opened", such as a
// file (as opposed to a list of contacts or timezones)

// Filter to show only images, using the image MIME data type.
// If one wanted to search for ogg vorbis files, the type would be "audio/ogg".
// To search for all documents available via installed storage providers,
// it would be "*/*".
intent.setType("image/*");

startActivityForResult(intent, GALLERY_METHOD);

}
});

// кнопка надсилання
btnSend.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        // валідуємо дані
        if (Validate()){
            // надсилаємо дані на сервер
            Send();
        }
    }
});

}

// прив'язуємо View
private void LinkView(){

    tvTitleSupport = (TextView)findViewById(R.id.tvSupportTitel);
    tvTitelMail = (TextView)findViewById(R.id.tvSupportTitelMail);

    etMail = (EditText)findViewById(R.id.etSupportMail);
    etTopic = (EditText)findViewById(R.id.etSupportTopic);

    btnAddPhoto = (ImageView)findViewById(R.id.ivAddPhoto);
    btnAddPicture = (ImageView)findViewById(R.id.ivAddPicture);
    btnSend = (Button)findViewById(R.id.btnSupportSend);

    llSupport = (LinearLayout)findViewById(R.id.llSupport);

}

// валідація введених даних
private boolean Validate(){

    if (Empty(etTopic)){ return false; }
    if (Empty(etMail)){ return false; }

```

```

        return true;
    }

    // перевірка на порожність
    private boolean Empty(EditText field){

        if (field.getText().toString().trim().length() == 0){
            field.requestFocus();
            Toast.makeText(SupportActivity.this,"Заповніть поле",
Toast.LENGTH_SHORT).show();
            return true;
        }

        return false;
    }

    // створюємо новий ImageView для відображення обраного зображення
    private ImageView CreateNewImageView(){
        ImageView imageView = new ImageView(SupportActivity.this);
        LinearLayout.LayoutParams layoutParams = new
        LinearLayout.LayoutParams(LinearLayout.LayoutParams.MATCH_PARENT,
        LinearLayout.LayoutParams.WRAP_CONTENT, 1);
        layoutParams.setMargins(30, 20, 30, 0);

        // позиція елемента на LinearLayout
        int postion = (llSupport.getChildCount()-2);
        imageView.setAdjustViewBounds(true);
        imageView.setLayoutParams(layoutParams);

        // додаємо на вказану позицію
        llSupport.addView(imageView, postion);
        registerContextMenu(imageView);
        return imageView;
    }

    // відправка даних на сервер
    private void Send(){

        progressDialogSending = new ProgressDialog(SupportActivity.this);

        progressDialogSending.show();
        progressDialogSending.setCancelable(false);

        String mail = etMail.getText().toString().trim();
        String topic = etTopic.getText().toString().trim();

        mSettings = getSharedPreferences(MainActivity.APP_PREFERENCES,
Context.MODE_PRIVATE);

        RequestBody name_rb = RequestBody.create(MediaType.parse("multipart/form-data"),
mSettings.getString(MainActivity.APP_PREFERENCES_DISPLAY_NAME, ""));
        RequestBody company_rb = RequestBody.create(MediaType.parse("multipart/form-data"),
mSettings.getString(MainActivity.APP_PREFERENCES_COMPANY_NAME, ""));

```

```

mail);
        requestBody mail_rd = RequestBody.create(MediaType.parse("multipart/form-data"),
topic);
        requestBody topic_rb = RequestBody.create(MediaType.parse("multipart/form-data"),
topic);

        List<MultipartBody.Part> partsImages = new ArrayList<>();
        for (File image : arrayFilePhotos){
            partsImages.add(Extras.preparePart(image.getName(),
getMimeType(Uri.fromFile(image))));
        }

        serverApi = retrofit.create(ServerApi.class);
        callSupport = serverApi.callSupport( topic_rb, mail_rd, name_rb, company_rb,
partsImages);
        callSupport.enqueue(new Callback<JsonObject>() {
            @Override
            public void onResponse(Call<JsonObject> call, Response<JsonObject> response) {
                progressDialogSending.dismiss();
                if (response.isSuccessful()){
                    JsonObject jsonObject = response.body();
                    Toast.makeText(SupportActivity.this, jsonObject.get("message").getAsString(),
Toast.LENGTH_LONG).show();
                    finish();
                }else{
                    try {
                        JSONObject jsonObject = new
JSONObject(response.errorBody().string().trim());
                        DialogError("Ошибка", String.valueOf(jsonObject.get("message")));
                    } catch (IOException | JSONException e) {
                        e.printStackTrace();
                    }
                }
            }

            @Override
            public void onFailure(Call<JsonObject> call, Throwable t) {
                progressDialogSending.dismiss();
                Toast.makeText(SupportActivity.this, getString(R.string.errorNetwork),
Toast.LENGTH_SHORT).show();
            }
        });

    }

    // отримання реального шляху до файлу
    public String getRealPathFromUri(Uri contentUri) {
        Cursor cursor = null;
        try {
            String[] proj = {MediaStore.Images.Media.DATA};
            cursor = getContentResolver().query(contentUri, proj, null, null, null);
            assert cursor != null;
            int column_index =
cursor.getColumnIndexOrThrow(MediaStore.Images.Media.DATA);
            cursor.moveToFirst();
            return cursor.getString(column_index);
        } finally {
            if (cursor != null) {

```

```

        cursor.close();
    }
}

```

```

public void IntertnetNoOkClicked() {
}

```

```

// тип файлу
public String getMimeType(Uri uri) {
    String mimeType = null;
    if (uri.getScheme().equals(ContentResolver.SCHEME_CONTENT)) {
        ContentResolver cr = getContentResolver();
        mimeType = cr.getType(uri);
    } else {
        String fileExtension = MimeTypeMap.getFileExtensionFromUrl(uri
            .toString());
        mimeType = MimeTypeMap.getSingleton().getMimeTypeFromExtension(
            fileExtension.toLowerCase());
    }
    return mimeType;
}

```

```

@Override
protected void onActivityResult(int requestCode, int resultCode, @Nullable Intent data) {
    super.onActivityResult(requestCode, resultCode, data);
    // після обраної з галереї фотографії
    if (requestCode == GALLERY_METHOD && resultCode == RESULT_OK){
        Uri selectedImage = data.getData();
        ImageView imageView = CreateNewImageView();
        imageView.setImageURI(selectedImage);

        arrayViewPhotos.add(imageView);

        arrayFilePhotos.add(new File(getRealPathFromUri(selectedImage)));

    }
    // після зробленого фото
    if (requestCode == CAMERA_METHOD ) {
        if (resultCode == RESULT_OK){
            galleryAddPic();
            ImageView imageView = CreateNewImageView();
            imageView.setImageURI(photoURI);
            arrayViewPhotos.add(imageView);
            arrayFilePhotos.add(new File(mCurrentPhotoPath));
            Log.d("MyLog", mCurrentPhotoPath);
        }else{
            File fileRemove = new File(mCurrentPhotoPath);
            fileRemove.delete();
            photoURI = null;
            mCurrentPhotoPath = "";
        }
    }
}

```

```

        if (requestCode == ACTIVITY_ALERT){
            finish();
        }
    }

    // відкриття камери
    private void dispatchTakePictureIntent() {

        Intent takePictureIntent = new Intent(MediaStore.ACTION_IMAGE_CAPTURE);
        // Ensure that there's a camera activity to handle the intent
        if (takePictureIntent.resolveActivity(SupportActivity.this.getPackageManager()) != null) {
            File photoFile = null;
            try {
                photoFile = createImageFile();
            } catch (IOException ex) {
                Toast.makeText(SupportActivity.this, "Помилка створення файла фото!",
                    Toast.LENGTH_SHORT).show();
            }
            if (photoFile != null) {
                if (android.os.Build.VERSION.SDK_INT >= Build.VERSION_CODES.LOLLIPOP)
                {
                    photoURI = FileProvider.getUriForFile(SupportActivity.this,
                        BuildConfig.APPLICATION_ID + ".provider",
                        photoFile);
                } else {
                    photoURI = Uri.fromFile(photoFile);
                }
                takePictureIntent.putExtra(MediaStore.EXTRA_OUTPUT, photoURI);
                startActivityForResult(takePictureIntent, CAMERA_METHOD);

            }
        }
    }

    //створення файлу після фотографування
    private File createImageFile() throws IOException {
        SimpleDateFormat sdf = new SimpleDateFormat("yyyyMMddHHmmss");
        String currentDateTime= sdf.format(new Date());
        String imageFileName = "support_" + currentDateTime ;
        File storageDir
        Environment.getExternalStoragePublicDirectory(Environment.DIRECTORY_PICTURES);
        File image = File.createTempFile(
            imageFileName, /* prefix */
            ".jpg",        /* suffix */
            storageDir      /* directory */
        );

        mCurrentPhotoPath = image.getAbsolutePath();
        return image;
    }

    // оновлення галереї для фото
    private void galleryAddPic() {
        Intent mediaScanIntent = new Intent(Intent.ACTION_MEDIA_SCANNER_SCAN_FILE);

```

```

        File f = new File(mCurrentPhotoPath);
        Uri contentUri = Uri.fromFile(f);
        mediaScanIntent.setData(contentUri);
        this.sendBroadcast(mediaScanIntent);
    }

    @Override
    // створення контекстного меню
    public void onCreateContextMenu(ContextMenu menu, View v,
        ContextMenu.ContextMenuInfo menuInfo) {
        super.onCreateContextMenu(menu, v, menuInfo);
        vSelectedPhotoInContextMenu = v;
        menu.add(Menu.NONE, DELETE_PHOTO, Menu.NONE, "Видалити");
    }

    @Override
    // обробка натискання на елементи контекстного меню
    public boolean onOptionsItemSelected(MenuItem item) {
        if (item.getItemId() == DELETE_PHOTO){
            for (int i = 0; i<arrayViewPhotos.size(); i++){
                if (vSelectedPhotoInContextMenu == (View)arrayViewPhotos.get(i)){
                    llSupport.removeView(vSelectedPhotoInContextMenu);
                    vSelectedPhotoInContextMenu = null;
                    arrayViewPhotos.remove(i);
                    arrayFilePhotos.remove(i);
                }
            }
        }
        return super.onOptionsItemSelected(item);
    }

    // діалог з помилкою
    private void DialogError(String titel, String text){
        android.app.AlertDialog.Builder builder = new
        android.app.AlertDialog.Builder(SupportActivity.this);
        builder.setTitle(titel)
            .setMessage(text)
            .setCancelable(false)
            .setNegativeButton("OK",
                new DialogInterface.OnClickListener() {
                    public void onClick(DialogInterface dialog, int id) {
                        dialog.cancel();
                    }
                });
        android.app.AlertDialog alert = builder.create();
        alert.show();
    }

    @Override
    public void onRequestPermissionsResult(int requestCode, @NonNull String[] permissions,
        @NonNull int[] grantResults) {
        switch (requestCode) {
            case
            SupportActivity.REQUEST_CODE_PERMISSION_WRITE_EXTERNAL_STORAGE:
                if (grantResults.length > 0){

```

```

        } else {
            finish();
        }
        return;
    }
}

public void btnClick(View view){
    switch (view.getId()){
        case R.id.btn_left:

            finish();

            break;

        }
    }
}

```

DataBaseHelper.java

package myhome.control;

/*підключення модулів та бібліотек

android.content.ContentValues;android.content.Context;android.database.sqlite.SQLiteDatabase
;android.content.ContentValues;android.content.Context;android.database.Cursor;android.database.sqlite.
SQLiteDatabase;android.database.sqlite.SQLiteOpenHelper;android.util.Log;java.text.SimpleDateFormat
;java.util.ArrayList;java.util.Date;
*/

```

public class DataBaseHelper extends SQLiteOpenHelper {
    //змінні для роботи з бд локальною
    public static final String TABLE_NAME = "passport";
    public static final String TABLE_NAME_SAVE= "otchet";
    public static final String TABLE_COMPANY="company";
    public static final String TABLE_POSITION="position";
    public static final String COLUM_ID = "id";
    public static final String COLUM_1= "imei";
    public static final String COLUM_2= "surname";
    public static final String COLUM_3= "name";
    public static final String COLUM_4= "patronymic";
    public static final String COLUM_5= "code_employee";
    public static final String COLUM_6= "PN";
    public static final String COLUM_7= "code";
    public static final String COLUM_8= "name_position";
    public static final String COLUM_9= "code_position";
    public static final String COLUM_10= "lat_input";
    public static final String COLUM_11= "long_input";
    public static final String COLUM_12= "lat_output";
    public static final String COLUM_13= "long_output";
    public static final String COLUM_14= "mark";

```

//контекст

private Context context;

//екземпляр бд

```

protected SQLiteDatabase db= getWritableDatabase();

public DataBaseHelper(Context context) {
    super(context, "MyDB", null, 12);
    this.context=context;
}
//створення первинне таблиць у бд
@Override
public void onCreate(SQLiteDatabase db) {
    db.execSQL("CREATE TABLE "+ TABLE_NAME +" ("
        + COLUMN_ID +" INTEGER PRIMARY KEY AUTOINCREMENT,"
        + COLUMN_1 +" TEXT,"+ COLUMN_2 +" TEXT,"+ COLUMN_3+" TEXT,"+
COLUMN_4 +" TEXT,"+ COLUMN_5 +" TEXT,"+ COLUMN_6 +" TEXT)");

    db.execSQL("CREATE TABLE "+ TABLE_NAME_SAVE +" ("
        + COLUMN_ID +" INTEGER PRIMARY KEY AUTOINCREMENT,"+ COLUMN_1
+" TEXT,"+ COLUMN_2 +" TEXT,"+ COLUMN_3+" TEXT,"+ COLUMN_4 +" TEXT,"+ COLUMN_5 +"
TEXT,"+ COLUMN_6 +" TEXT,"+COLUMN_7 +" REAL,"+COLUMN_10 +" REAL,"+COLUMN_11 +"
REAL,"+COLUMN_12 +" REAL,"+COLUMN_13 +" REAL,"+COLUMN_14 +" INTEGER)");

    db.execSQL("CREATE TABLE "+ TABLE_COMPANY +" ("
        + COLUMN_ID +" INTEGER PRIMARY KEY AUTOINCREMENT,"+ COLUMN_3
+" TEXT,"+ COLUMN_7 +" TEXT)");

    db.execSQL("CREATE TABLE "+ TABLE_POSITION +" ("
        + COLUMN_ID +" INTEGER PRIMARY KEY AUTOINCREMENT,"+ COLUMN_8
+" TEXT,"+ COLUMN_9 +" TEXT)");
}
//оновлення -для відладки
@Override
public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
    /*Log.d("MyLog", "DataBaseHelper onUpgrade");
    db.execSQL("ALTER TABLE " +
        TABLE_NAME_SAVE + " ADD COLUMN " + COLUMN_10 + " REAL");
    db.execSQL("ALTER TABLE " +
        TABLE_NAME_SAVE + " ADD COLUMN " + COLUMN_11 + " REAL");
    db.execSQL("ALTER TABLE " +
        TABLE_NAME_SAVE + " ADD COLUMN " + COLUMN_12 + " REAL");
    db.execSQL("ALTER TABLE " +
        TABLE_NAME_SAVE + " ADD COLUMN " + COLUMN_13 + " REAL");*/
}

//вставка даних про підрозділи в бд
public void insertCompanyLocalDB(Company comp){

    SQLiteDatabase db = getWritableDatabase();
    try{
        ContentValues contentValues = new ContentValues();
        contentValues.put("id", comp.id);
        contentValues.put("name", comp.name_company);

        comp.id = db.insert(Company.table, null, contentValues);
    }catch (Exception e){
        Log.d("MyLog", "DataBaseHelper @ insert " + e.getMessage());
    }
    //return site;
}

```

```

//отримання користувачів бд
public ArrayList<UserTest> getUsers(){
    SQLiteDatabase db = getWritableDatabase();
    ArrayList<UserTest> users = new ArrayList<>();
    Cursor cursor = null;
    try {
        cursor = db.query(TABLE_NAME, null, null, null, null, null, null);
        if (cursor.moveToFirst()){
            while (!cursor.isAfterLast()){
                UserTest user = new UserTest();

                user.id = cursor.getLong(cursor.getColumnIndex(COLUM_ID));

                users.add(user);
                cursor.moveToNext();
            }
        }
    }catch (Exception e){
        e.printStackTrace();
    }finally {
        if (cursor != null) {
            cursor.close();
        }
    }

    return users;
}
}

```

ДОДАТОК Б
Слайди презентації

Дослідження та програмна реалізація методів обліку робочого часу на підприємствах

Виконала:

ст. гр. КНТ-111м

Мартиненко Г.Ю.

Керівник:

к.т.н., доцент

Каплієнко Т. І.

Рисунок Б.1 – Слайд 1

2

Огляд роботи

- Об'єкт дослідження – методи обліку робочого часу підприємства.
- Предмет дослідження – комп'ютерні програми для обліку робочого часу працівників на підприємстві.
- Мета роботи – створення програмної системи для автоматизації та прискорення процесу обліку робочого часу нормувальниками підприємства.

Рисунок Б.2 – Слайд 2

3

Огляд існуючих реалізацій

Трекінг за допомогою смарт-годинників [22]



Переваги	Недоліки
Мобільність системи	Залежність від погодних умов
Сучасність технології	Залежність від висот місцевості
Достовірність інформації	Висока собівартість
	Високий поріг навичок для впровадження

Система Контроля та Управління Доступом [23]

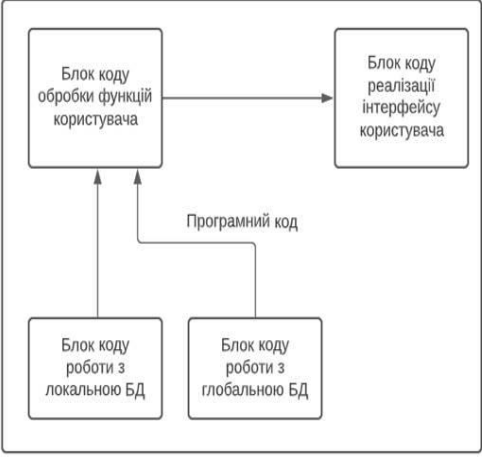
Переваги	Недоліки
Достовірність інформації	Статичність
Незалежність від погодних умов	Висока собівартість



Рисунок Б.3 – Слайд 3

Умовний поділ програмного коду на блоки

4



Концептуальна схема взаємодії програмного продукту



Рисунок Б.4 – Слайд 4

Use-case діаграма функцій користувача

5

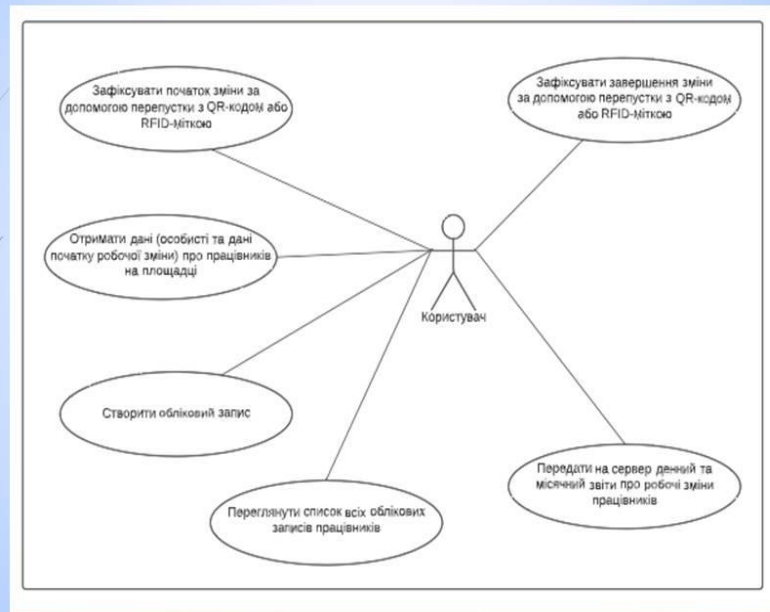
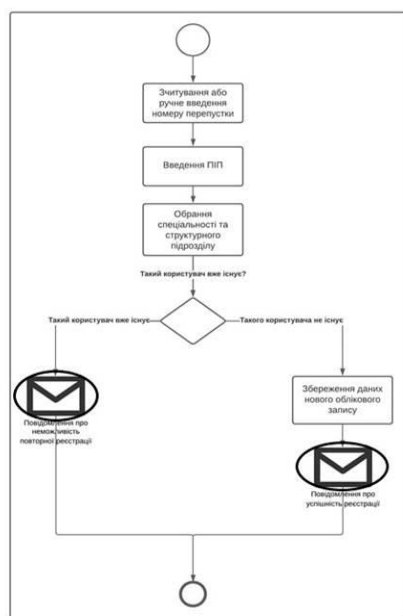


Рисунок Б.5 – Слайд 5

BPMN 2.0 діаграма реєстрації облікового запису користувача

6



BPMN 2.0 діаграма подій загального випадку фіксації початку робочої зміни

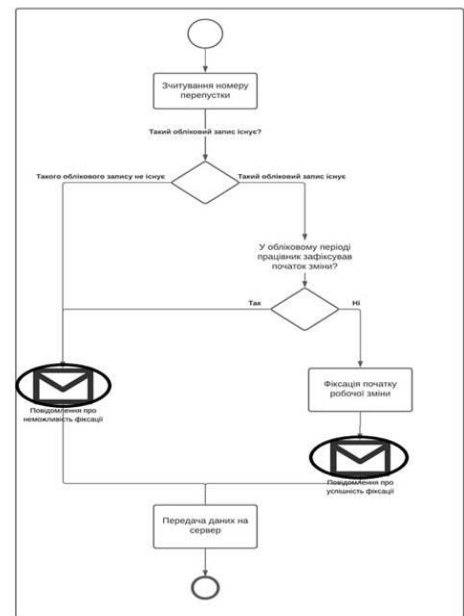
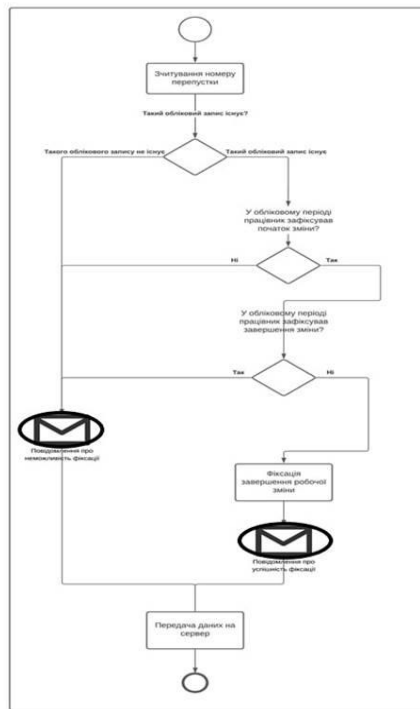


Рисунок Б.6 – Слайд 6

7



BPMN 2.0 діаграма подій загального випадку фіксації завершення робочої зміни

Передбачене зчитування перепусток двох типів:

- RFID-мітка;
- QR-код.

RFID-перепустка зчитується за допомогою портативного або вбудованого зчитувача.

QR-перепустка зчитується за допомогою камери пристрою.

Рисунок Б.7 – Слайд 7

8

Обґрунтування вибору мови програмування

Порівняння мов Java і Swift

	Java	Swift
Нативність для Android	+	-
Популярність	Дуже популярний	Дуже популярний
Фреймворки	Багато фреймворків	Мало фреймворків
Вивчення	Легше вчити	Складніше вчити
Схожість з C++	+	-
Надійність та захищеність від розповсюджених помилок	+	-

Рисунок Б.8 – Слайд 8

Структура проєкту

9

Нижче будуть наведені файли розмітки з описом призначення, що формують інтерфейс користувача:

- activity_add.xml – сторінка створення нового облікового запису користувача;
- activity_bd.xml – сторінка перегляду переліку всіх облікових записів користувачів;
- activity_document.xml – сторінка перегляду дозвільного документу користувача;
- activity_document_list.xml – сторінка перегляду переліку дозвільних документів користувача;
- activity_person.xml – сторінка перегляду карточки облікового запису користувача;
- activity_registration.xml – сторінка основного меню користувача;
- activity_report.xml – сторінка перегляду переліку співробітників, що знаходяться на зміні у поточний момент часу, відправка щоденного звіту на сервер;
- activity_send_all_report.xml – сторінка відправки місячного звіту на сервер;
- activity_support.xml – сторінка звернень до технічної підтримки;
- activity_trackers.xml – сторінка фіксації початку та завершення робочої зміни.

Перелік файлів з реалізацією логіки:

- TrackersActivity.java;
- Trackers_backActivity.java;
- BDActivity.java;
- RegistrationActivity.java;
- ReportActivity.java;
- AddActivity.java;
- SendAllReportActivity.java;
- ActivityPerson.java;
- ActivityDocumentList.java;
- ActivityDocument.java;
- SupportActivity.java;
- DataBaseHelper.java;
- ServerApi.java.

Рисунок Б.9 – Слайд 9

Інтерфейс (основне та додаткове меню відповідно)

10

Рисунок Б.10 – Слайд 10

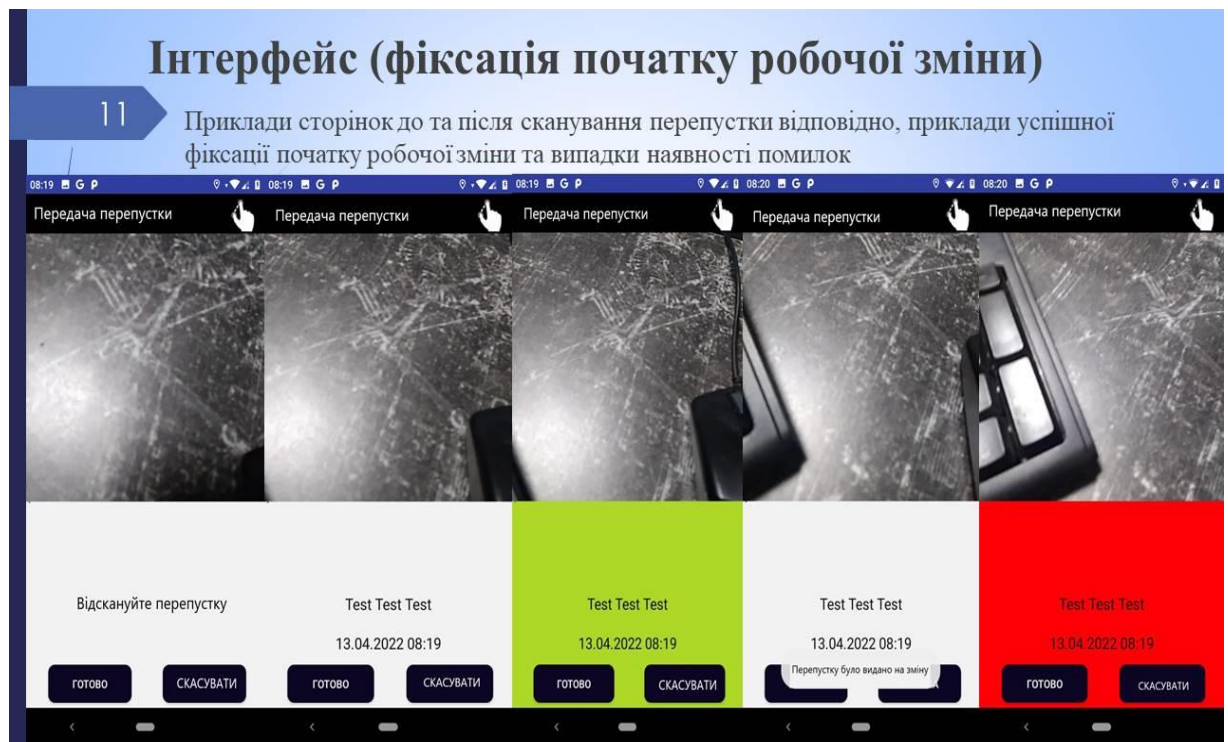


Рисунок Б.11 – Слайд 11

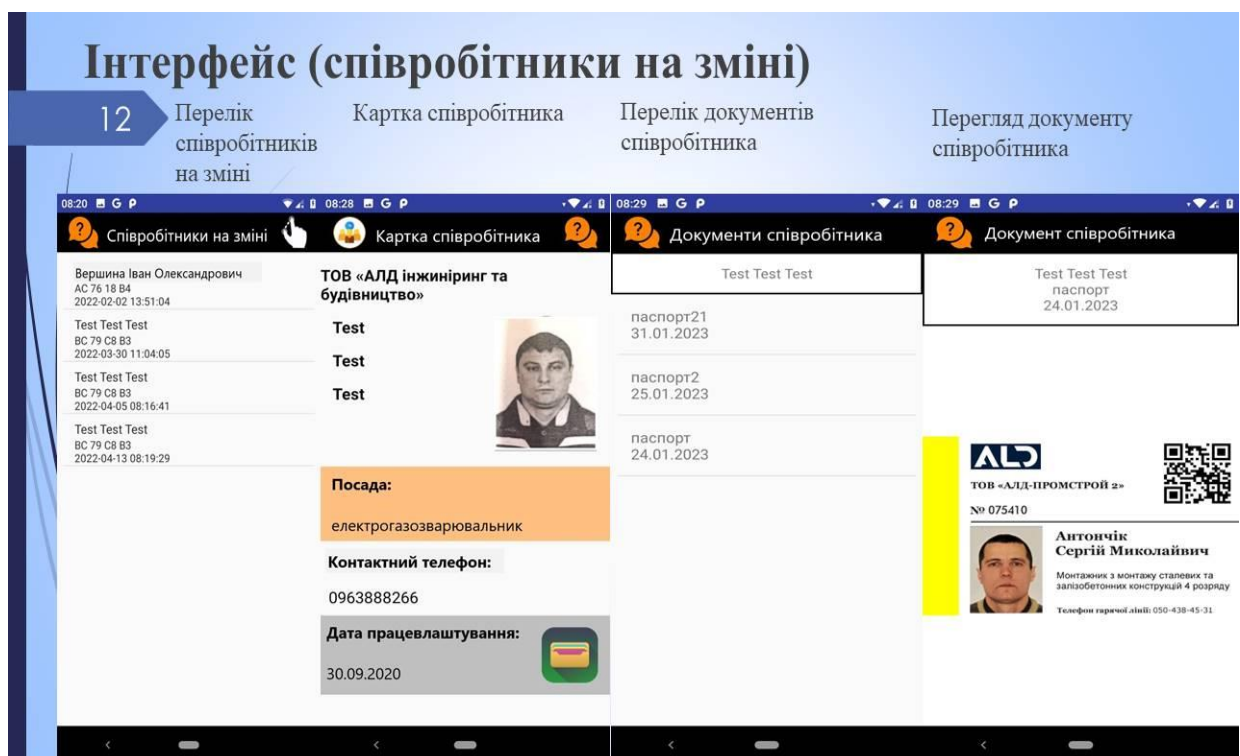


Рисунок Б.12 – Слайд 12

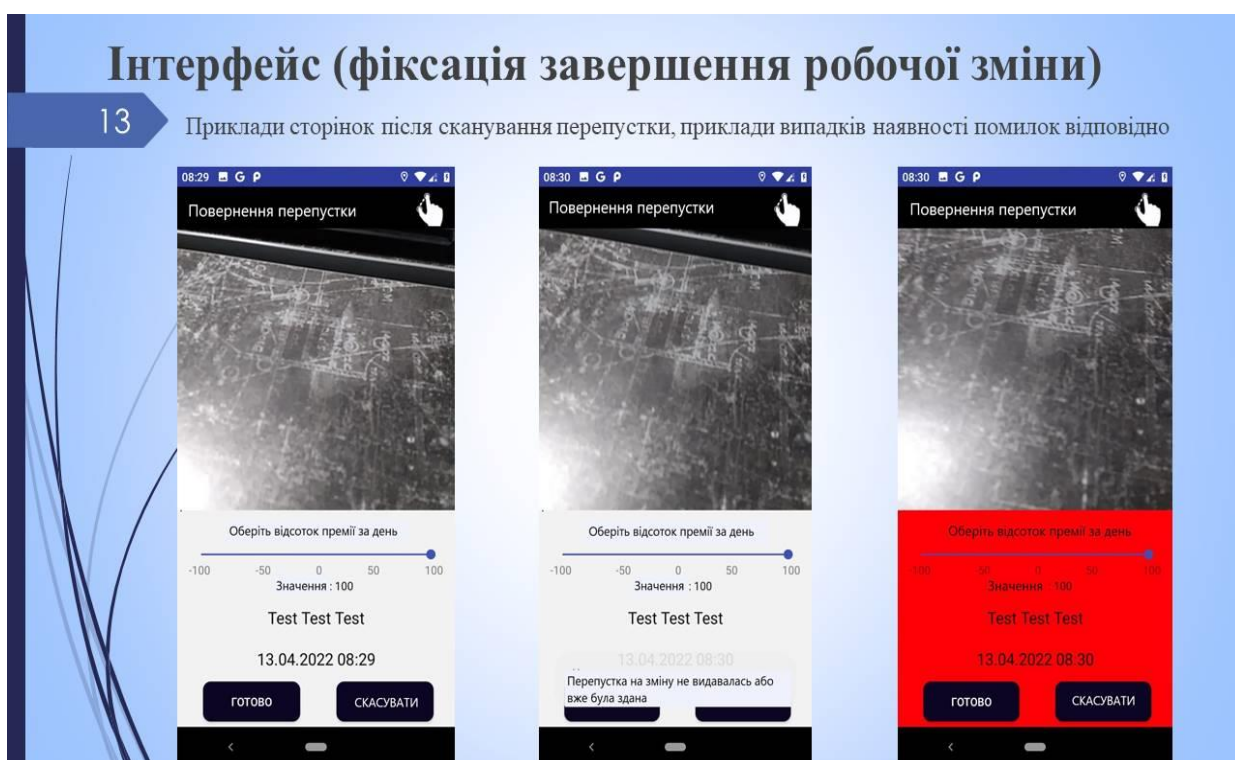


Рисунок Б.13 – Слайд 13

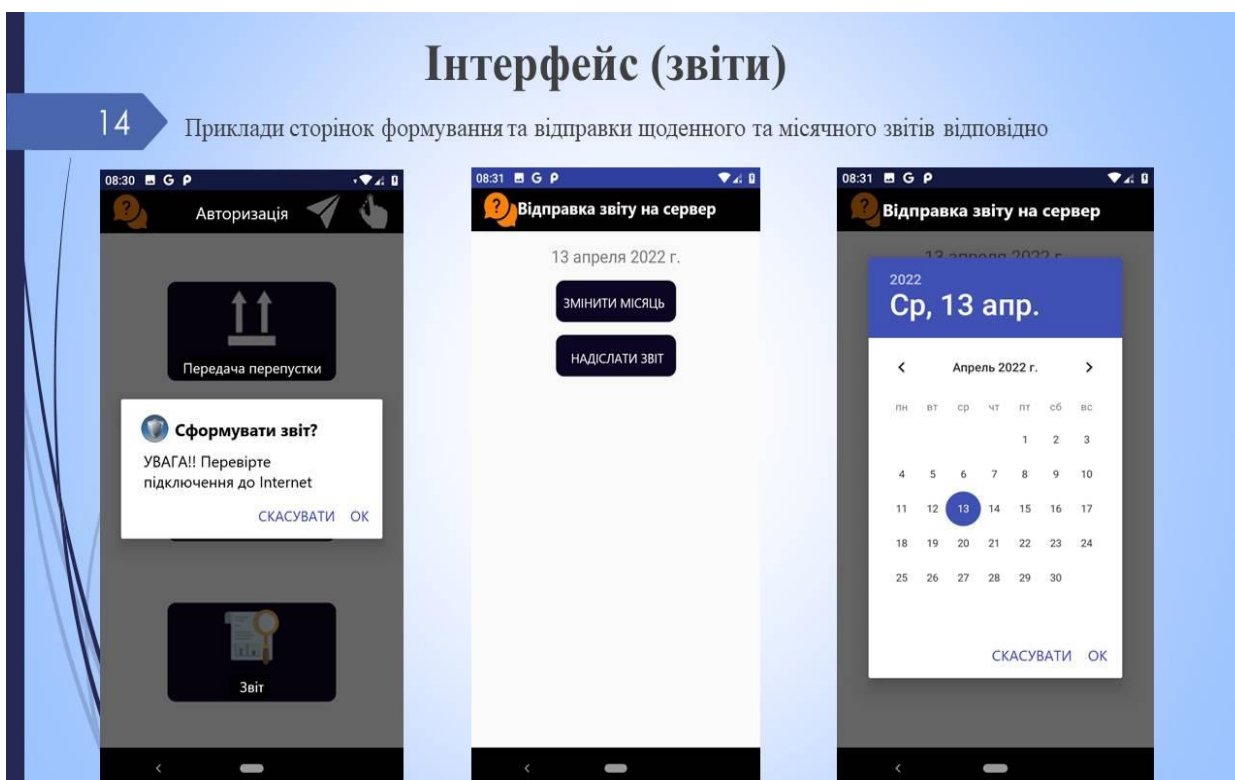


Рисунок Б.14 – Слайд 14

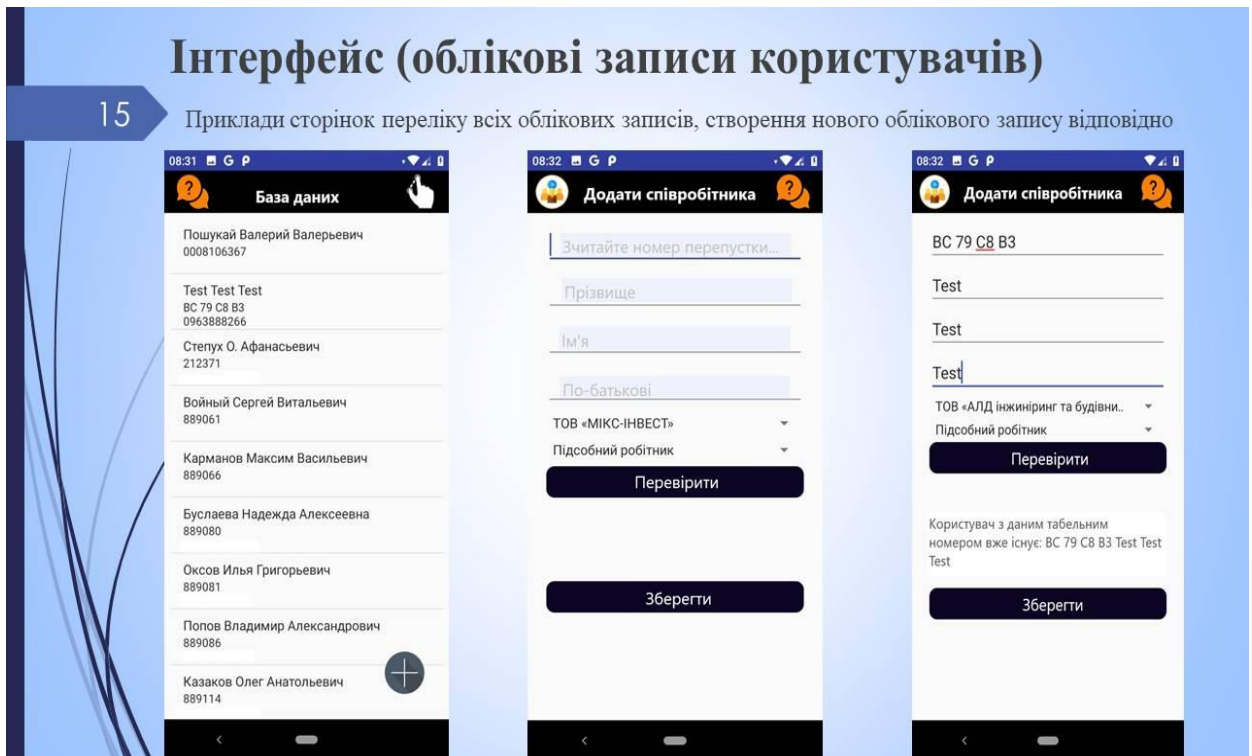


Рисунок Б.15 – Слайд 15

Чек-лист тестування

16

	Термінал, RFID	Термінал, QR	Телефон, RFID	Телефон, QR
Створення нового облікового запису	+	+	+	+
Перегляд всіх облікових записів	+	+	+	+
Перегляд облікових записів користувачів на зміні	+	+	+	+
Фіксація початку робочої зміни	+	+	+	+
Фіксація завершення робочої зміни	+	+	+	+
Передача щоденного звіту на сервер	+	+	+	+
Передача щомісячного звіту на сервер	+	+	+	+
Перегляд картки співробітника	+	+	+	+
Перегляд переліку документів співробітника	+	+	+	+
Перегляд дозвільних документів співробітника	+	+	+	+

Рисунок Б.16 – Слайд 16

Порівняння реалізацій систем фіксації робочого часу

17

Підстава для порівняння	СКУД	Система трекінгу за допомогою «розумних годинників»	«Trackers»
Відносна собівартість	висока	висока	середня
Мобільність системи	статична	мобільна	мобільна
Залежність від погодних умов	незалежна	залежна	незалежна
Достовірність отриманої інформації	умовно достовірна	умовно достовірна	достовірна
Залежність від висот місцевості	незалежна	дуже залежна	незалежна
Поріг навичок для впровадження	середній	високий	середній
Відповідність необхідних технічних засобів наявним	низька	низька	висока
Відповідність необхідних індивідуальних засобів для трекінгу наявним	середня	низька	висока

Рисунок Б.17 – Слайд 17

ВИСНОВКИ

18

- Створений проєкт відповідає меті розробки, всім вимогам, що були пред'явлені до програмного засобу.
- Програмний продукт пройшов всі тести: для тестування було обрано основні функції користувача, тести проведено на цільових пристроях та з різними технологіями зчитування перепустки.
- Передбачено зручний інтерфейс користувача, передбачено введення помилкових даних та невірних дій користувача.
- Новизна полягає у портативному нестатичному використанні RFID та QR технологій сканування для мобільних телефонів та промислових терміналів.
- За темою роботи є 1 наукова публікація, а саме тези доповіді на науково-практичній конференції «Сучасні проблеми і досягнення в галузі радіотехніки, телекомунікацій та інформаційних технологій», яка дистанційно проводилася у Національному університеті «Запорізька політехніка» у грудні 2022 року.

Шляхи майбутнього удосконалення:

- Авторизація нормувальника.
- Доступ до даних усіх пристроїв, що використовують застосунок.
- Вдосконалення перегляду документів.
- Фіксація місцезнаходження пристрою на момент фіксування робочого часу.

Рисунок Б.18 – Слайд 18