

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Інститут інформатики та радіoeлектроніки,
Факультет комп'ютерних наук і технологій

(повне найменування інституту, назва факультету)

Кафедра програмних засобів

(повне найменування кафедри)

Пояснювальна записка

до дипломного проєкту (роботи)

магістра

(ступінь вищої освіти)

на тему ДОСЛІДЖЕННЯ ТА ПРОГРАМНА РЕАЛІЗАЦІЯ МЕТОДІВ
ДИНАМІЧНОГО ВИЗНАЧЕННЯ КІЛЬКОСТІ УКЛАДЕНИХ ДЕКЛАРАЦІЙ З
ЛІКАРЕМ

RESEARCH AND SOFTWARE IMPLEMENTATION OF METHODS TO
DYNAMICALLY FIND OUT THE NUMBER OF DECLARATIONS SIGNED
WITH A DOCTOR

Виконав: студент(ка) 2 курсу, групи КНТ-219м
Спеціальності 122 Комп'ютерні науки

(код і найменування спеціальності)

Освітня програма (спеціалізація)

Системи штучного інтелекту

Федічкін І.О.

(прізвище та ініціали)

Керівник Льовкін В.М.

(прізвище та ініціали)

Рецензент Зеленьова І.Я.

(прізвище та ініціали)

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

Інститут, факультет ІРЕ, ФКНТ
 Кафедра програмних засобів
 Ступінь вищої освіти магістр
 Спеціальність 122 Комп'ютерні науки
 (код і найменування)

Освітня програма (спеціалізація) Системи штучного інтелекту
 (назва освітньої програми (спеціалізації))

ЗАТВЕРДЖУЮ

Завідувач кафедри ПЗ, д.т.н, проф.
С.О. Субботін
 “ ” 2020 року

З А В Д А Н Н Я

НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА(КИ)

Федічкіна Івана Олександровича
 (прізвище, ім'я, по батькові)

1. Тема проєкту (роботи) Дослідження та програмна реалізація методів динамічного визначення кількості укладених декларацій з лікарем
Research and Software Implementation of Methods to Dynamically Find out the Number of Declarations Signed with a Doctor

керівник проєкту (роботи) Льовкін Валерій Миколайович, к.т.н., доцент,
 (прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом закладу вищої освіти від “3” листопада 2020 року № 301

2. Строк подання студентом проєкту (роботи) грудень 2020 року

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

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1. Аналіз проблеми організації медичної програмної системи. 2. Матеріали і методи. 3. Проєктування програмного забезпечення. 4. Основні рішення щодо реалізації компонентів системи. 5. Експлуатація та тестування програми. 6. Економіко-організаційна частина, 7. Охорона праці та безпека в надзвичайних ситуаціях.

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	Льовкін В.М. доцент		
Економіко-організаційна частина	Пожуєва Т.О., професор		
Охорона праці та безпека в надзвичайних ситуаціях	Коробко О.В., ст. викладач		
Нормоконтролер	Дейнега Л.Ю., ст. викладач		

7. Дата видачі завдання “ 7 ” вересня 2020 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1	Аналіз та дослідження предметної області	1-2 тиждень	
2	Розробка та затвердження технічного завдання	3 тиждень	Технічне завдання
3	Дослідження методів розв’язання проблеми	4 тиждень	
4	Проектування структури програмного забезпечення	5 тиждень	Структурна схема
5	Розроблення схеми функціонування програмного забезпечення та модулів	6-7 тиждень	Функціональна схема. Модулі
6	Розроблення програмного коду	8 тиждень	Текст програми
7	Тестування та відлагодження програмного забезпечення	9 тиждень	Програма
8	Розробка розділів пояснювальної записки	10-11 тиждень	Пояснювальна записка
9	Розробка графічного матеріалу	12 тиждень	Графічний матеріал
10	Захист магістерської роботи	13 тиждень	

Студент(ка)

(підпис)

Федічкін І.О.

(прізвище та ініціали)

Керівник проєкту (роботи)

(підпис)

Льовкін В.М.

(прізвище та ініціали)

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи магістра:
114 с., 27 рис., 6 табл., 3 дод., 30 джерел.

МЕДИЧНА СИСТЕМА, ЕЛЕКТРОННА МЕДИЧНА КАРТКА,
ДЕКЛАРАЦІЇ, ПРОГНОЗУВАННЯ ЧАСОВОГО РЯДУ, LONG SHORT
TERM MEMORY, C#.

Об'єкт дослідження – процес організації медичної системи. Предмет дослідження – методи динамічного визначення кількості укладених декларацій з лікарем.

Мета роботи – розроблення медичної програмної системи з можливостями динамічного визначення кількості укладених декларацій з лікарем для забезпечення обґрунтованості прийняття рішень відносно планування роботи медичного закладу.

Матеріали, методи та технічні засоби: об'єктно-орієнтоване програмування, мови програмування C# і Python, бібліотека Keras.

Результати. Проаналізовано проблему організації медичної системи.

Наукова новизна роботи полягає в запропонованому методі динамічного визначення кількості укладених декларацій з лікарем, що характеризується використанням нейронної мережі архітектури LSTM для прогнозування кількості декларацій на майбутню послідовність періодів часу, що дозволяє менеджменту лікарні приймати рішення про планування своєї діяльності.

Висновки. Програмно реалізовано медичну систему, що дозволяє виконувати оформлення декларацій з лікарем, отримувати лікарям та пацієнтам єдиний доступ до медичної інформації, вести електронні медичні картки.

Галузь використання – організація електронних медичних карток.

ABSTRACT

Explanatory note to the diploma qualifying work of the master: 114 pages, 27 figures, 6 tables, 3 appendixes, 30 sources.

The object of research is the process of organizing the medical system. The subject of research is methods of dynamic determination of the number of concluded declarations with the doctor.

The purpose of the work is to develop a medical software system with the ability to dynamically determine the number of declarations concluded with a doctor to ensure the validity of decisions regarding the planning of the medical institution activity.

Materials, methods and technical means: object-oriented programming, C# and Python programming languages, Keras library.

Results. The problem of organization of medical system is analyzed.

The scientific novelty of the work lies in the proposed method of dynamic determination of the number of concluded declarations with the doctor, characterized by the usage of long short term memory as neural network architecture to predict the number of declarations for future time periods, which allows hospital management to decide on planning its activities.

Conclusions. The software implements a medical system which allows users to make declarations with a doctor, provide doctors and patients with a single access to medical information, keep electronic medical records.

Field of usage is organization of electronic medical records.

ЗМІСТ

	С.
Перелік скорочень та умовних познач.....	8
Вступ.....	9
1 Аналіз проблеми організації медичної програмної системи.....	11
1.1 Системи управління охороною здоров'я.....	11
1.2 Огляд існуючих медичних програмних систем.....	15
1.3 Висновки за розділом 1.....	17
2 Матеріали і методи.....	18
2.1 Задача динамічного визначення кількості укладених декларацій з лікарем.....	18
2.2 Архітектура нейронних мереж LSTM.....	20
2.3 Розроблений метод динамічного визначення кількості укладених декларацій з лікарем.....	23
2.4 Висновки за розділом 2.....	25
3 Проєктування програмного забезпечення.....	26
3.1 Вибір засобів реалізації програмного забезпечення.....	26
3.2 Функціональні вимоги до програмного забезпечення.....	27
3.3 Проєктування бази даних.....	30
3.4 Структура програмного проєкту.....	35
3.5 Висновки за розділом 3.....	36
4 Основні рішення щодо реалізації компонентів системи.....	37
4.1 Алгоритм функціонування системи.....	37
4.2 Опис модулів реалізованої системи.....	39
4.3 Висновки за розділом 4.....	42
5 Експлуатація та тестування програми.....	43
5.1 Призначення програми.....	43
5.2 Умови виконання програми.....	43
5.3 Виконання програми.....	44
6 Економіко-організаційна частина.....	51

6.1	Планування розробки програмного продукту.....	51
6.2	Розрахунок основної заробітної плати.....	53
6.3	Розрахунок додаткової заробітної плати	54
6.4	Єдиний соціальний внесок.....	54
6.5	Визначення витрат на матеріали	54
6.6	Витрати на спеціальне обладнання	55
6.7	Розрахунок накладних витрат.....	59
6.8	Розрахунок економічної ефективності програмного продукту.....	59
7	Охорона праці та безпека в надзвичайних ситуаціях.....	62
7.1	Аналіз потенційних небезпек.....	62
7.2	Заходи по забезпеченню безпеки	62
7.3	Заходи по забезпеченню виробничої санітарії та гігієни праці	64
7.4	Заходи по забезпеченню пожежної безпеки.....	68
7.5	Заходи безпеки у надзвичайних ситуаціях	70
	Висновки	72
	Перелік джерел посилання	74
	Додаток А Технічне завдання	78
	Додаток Б Текст програми	83
	Додаток В Слайди презентації.....	107

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

LSTM – Long Short Term Memory;

ВДТ – візуальний дисплейний термінал;

ЕОМ – електронно-обчислювальна машина;

ПК – персональний комп'ютер.

ВСТУП

Медична сфера є однією з найпотужніших економічно сфер. Будь-які зміни, які відбуваються в цій сфері призводять до великого впливу на людське життя. Тому будь-які, навіть маленькі зміни, є важливим. Впровадження програмних інструментів у роботу даної сфери є дуже важливим, але і складним процесом.

У відповідності з реформою сфери охорони здоров'я, яка впроваджується в Україні, пацієнти мають заключати декларації з сімейними лікарями на виконання медичних послуг.

На даний момент існують програмні продукти, які дозволяють заключати такі декларації з лікарями по своїй країні, але при цьому необхідно розуміти, що кількість декларацій для кожного лікаря – це показник ефективності його роботи для нього самого та для медичного закладу, в якому він працює.

Якщо медичний заклад загалом і лікар зокрема будуть мати механізм прогнозування кількості декларацій, які з лікарем буде підписано в майбутньому, то це дозволить ефективніше планувати власну діяльність як лікарям, так і медичним закладам.

Тому проблему організації медичної програмної системи загалом та зокрема проблему прогнозування кількості декларацій, які будуть укладені з лікарем, можна вважати актуальною. У даній роботі буде розглянуто створення медичної програмної системи, яка має окрім базових функцій забезпечувати прогнозування кількості декларацій, які будуть укладені з лікарем.

Об'єкт дослідження – процес організації медичної системи.

Предмет дослідження – методи динамічного визначення кількості укладених декларацій з лікарем.

В якості мети дипломної кваліфікаційної роботи в результаті було обрано розроблення медичної програмної системи з можливостями

динамічного визначення кількості укладених декларацій з лікарем для забезпечення обґрунтованості прийняття рішень відносно планування роботи медичного закладу.

1 АНАЛІЗ ПРОБЛЕМИ ОРГАНІЗАЦІЇ МЕДИЧНОЇ ПРОГРАМНОЇ СИСТЕМИ

1.1 Системи управління охороною здоров'я

Дослідження університету в Нью-Мехіко показало, що проєктування клінічних процесів та клінічна культура, які безпосередньо пов'язані з вивчаємим питанням – застосуванням електронних медичних карток пацієнтів, сприяють майже 40 відсоткам загального стресу, який зазнають медичні працівники, зокрема наступні складності електронних медичних карток призводять до вигорання лікаря:

- трудомісткі операції: у дослідженні університету в Нью-Мехіко вказується, що лікарі витрачають одну-дві години на роботу над медичними картками на кожну годину, проведену з пацієнтами, і щодня витрачають додатково одну-дві години особистого часу на операції, пов'язані з такими системами, при цьому планування комп'ютерної томографії за допомогою таких систем може зайняти 45 хвилин, більше того, такі системи, навіть якщо вони надаються одним і тим же постачальником, не є взаємодіючими між різними установами, отже, тести пацієнта можуть повторюватися стільки разів, скільки пацієнт переходить до іншого закладу;

- програмні системи електронних медичних карток пацієнтів схильні до помилок: проблема помилкової ідентифікації пацієнта стає все більш очевидною (минулого року в лікарні в Нью-Джерсі стався випадок, коли пацієнт отримав трансплантацію нирки, призначеної для іншого пацієнта з таким самим ім'ям та віком, на щастя, призначений реципієнт отримав трансплантацію через декілька днів, і обидва пацієнти в порядку, але багато подібних випадків було спричинено використанням систем підтримки електронних медичних карток пацієнтів);

- системи підтримки електронних медичних карток пацієнтів важкі у використанні: дані системи є складними та функціональними, налаштування системи вимагає великої кількості робіт вручну і може бути не повністю

успішним через жорсткість систем, необхідність бути технічно підкованими витісняє деяких лікарів з професії, оскільки помилка може коштувати життя, крім того дані системи постійно вимагають надмірних дій, щоб вставити інформацію під час зустрічей, відволікаючи увагу від пацієнтів [1]-[3].

Системи управління охороною здоров'я, також відомі як системи управління інформацією про охорону здоров'я, розроблені, щоб допомогти працівникам охорони здоров'я ефективніше збирати, зберігати, отримувати та обмінюватися інформацією про охорону здоров'я пацієнтів та забезпечувати кращий догляд за пацієнтами, ці технологічні проєкти, які часто керуються дотриманням вимог, можуть вимагати значних інвестицій у час та ресурси, при виборі та впровадженні системи управління охороною здоров'я постачальники повинні мати на увазі три фактори:

- існують два основні типи систем управління охороною здоров'я – електронні медичні картки та управління на основі практики, які організації повинні оцінити залежно від їх розміру, потреб бізнесу та дотримання вимог, програмні рішення для управління на основі практики, які можна підключити до систем електронних медичних карток, зазвичай використовуються малими та середніми медичними практиками для організації конкретних деталей про пацієнтів та допомогу, яку вони отримують від практики;

- більшість систем управління охороною здоров'я, особливо для великих постачальників медичних послуг, постачаються як рішення типу «програмне забезпечення як послуга», оскільки програмне забезпечення постачається на замовлення через хмарні сховища, а не через внутрішні системи, постачальники медичних послуг повинні особливо уважно ставитися до збереження конфіденційності та безпеки даних (інформація про охорону здоров'я пацієнта), отже, вибираючи постачальника для постачання своєї системи управління охороною здоров'я, постачальники повинні поставити та отримати чіткі відповіді на ряд питань (як ви захищаєте дані? де зберігаються дані? як ваша практика відповідає вимогам дотримання безпеки даних у відповідності з законами);

– система управління охороною здоров'я – це набагато більше, ніж IT-проект, впровадження нової технологічної інфраструктури та оцифрування клінічної інформації та інформації про пацієнтів – лише частини процесу переходу до системи управління охороною здоров'я, медичним працівникам, які будуть користуватися системою щодня, – лікарям, медсестрам, технічним працівникам, медичним адміністраторам та багатьом іншим – потрібно залучатись на самому початку процесу, щоб вони могли сказати, як розроблена система, пройти відповідну підготовку та мати достатній час підготуватися до того, що буде монументальним зрушенням у їх роботі, це вимагає від постачальників зосередитись на управлінні змінами з самого початку проекту системи управління охороною здоров'я, багато постачальників вирішують звернутися за сторонньою допомогою, щоб забезпечити плавний перехід до системи управління охороною здоров'я, наприклад, вони звертаються до служб, які можуть надати консультантам знання з управління охороною здоров'я та бізнес-систем, вони також можуть звернутися до експертів з питань відповідності для отримання вказівок щодо впровадження систем, які відповідають вимогам відповідних нормативних актів [4].

Хоча лікарі можуть отримати певну інформацію безпосередньо з різних джерел, вони все одно повинні вводити або завантажувати дані до електронної медичної картки, відповідно під час даного процесу можуть траплятися людські помилки, іноді такі системи навіть можуть сприяти появі помилок (наприклад, деякі системи видаляють десяткову крапку, ця помилка може призвести до передозування, тож розробники повинні уникати таких невідповідностей), для цього система може мінімізувати можливість помилок наступним чином:

– введення даних: можуть бути випадки, коли пацієнти вважать у фунтах, але дані про пацієнта повинні бути введені в систему у кілограмах, для забезпечення точності одиниці повинні бути чітко визначені в інтерфейсі,

загальним принципом процесу введення даних є простота та послідовність даних;

- точне редагування: копіювання та вставка – важлива функція, яка скорочує час та когнітивне навантаження для медичних працівників, однак погано реалізована функція копіювання-вставлення може становити ризик для пацієнтів, користувачі можуть не перевіряти та редагувати дані, оскільки система не допомагає їм це робити, щоб дозволити користувачам виявляти неточну інформацію, система може забезпечити характеристики, які описано далі;

- автоматичне оновлення даних: система може автоматично оновлювати примітки з поточними життєвими показниками та результатами тестів, відображення імені автора у всіх примітках може допомогти лікарям зрозуміти, чи оновлено нещодавню нотатку з попередньої версії, ім'я автора може залишатися незмінним, якщо примітка копіюється без змін, але змінюється, якщо інший лікар переглядає та редагує її;

- централізована система узгодження даних: централізована система зіставлення даних автоматизує дедуплікацію записів та цілісність даних, ефективний головний індекс корпоративного пацієнта необхідний для забезпечення безперебійного потоку даних від постачальника до постачальника, індекси, які використовують інформацію про місцезнаходження та геокодування для перевірки адрес, допомагають медичним працівникам стандартизувати та автентифікувати адреси в режимі реального часу, це допомагає уникнути ризиків, пов'язаних з дублікатами записів, та виявити шахрайство [1], [5]-[6].

Гарантувати найвищий рівень зручності для медичного персоналу та пацієнтів можна наступним чином:

- мінімізувати кількість необхідних кліків: більшість систем такого типу вимагають від користувачів надмірних кліків;

- розглядати лікарів головним чином як людей: необхідно знайти людський тон голосу, тому якщо немає справжньої надзвичайної ситуації,

зробіть кольори та символи інтерфейсу нейтральними, навіть спокійними, періодичне заохочення повідомленнями на кшталт «дякую за всю вашу важку працю» буде надихати та підтримувати користувачів [1], [5]-[6].

1.2 Огляд існуючих медичних програмних систем

Одним з найбільш активно використовуваних медичних сервісів в Україні є сервіс Helsi.me (рис. 1.1), що дозволяє автоматизовувати роботу лікаря і реєстратури медичного закладу.

Сервіс Helsi.me надає можливість виконання наступних функцій:

- укладати декларацію з лікарем;
- виконувати пошук лікарів;
- переглядати лікарів за лікарнями;
- записуватися на прийом до лікарів;
- отримувати доступ до медичної картки пацієнтом;
- вносити дані в медичні картки пацієнтів лікарям;
- автоматизовувати діяльність медичного закладу;
- отримувати результати аналізів;
- формувати звіти;
- налаштовувати права доступу до програми;
- знаходити потрібні ліки.

Одним з найвідоміших програмних продуктів з цієї групи в США є сервіс Cerner.

Електронні медичні записи дають можливість автоматизувати клінічні операції в процесі виконання медичної практики, такі як документування діагнозів та ліків [8], для цього Cerner має рішення, яке підходить для будь-якого типу медичних працівників, при цьому для кожного такого варіанту створено окремий програмний продукт, що дуже важливо.

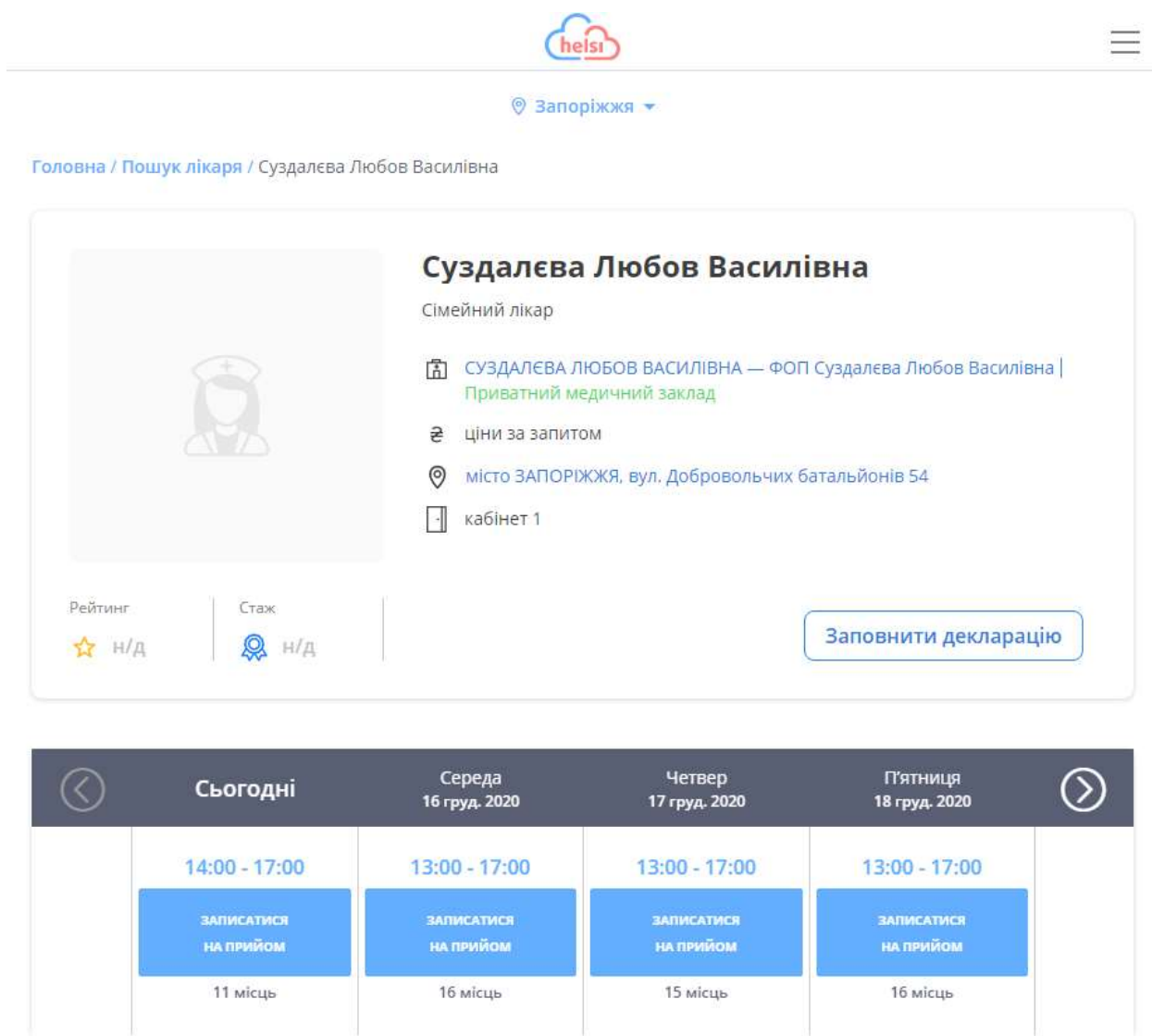


Рисунок 1.1 – Сервіс Helsi.me

Розрізняють наступні програмні продукти:

– Cerner PowerChart – це рішення, яке підходить для закладів життєзабезпечення, центрів довготривалого догляду, пенсійних громад та спеціальних відділень по догляду всіх розмірів, розгортається на основі хмари;

– Cerner Millennium – це рішення, яке ідеально підходить для постачальників медичних послуг для підприємств, що обслуговують різноманітні спеціалізації, такі як великі лікарні або провайдери поведінкових послуг, надається документація щодо медичного

адміністрування, звітність про аудит та попередження про лабораторні дослідження;

– Cerner practice Management – рішення, яке детально підходить для практик малого та середнього бізнесу з 10 або менше постачальниками, заслуговують на увагу функції перевірки страхових відшкодувань у режимі реального часу, проводки платежів, планування роботи з пацієнтами та підтримки виставлення рахунків швидкої допомоги, варіанти розгортання включають варіанти на базі хмари або локально [8].

1.3 Висновки за розділом 1

Проаналізовано проблему організації медичної системи, для цього виділено основні вимоги та особливості взаємодії з користувачами, якими характеризуються системи керування охороною здоров'я.

Розглянуто програмні аналоги для програмного забезпечення, що розробляється.

Проаналізовано функціональні можливості українського сервісу Helsi.me, що виконує подібні дії в даній предметній області.

В якості завдань дипломної кваліфікаційної роботи магістра виділено наступні:

- розроблення методу динамічного визначення кількості укладених декларацій з лікарем;
- розробка технічного завдання;
- проєктування програмного забезпечення;
- розробка програмного забезпечення організації медичного сервісу;
- тестування та відлагодження програми;
- економічне обґрунтування доцільності розробки.

2 МАТЕРІАЛИ І МЕТОДИ

Програмне забезпечення, що розробляється, має накопичувати дані щодо стану здоров'я пацієнта, основою для чого є його медична картка. Для забезпечення ведення медичної картки пацієнти фіксують свої відносини з лікарями шляхом укладання декларації.

Декларації укладаються не тільки з сімейним лікарем, а і з будь-яким іншим лікарем, відповідно тільки за наявності декларації лікар може працювати з медичною картою пацієнта: переглядати її дані та вносити нові дані.

2.1 Задача динамічного визначення кількості укладених декларацій з лікарем

Програмне забезпечення організації медичного сервісу організує підтримку цілого ринку медичних послуг. Аналогічно будь-яким ринковим відносинам, одним із головних показників щодо діяльності лікаря є кількість декларацій, яку з ним було укладено. При цьому важливо розуміти, що ці дані не статичні, тобто вони змінюються з часом. А раз вони змінюються, то потрібно мати механізм прогнозування кількості декларацій з конкретним лікарем. Зрозуміло, що кількість декларацій повинна залежати від того, як виконує свою роботу лікар. Але це не завжди повністю справедливо. Недосвідчений лікар не може розраховувати на велику кількість декларацій одразу, навіть якщо він має вже всі знання, потрібні для лікування, зараз. Відповідно слід говорити про те, що задача динамічного визначення кількості укладених декларацій з лікарем, що полягає в постійному оновленні накопичених даних та виконанні прогнозування на майбутній період і подальшій актуалізації отриманих результатів, є задачею прогнозування часового ряду.

Задача динамічного визначення кількості укладених декларацій з лікарем полягає в прогнозуванні кількості укладених декларацій з заданим лікарем на наступний період (або наступні k періодів) за значеннями попередніх n періодів часу тієї ж тривалістю, що формують послідовність відповідних значень кількості укладених декларацій всіх пацієнтів з заданим лікарем за ці проміжки часу.

Для цього має бути визначено модель, яка буде виконувати моделювання цієї залежності виходів від входів і дозволити обчислювати значення на виході за заданими значеннями входів.

Під час процесу навчання мережі основною метою є мінімізація збитків (з точки зору помилок або витрат), що спостерігаються на виході, коли навчальні дані надсилаються через неї, для цього обчислюється градієнт, тобто втрати щодо певного набору ваг, регулюються ваги відповідно і повторюється цей процес, поки не отримається оптимальний набір ваг, для яких втрати мінімальні, це називається концепцією зворотного відстеження, іноді трапляється так, що градієнт майже незначний, слід зазначити, що градієнт шару залежить від певних компонентів у наступних шарах, якщо деякі з цих компонентів малі (менше 1), отриманий результат, який є градієнтом, буде ще меншим, це називається ефектом масштабування, відповідно коли цей градієнт помножується на швидкість навчання, яка сама по собі є малим значенням в діапазоні 0,1-0,001, це призводить до меншого значення, як наслідок, зміни вагових показників є досить незначними, що дає майже такий же вихід, як і раніше, подібним чином, якщо градієнти досить великі за величиною через великі значення компонентів, ваги оновлюються до значення, що перевищує оптимальне значення, ця ситуація відома як проблема зникаючого градієнта, а щоб уникнути цього ефекту масштабування, блок нейронної мережі був перебудований таким чином, що коефіцієнт масштабування був зафіксований до одиниці, потім комірку збагатили декількома воротами і отримана архітектура нейронної мережі є архітектурою Long Short Term Memory (LSTM) [9]-[11].

Одночасно потрібно враховувати, що в залежності від обраного базису для прогнозування – часового періоду (місяць, квартал або рік) – може бути різною кількість накопичених даних, відповідно кількість таких часових періодів. Але зокрема для короткострокового прогнозування кількість місяців, для яких вже було накопичено історію роботи лікаря з пацієнтами, є достатньо великою. Адже не зважаючи на те, що в Україні така система тільки почала з'являтися, в багатьох світових країнах історія такого підходу налічує десятки років. Відповідно можна говорити про те, що в задачі динамічного визначення кількості укладених декларацій з лікарем буде моделюватися довгострокова залежність, адже період, за яким відбувається прогнозування в такому випадку може ставати достатньо великим. Враховуючи цей фактор, можна говорити про те, що рекурентні нейронні мережі і зокрема архітектура LSTM є актуальним засобом для розв'язання задачі.

2.2 Архітектура нейронних мереж LSTM

Модель LSTM – це потужна рекурентна нейронна мережа, спеціально розроблена для подолання проблем зникаючого градієнта, що зазвичай виникає при навчанні довготривалим залежностям, загалом цього можна запобігти, використовуючи постійну карусель помилок, яка підтримує сигнал помилки в комірці кожного блоку [12].

Ванільний блок LSTM складається з комірки, вхідного затвора, вихідного затвора та затвора забуття: затвор забуття спочатку не був частиною мережі LSTM, але був запропонований потім, щоб дозволити мережі скинути стан: комірка запам'ятовує значення через довільні інтервали часу, і три ворота регулюють потік інформації, пов'язаний з коміркою, при цьому архітектура LSTM складається з набору періодично підключених підмереж, відомих як блоки пам'яті, ідея блоку пам'яті полягає в тому, щоб

підтримувати його стан з часом і регулювати інформаційний потік нелінійних блоків управління [13].

До основних проблем, які виникають під час використання нейронних мереж для розв'язання проблеми прогнозування часового ряду, належать:

- відсутність у літературі систематизованого способу визначення відповідного набору входів до нейронної мережі;
- немає загального правила з розподілу набору спостережень під час навчання, перевірки та прогнозування таким чином, щоб гарантувати оптимальні результати;
- немає чітких вказівок, які дозволяють апріорі виражати, яку передавальну функцію слід використовувати в моделі нейронної мережі відповідно до характеристик часових рядів;
- немає ясності щодо процедур, орієнтованих на відбір нейронів у прихованому шарі, що, в свою чергу, дозволяє мінімізувати час навчання мережі;
- немає емпіричних, методологічних чи теоретичних причин, щоб віддати перевагу конкретній моделі серед кількох альтернатив;
- немає згоди щодо того, як вибрати остаточну модель, коли розглядається кілька альтернатив;
- не зрозуміло, коли і як трансформувати дані перед виконанням моделювання;
- немає ясності щодо необхідності усунення чи не усунення тенденції та сезонних компонентів у моделях нейронних мереж;
- важко включити якісну, суб'єктивну та контекстуальну інформацію до прогнозів;
- існує мало розуміння статистичних властивостей різних архітектур нейронних мереж;
- немає ясності щодо того, які процедури є найбільш адекватними процедурами для оцінки, перевірки та тестування різних архітектур нейронних мереж;

- немає ясності щодо того, як поєднувати прогнози з декількох альтернативних моделей, і чи є вигоди від цієї практики;

- у критеріях оцінки продуктивності різних архітектур нейронних мереж немає чіткості;

- немає ясності щодо того, за якими критеріями різні архітектури нейронних мереж дозволяють обробляти динамічну поведінку в даних [14].

Незважаючи на таку велику кількість проблем, нейронні мережі можна налаштувати евристичним чином на практиці і при цьому вони дозволяють отримати результат прогнозування, що є значно кращим за існуючі традиційні підходи, що зокрема охоплюють лінійну регресію, метод опорних векторів, алгоритму випадкового лісу тощо.

Загалом у великій кількості робіт, направлених на розв'язання задач прогнозування в різних практичних сферах, визначається, що саме нейронні мережі на основі архітектури LSTM є на даний момент найактуальнішим засобом роботи з часовими рядами.

Кількість декларацій, яку було укладено лікарем в теперішній проміжок часу (зазвичай приймається рівним місяцю), залежить від багатьох факторів:

- тенденції, якої дотримувалися пацієнти, підписуючи декларації кожен місяць, в попередній проміжок часу, при цьому можливий спадний чи висхідний тренд;

- кількість декларацій, яку було укладено з лікарем протягом попереднього часового проміжку (місяця), оскільки багато пацієнтів порівнюють кількість декларацій для даного лікаря за попередній часовий проміжок перед покупкою;

- фактори, які можуть вплинути на кількість укладених декларацій на сьогодні: це може бути нова політика компанії, яка широко критикується, або падіння прибутку компанії, або, можливо, несподівана зміна вищого керівництва компанії [15].

В результаті саме нейронні мережі архітектури LSTM було обрано для забезпечення прогнозування кількості декларацій, які було укладено з лікарем.

2.3 Розроблений метод динамічного визначення кількості укладених декларацій з лікарем

У даному підрозділі має бути сформульовано поетапний метод динамічного визначення кількості укладених декларацій з лікарем, що ґрунтується на розв'язанні задачі динамічного визначення кількості укладених декларацій з лікарем на основі рішень, які було описано в даному розділі.

Робота методу ґрунтується на моделі прогнозування кількості укладених декларацій з лікарем на основі LSTM-мережі. Для її навчання мають бути підготовлені дані.

Етап 1. Витягання даних з бази даних, акумуляція статистики щодо кожного лікаря в системі і формування вибірки даних.

Для цього має бути задано часовий горизонт для формування вибірки. Часовий горизонт охоплює мінімальну дату, з якої накопичується статистика, і триває до поточної дати. При цьому має бути задано базовий часовий період. В якості такого може використовуватися місяць. Відповідно кількість повних місяців – тривалість часового горизонту.

Кожен екземпляр вибірки даних представлений сумарною кількістю декларацій d_j^i , які були укладені з заданим лікарем i за заданий базовий період j .

Кожен екземпляр вибірки має складатися з M значень $d_j^i, j = r, \dots, r + M - 1$.

Якщо для якогось лікаря кількість базових періодів, за якою для нього накопичено статистику, є більшою ніж M , то з такої послідовності даних має бути сформовано набір підпослідовностей даних. Перша підпослідовність

має відповідно номери $j = 1, \dots, M$, друга підпоследовність має номери $j = 2, \dots, M + 1$, третя – $j = 3, \dots, M + 2$ тощо. Таким чином на основі даних кожного лікаря може бути накопичено цілий набір екземплярів вибірки.

Кожен екземпляр вибірки складається зі входних ознак, представлених відповідно значеннями $d_j^i, j = r, \dots, r + M - 1$ та вихідної ознаки, представленої значенням d_{r+M}^i , тобто кількістю декларацій, яку укладено з даним лікарем в наступний базовий період (місяць), або значеннями $d_{r+M}^i, d_{r+M+1}^i, \dots, d_{r+M+P}^i$.

Етап 2. Створити модель прогнозування кількості укладених декларацій з лікарем на основі використання нейронної мережі з архітектурою LSTM. Конкретні налаштування такої моделі мають бути встановлені на практиці, враховуючи особливості, які було перелічено в даному розділі, відповідно параметри залежать також і від вибірки даних, зокрема і її обсягу.

На практиці використовувалась модель з двома прихованими LSTM шарами і одним повнозв'язним, результати обчислення якого подавалися на вихідний шар. Кількість нейронів входного шару дорівнювала 24, а кількість нейронів вихідного шару дорівнювала 3. Тобто прогнозування здійснювалось на основі даних про 2 попередні роки роботи лікаря і включало дані про кількість декларацій про 24 послідовні місяці. В той же час прогнозування реалізовувалось на 3 наступні місяці, тобто модель повинна дозволити на основі 24 попередніх місяців отримати прогноз на наступні 3 місяці.

Етап 3. Виконати навчання моделі.

Вся навчальна вибірка, отримана в результаті виконання першого етапу, має бути розбита на навчальну і тестову вибірки. Навчальна вибірка складає 80 %.

Етап 4. Виконати тестування моделі.

На даному етапі має бути прийняте рішення, чи отримана в результаті модель є практично застосовною. Якщо так, то циклічно може здійснюватися

наступний етап, п'ятий. Якщо ні, то має вирішуватися питання перенавчання можливого моделі.

Етап 5. Виконати оновлення кількості очікуваних для укладення декларацій з лікарем на наступні базові періоди (у даному випадку на наступні 3 місяці) на основі поданих значень про попередні періоди (24 місяця).

Відповідно для цього реалізується переобчислення кількості декларацій, укладених за місяць. Таким числом вважається саме активна кількість декларацій. Тобто якщо пацієнт прийняв рішення змінити лікаря і уклав нову декларацію, то в статистиці за місяць має бути враховано тільки його нову декларацію. Для будь-якого пацієнта декларація за базовий період має враховуватися тільки 1 раз.

За поданою на вхід моделі кількістю декларацій за 24 місяці відбувається прогнозування кількості укладених декларацій в наступні 3 місяці.

Отримані дані мають зберігатися в базу даних. В таблиці при цьому робиться позначка, що дане значення є прогнозом.

2.4 Висновки за розділом 2

Проаналізовано задачу динамічного визначення кількості укладених декларацій з лікарем. Обґрунтовано вибір LSTM-мереж в якості засобу вирішення задачі. Розглянуто особливості LSTM-мереж і проблеми, якими характеризується їх використання.

Наукова новизна роботи полягає в запропонованому методі динамічного визначення кількості укладених декларацій з лікарем, що характеризується використанням нейронної мережі архітектури LSTM для прогнозування кількості декларацій на майбутню послідовність періодів часу, що дозволяє менеджменту лікарні приймати рішення про планування своєї діяльності.

3 ПРОЄКТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Вибір засобів реалізації програмного забезпечення

Основним засобом реалізації даного програмного забезпечення є мова програмування, якою має бути реалізовано програмну частину, пов'язану зі взаємодією з користувачем та даними. Від вибору даного засобу напряду залежить вибір інших засобів. В якості альтернативних варіантів розглядались мови програмування C# і C++ [16]-[17].

У табл. 3.1 зведено результати порівняння мов програмування.

Таблиця 3.1 – Порівняння мов програмування

Критерій	C#	C++
Тип мови	Компільована в байт-код	Компільована
Парадигма об'єктно-орієнтованого програмування	+	+
Робота з пам'яттю	Автоматично	Вручну
Створення застосунків для настільних систем	Зручне	Вимагає більших витрат порівняно з C#

Саме мову C# було обрано для реалізації програми, оскільки вона дозволяє швидше писати код і дозволяє використовувати до того ж простіші в застосуванні засоби створення інтерфейсу для настільних застосунків.

Даний вибір визначає вибір наступних засобів:

- Microsoft Visual Studio 2019 в якості інтегрованого середовища розробки;

- Microsoft SQL Server 2019 в якості системи керування базами даних.

При цьому програмну реалізацію безпосередньо описаного методу було вирішено виконувати за допомогою мови програмування Python, оскільки метод потребує роботи зі штучним інтелектом загалом і LSTM-мережами зокрема, а для цього найкращим засобом є мова програмування Python. Вона має широкі засоби реалізації штучного інтелекту, спрощує і пришвидшує розробку таких рішень, а також має величезну підтримку такої розробки. Одночасно її можна, але незручно використовувати для розробки настільних систем, оскільки в такому випадку використовуються достатньо особливі і обмежені в можливостях засоби.

3.2 Функціональні вимоги до програмного забезпечення

Програмне забезпечення, що розробляється в роботі, повинно виконувати наступні функції:

- укладання декларації з лікарем;
- перегляд лікарем власних пацієнтів;
- прогнозування кількості укладених декларацій з лікарем на період;
- перегляд кількості фактично укладених декларацій з лікарем за період;
- додавання записів до медичинської карти пацієнта;
- внесення результатів аналізу крові;
- внесення файлів інших аналізів;
- перегляд результатів аналізів пацієнта;
- прив'язування результату аналізу до запису в медичній картці;
- оформлення лікарняного листа;
- перегляд лікарняних листів пацієнта;
- пошук лікарняних листів;
- пошук пацієнтів;
- пошук лікарів;
- внесення даних пацієнта;

- оформлення рецепту на лікування;
- перегляд рецептів, виданих пацієнту;
- перегляд записів медичної картки;
- авторизація пацієнтів та лікарів.

На рис. 3.1 сформовано діаграму прецедентів роботи пацієнта з програмою.

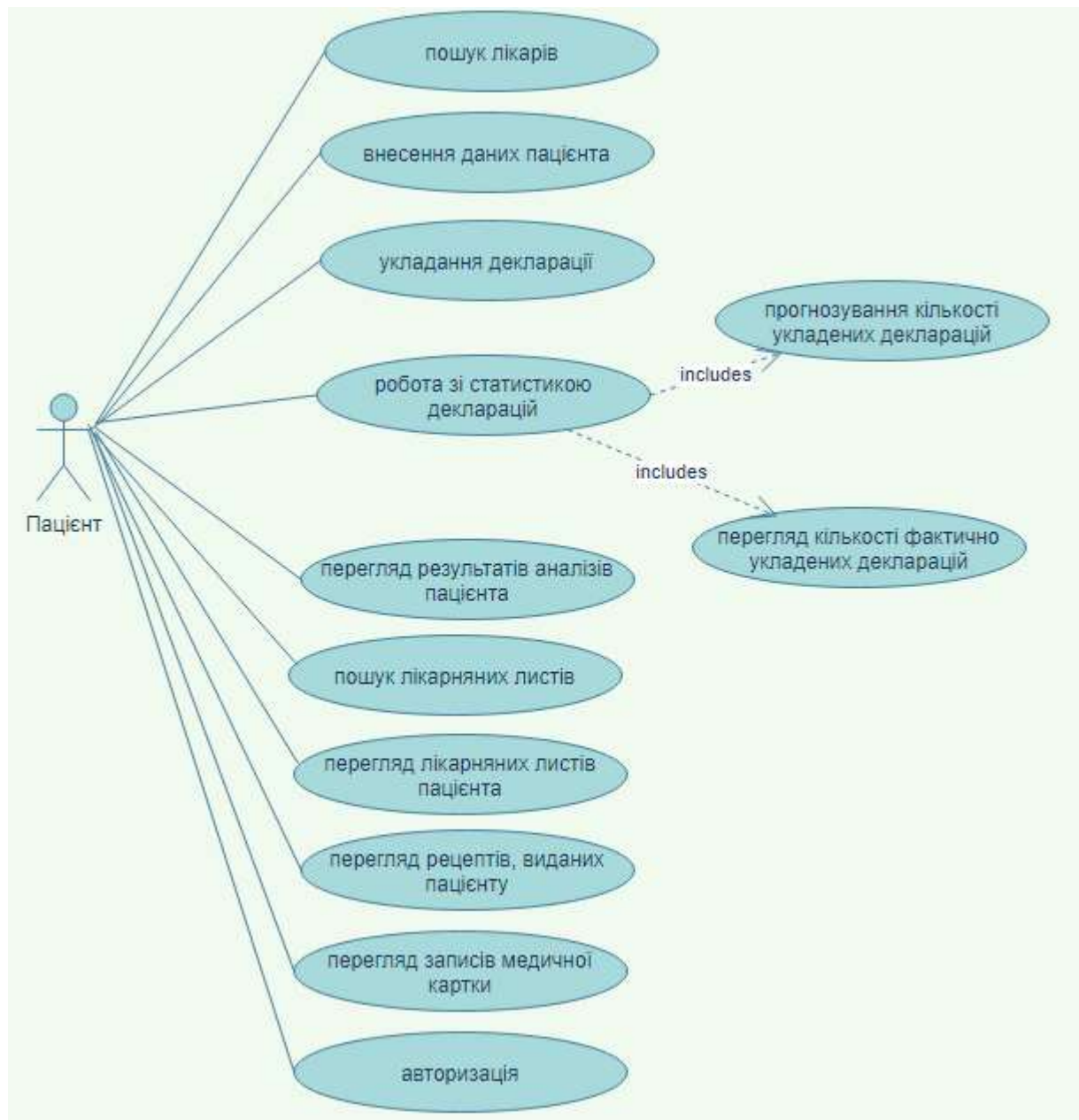


Рисунок 3.1 – Діаграма прецедентів роботи пацієнта з програмою

них призначено програму. Додаткова роль представлена адміністратором, який може додавати акаунти лікарів і пацієнтів.

Пацієнтам надаються засоби роботи зі всіма власними даними, а також засоби, пов'язані з пошуком і відслідковуванням лікарів, – ця група засобів направлена на забезпечення знаходження пацієнтом потрібного йому лікаря та укладання цим пацієнтом декларації з лікарем.

Лікарю доступні окрім засобів роботи з даними окремих пацієнтів засоби перегляду всіх власних пацієнтів.

3.3 Проєктування бази даних

За результатами проєктування бази даних було виділено наступні таблиці:

- Account – акаунт;
- Patient – пацієнт;
- Physician – лікар;
- Card – медична картка;
- Declaration – декларація пацієнта з лікарем, при цьому передбачається, що це декларація з будь-яким лікарем (не обов'язково сімейним);
- Speciality – спеціалізація лікаря;
- Prescription – рецепт;
- BloodAnalysis – аналіз крові;
- DeclarationStatistics – статистика підписування декларацій з лікарем;
- Analysis – аналіз;
- MedCertificate – лікарняний лист.

До складу таблиці Account входять стовпці:

- id – ідентифікатор;
- login – логін;
- password – пароль;

– type – тип.

Поле type введено для того, щоб можна було зберігати акаунти адміністраторів також, а не тільки пацієнтів та лікарів.

До складу таблиці Patient входять стовпці:

- id – ідентифікатор;
- id_account – акаунт;
- surname – прізвище пацієнта;
- name – ім'я;
- fname – по батькові;
- gender – стать;
- birthday – дата народження;
- job – місце роботи;
- nation – національність;
- weight – вага;
- height – зріст;
- idcode – ідентифікаційний номер;
- passport – паспорт;
- phone – номер телефону;
- email – електронна поштова адреса;
- address – домашня адреса.

До складу таблиці Physician входять стовпці:

- id – ідентифікатор;
- id_account – акаунт;
- surname – прізвище пацієнта;
- name – ім'я;
- fname – по батькові;
- id_speciality – спеціалізація;
- department – відділення;
- email – електронна пошта;
- phone – номер телефону.

До складу таблиці Card входять стовпці:

- id – ідентифікатор;
- id_declaration – декларація;
- id_physician – ім'я лікаря;
- infodate – по батькові;
- diagnosis – встановлений діагноз;
- cure – прописане лікування;
- compl – скарги пацієнта.

До складу таблиці Declaration входять стовпці:

- id – ідентифікатор;
- id_patient – пацієнт;
- id_physician – лікар;
- agreemdate – дата укладання;
- type – тип.

До складу таблиці Prescription входять стовпці:

- id – ідентифікатор;
- id_declaration – декларація;
- id_card – медична картка;
- dateinfo – дата виписання рецепту;
- description – текст рецепту.

До складу таблиці BloodAnalysis входять стовпці:

- id – ідентифікатор;
- id_card – запис в медичній картці;
- laboratory – лабораторія;
- dateinfo – дата проведення аналізу;
- eritrocity – показник еритроцитів;
- gemoglobin – показник гемоглобіну;
- leukocity – показник лейкоцитів;
- color – показник кольору;
- gematokrit – показник гематокриту;

- retikulosity – показник ретикулоцитів;
- trombocyt – показник тромбоцитів;
- soe – показник сое;
- basofily – показник базофілів;
- eozinofily – показник еозинофілів;
- meilocyt – показник мієлоцитів;
- metamielocyt – показник метамієлоцитів;
- neutrofilyp – показник нейтрофілів паличкоядерних;
- neutrofilys – показник нейтрофілів сегментоядерних;
- limfocyt – показник лімфоцитів;
- monocyt – показник моноцитів.

До складу таблиці Speciality входять стовпці:

- id – ідентифікатор;
- name – назва.

До складу таблиці Analysis входять стовпці:

- id – ідентифікатор;
- id_card – запис в медичній картці;
- laboratory – лабораторія;
- dateinfo – дата проведення аналізу;
- file – файл аналізу.

До складу таблиці MedCertificate входять стовпці:

- id – ідентифікатор;
- id_declaration – декларація;
- dateinfo – дата оформлення;
- begdate – дата початку дії;
- enddate – дата завершення дії;
- diagnosis – діагноз;
- type – тип.

Схема бази даних представлена на рис. 3.3.

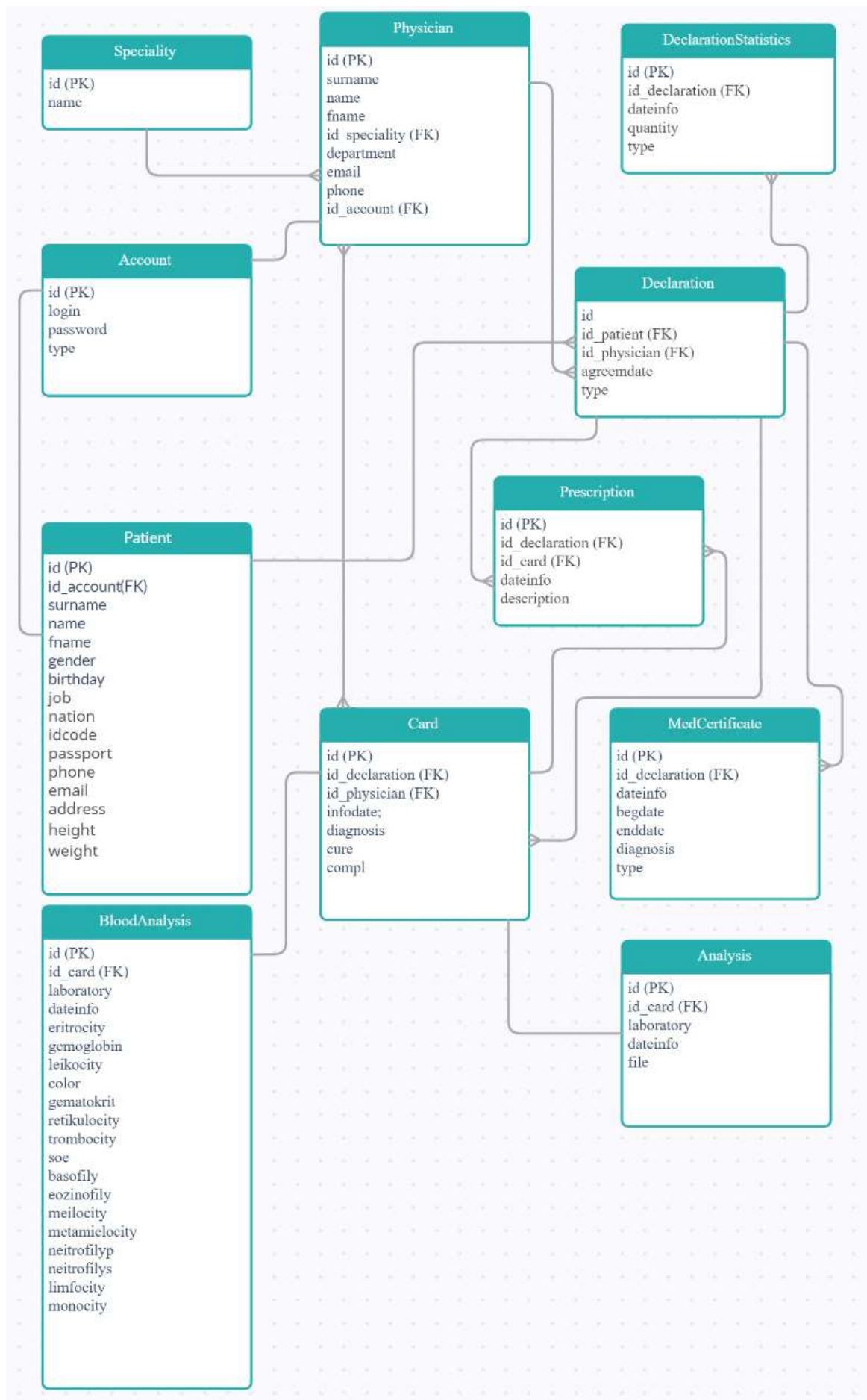


Рисунок 3.3 – Схема бази даних

До складу таблиці DeclarationStatistics входять стовпці:

- id – ідентифікатор;
- id_declaration – декларація;
- dateinfo – дата, якій відповідає статистика;
- quantity – кількість декларацій;
- type – тип запису (прогноз або статистика).

3.4 Структура програмного проєкту

Структурна схема програмного проєкту наведена на рис. 3.4.

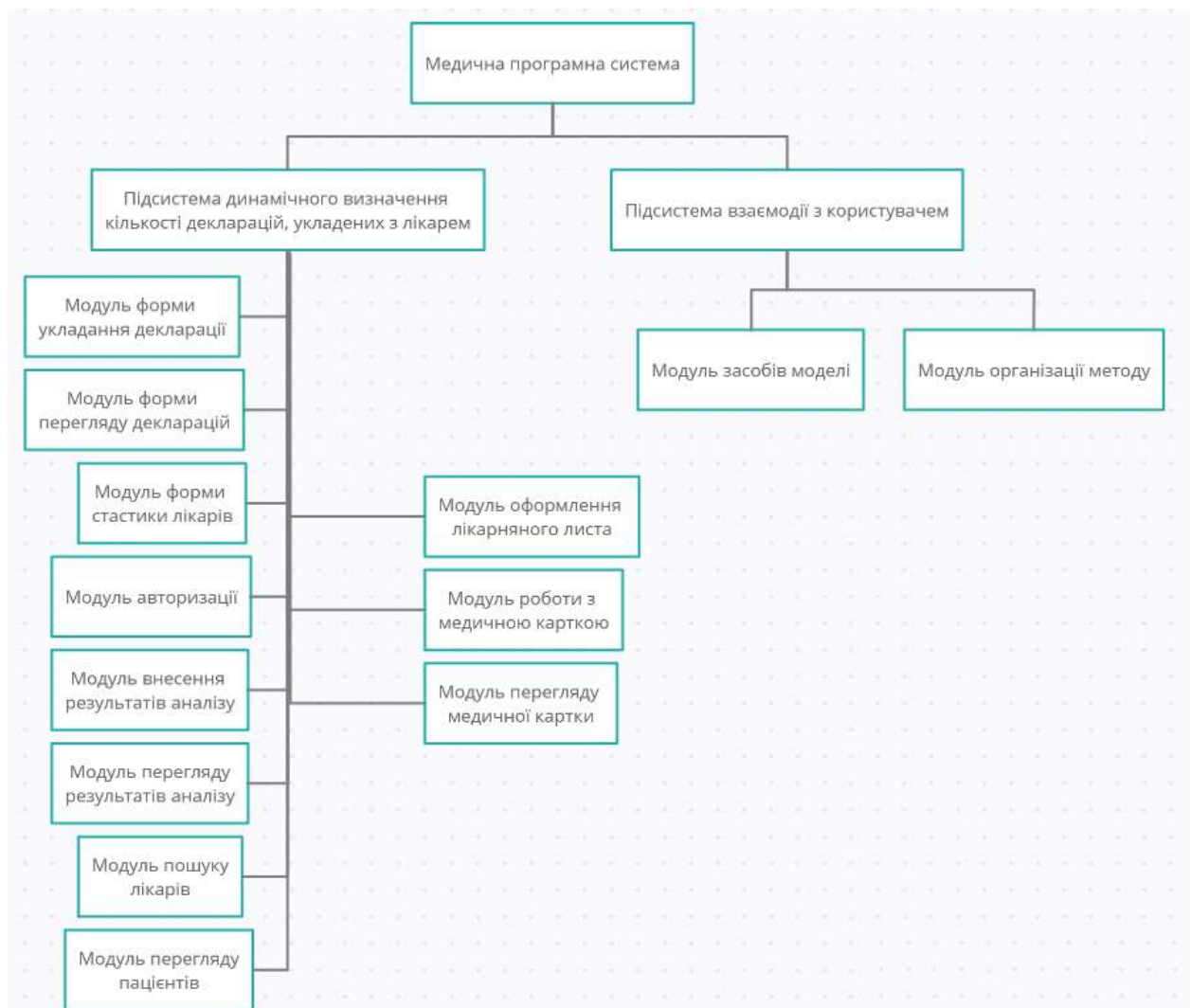


Рисунок 3.4 – Структурна схема програмного проєкту

Програмне забезпечення складається з підсистеми динамічного визначення кількості декларацій, укладених з лікарем, що реалізована мовою Python, та підсистеми взаємодії з користувачем, що реалізована мовою C#.

Підсистема взаємодії з користувачем складається з набору пар файлів, написаних мовами XAML, що визначає графічний інтерфейс користувача у вигляді форми, та C#, що обробляє відповідно дії користувача в таких формах. Деякі з форм винесені на рис. 3.4.

3.5 Висновки за розділом 3

Засоби реалізації програмного забезпечення включають:

- мову C# для реалізації програмного забезпечення;
- Microsoft Visual Studio 2019 в якості інтегрованого середовища розробки;
- Microsoft SQL Server 2019 в якості системи керування базами даних.

Функціональні вимоги до програми, що розробляється, виділені на основі розподілу основних функціональних ролей між лікарем та пацієнтом. Для кожного з них визначено сценарії їх роботи з програмою, в результаті чого побудовано відповідні діаграми прецедентів.

Запроєктовано структуру бази даних зі створенням схеми бази даних та докладним описом таблиць, що були створені, та їх складу.

Створено структурну схему програмного забезпечення, що виходить з використання мов XAML для створення інтерфейсу та C# для оброблення дій користувача.

4 ОСНОВНІ РІШЕННЯ ЩОДО РЕАЛІЗАЦІЇ КОМПОНЕНТІВ СИСТЕМИ

4.1 Алгоритм функціонування системи

У даному підрозділі описано логіку виконання програми в межах взаємодії двох підсистем на основі роботи з даними з бази даних та даними моделі прогнозування.

Робота програми розпочинається з запуску підсистеми динамічного визначення кількості декларацій, укладених з лікарем. У цей момент підсистема перевіряє, чи є збережена на даний момент модель прогнозування.

Якщо модель прогнозування є, то вона завантажується в пам'яті і робота відбувається далі з нею.

Якщо моделі немає, то виконується спочатку формування вибірки даних, а далі створення і навчання моделі.

Після цього виконується цикл звернень, в якому реалізується наступна логіка.

На початку кожного місяця виконується формування статистики, тобто визначення кількості декларацій, які було укладено за місяць. Далі на основі цих даних і даних, раніше збережених в базі даних, а також завантаженої моделі відбувається прогнозування кількості декларацій на наступні періоди.

Якщо це не початок місяця, то реалізується накопичення даних в базі даних, тобто реалізується укладання декларацій з лікарями, в результаті чого накопичується дані, на основі яких на початку наступного місяця реалізується формування статистики, що стає вхідними даними для моделі прогнозування.

На рис. 4.1 представлена відповідна функціональна схема програми.

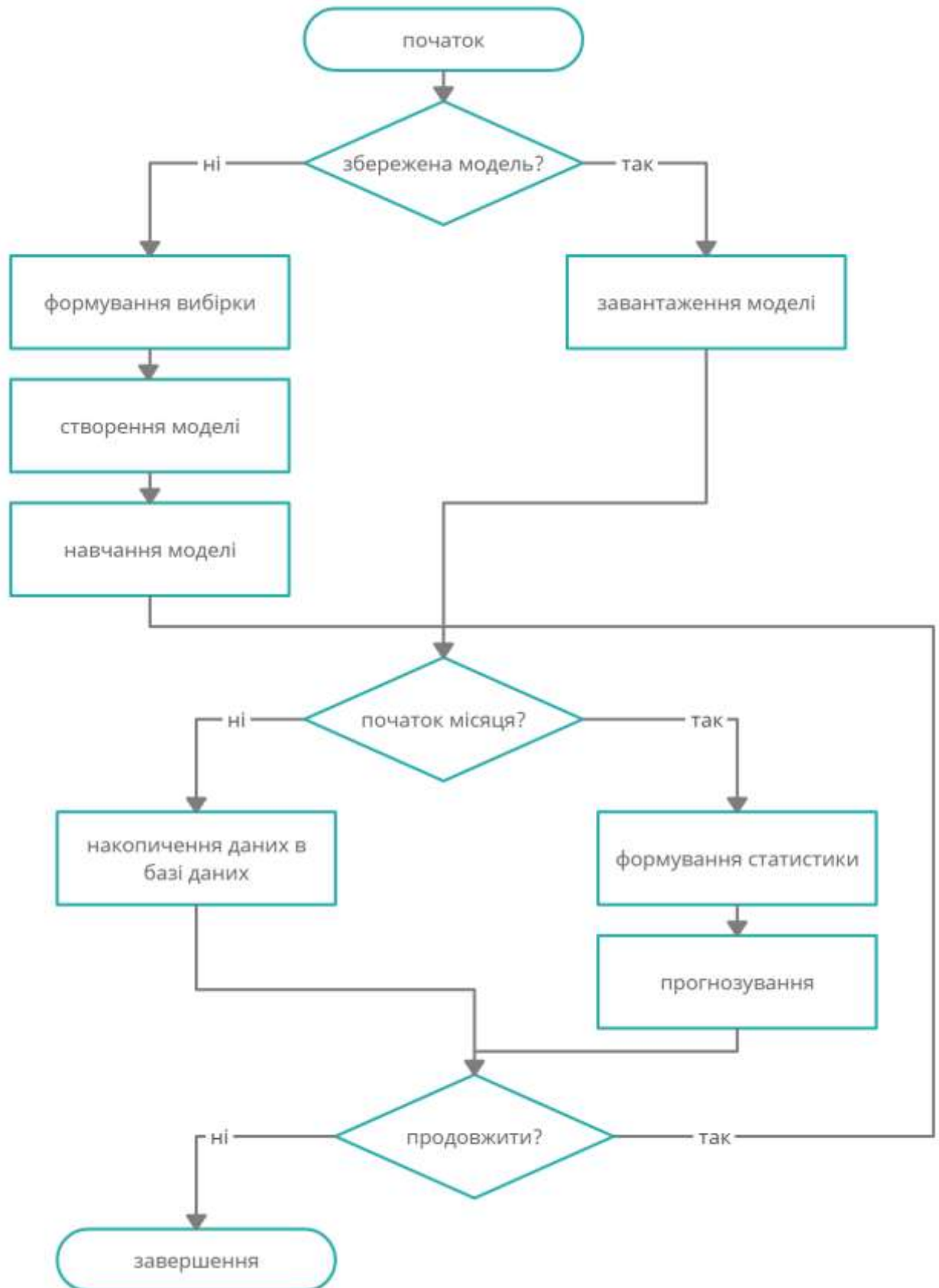


Рисунок 4.1 – Функціональна схема програми

4.2 Опис модулів реалізованої системи

Програмне забезпечення реалізовано на основі наступної логіки: створено класи для роботи з даними, в межах цих класів реалізовано основні сутності предметної області та їх взаємодію з базою даних. Окремо створено класи для реалізації форм, в яких забезпечується звернення до класів для роботи з даними, а також оброблення дій користувача.

До класів предметної області належать (основні):

- Patient – пацієнт;
- Physician – лікар;
- Card – медична картка;
- Declaration – декларація пацієнта з лікарем;
- Prescription – рецепт;
- BloodAnalysis – аналіз крові;
- DeclarationStatistics – статистика підписування декларацій з лікарем;
- MedCertificate – лікарняний лист.

Клас Patient містить такі атрибути:

- int id – ідентифікатор;
- Account acc – акаунт користувача;
- string surname – прізвище;
- string name – ім'я;
- string fname – по батькові;
- byte gender – стать;
- string birthday – дата народження;
- string job – місце роботи;
- string nation – національність;
- float weight – вага;
- float height – зріст;
- string idcode – ідентифікаційний номер;
- string passport – паспорт;

- string phone – номер телефону;
- string email – електронна поштова адреса;
- string address – домашня адреса.

Клас реалізує окрім набору get-терів та set-терів наступні методи.

Метод Patient () – конструктор, який виконує початкове заповнення даними об'єкта.

Метод void Fill() – метод, який реалізує заповнення даними об'єкта, який має номер, що задано в об'єкті.

Для цього забезпечується звернення до бази даних за допомогою використання технології ADO.NET:

```
sqc = new SqlConnection(DBInfo.Parameter);
adapt = new SqlDataAdapter("SELECT * FROM Patient WHERE
Patient.id = " + this.Id, sqc);
tableinfo = new DataSet();
adapt.Fill(tableinfo, "Patient");
```

Після цього відбувається розбір даних, при цьому визначається тільки номер акаунту, а його заповнення реалізується за допомогою аналогічного методу даного класу:

```
foreach (DataRow row in tableinfo.Tables["Patient"].Rows)
{
    this.Acc = new Account();
    this.Acc.Id = row["id_account"];
    this.Acc.Fill();
    this.Surname = row["surname"];
    this.Name = row["name"];
    this.Fname = row["fname"];
}
```

Клас Card реалізує наступні атрибути, що включають повні дані про об'єкти відповідних класів замість використання тільки номерів як у базі даних:

- int id – номер;
- Declaration dec – декларація;
- Physician phys – лікар, який вніс запис в медичну картку (якщо лікар, з яким укладено декларацію, відсутній, наприклад, у відпустці, то адміністратор може дозволити іншому лікарю виконувати його обов'язки, дане рішення фіксує такі ситуації;
- DateTime dateinfo – дата створення запису;
- List<string> description – текст запису.

На основі цих класів створено класи, які реалізують підтримку роботи з переліком об'єктів відповідного базового класу. До таких класів належить PatientList, PhysicianList, DeclarationList тощо.

Кожен такий клас забезпечує наступні методи (на прикладі PatientList):

- void PatientList () – метод зчитування даних з бази даних про всю множину пацієнтів;
- void AddPatient (int accid, string surname, string name, string fname, byte gender, string birthday, string job, string nation, float weight, float height, string idcode, string passport, string phone, string email, string address) – метод додавання до бази даних запису про нового пацієнта;
- void Update (int pid) – метод оновлення даних пацієнта на основі даних з переліку про пацієнта з номером pid.

Робота з моделлю на основі LSTM-мережі реалізована за допомогою бібліотеки Keras.

Засобами даної бібліотеки було створено модель на основі 2 LSTM-шарів нейронної мережі та 1 повнозв'язного шару:

```
MedDeclarationPredictor = tf.keras.models.Sequential([
    tf.keras.layers.LSTM(64, return_sequences=True, input_shape = (24, 1)),
    tf.keras.layers.LSTM(64, return_sequences=True),
    tf.keras.layers.Dense(32),
    tf.keras.layers.Dense(units=3)])
```

4.3 Висновки за розділом 4

Створено функціональну схему програми з описом логіки програмної взаємодії щодо роботи з даними та моделлю.

Описано логіку реалізації програмного забезпечення, основні розроблені класи.

5 ЕКСПЛУАТАЦІЯ ТА ТЕСТУВАННЯ ПРОГРАМИ

5.1 Призначення програми

Програмне забезпечення направлене на зберігання в зручній формі для подальшого аналізу даних, що належать сфері охорони здоров'я і включають накопичені дані щодо взаємодії пацієнта та лікаря.

В якості засобів аналізу основним є визначення кількості укладених декларацій з лікарем, що дозволяє приймати пацієнтам рішення щодо вибору лікаря з одного боку, а з іншого боку – лікарням та лікарям планувати власну діяльність.

5.2 Умови виконання програми

Умовою виконання програмної системи є наявність наступних технічних засобів:

- процесор з частотою не менше 1 ГГц;
- 8 Гб оперативної пам'яті;
- 50 Гб вільного місця на жорсткому диску з обсягом жорсткого диску не менше 100 Гб.

Для серверу, на якому буде розташовано підсистему динамічного визначення кількості декларацій, укладених з лікарем, вимоги включають наступний мінімальний набір:

- процесор з 4 ядрами з частотою 2 ГГц;
- оперативна пам'ять – 4 Гб;
- жорсткий диск з вільним простором у 1 Гб.

На клієнтських комп'ютерах також має бути інстальовано фреймворк .NET Framework версії не нижче 4.7.

5.3 Виконання програми

Після запуску на виконання файлу програми FedMedSystem.exe відкривається вікно авторизації, в якому потрібно ввести дані свого акаунта.

Для лікаря режим роботи з програмою описаний нижче.

Основним вікном для роботи лікаря є вікно з переліком пацієнтів (рис. 5.1), які уклали з ним декларації. Якщо пацієнт укладає декларацію з іншим лікарем, то його дані більше не відображаються в даному вікні.

Сімейний лікар Зоренко Іван Миколайович Додому

Прізвище пацієнта Ім'я По батькові

Номер телефону

Знайти Статистика

Всього пацієнтів: 104

Пацієнт	Телефон	Укладено
Арченко Руслан Петрович	+380505326281	15.02.2020
Асмалов Іван Сергійович	+380501346219	10.10.2019
Борман Микита Олександрович	+380996021428	21.03.2020
Бут Анатолій Михайлович	+380671692438	18.04.2020
Венедіктов Кирило Андрійович	+380674278291	15.02.2020
Вірш Гармен Аванесович	+380504632910	20.06.2020
Воронов Олександр Германович	+380957218439	02.03.2020
Вустлий Іван Мирославович	+380977215639	05.07.2020
Горенко Семен Іванович	+380673026528	02.09.2020
Гудімов Анатолій Сергійович	+380678261927	01.02.2020

Докладніше Медична картка

Рисунок 5.1 – Вікно роботи лікаря

Лікар може або обирати потрібного йому пацієнта з таблиці, або вводити пошукові дані, обмежуючи склад таблиці, а тоді вже обирати

потрібного пацієнта.

Для пошуку пацієнтів лікар може вводити їх прізвище, ім'я та по батькова (пропускаючи за необхідності один або декілька з цих параметрів). Окрім того лікар може виконувати пошук за номером телефону пацієнта.

Після введення пошукових даних потрібно натиснути на кнопку «Знайти».

З цього ж вікна можна обрати одного з пацієнтів і перейти до його медичної картки (рис. 5.2). Окрім того можна за допомогою кнопки «Докладніше» переглянути повні дані пацієнта.

За допомогою кнопки «Статистика» можна переглянути дані про місячний розподіл декларацій з лікарем. За допомогою кнопки «Додому» можна вийти з програми.

Медична картка Додому

Пацієнт Мисніков Петро Андрійович Сімейний лікар Іваненко Леонід Григорович

Інтервал часу з по Обмежити

Записи	Діагноз	Лікування					
<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th>Дата</th> </tr> </thead> <tbody> <tr style="background-color: #007bff; color: white;"> <td>04.12.2020</td> </tr> <tr> <td>29.11.2020</td> </tr> <tr> <td>01.11.2020</td> </tr> <tr> <td> </td> </tr> </tbody> </table>	Дата	04.12.2020	29.11.2020	01.11.2020		Запалення легенів Скарги Біль у грудях під час дихання, важке дихання, слабкість	Рекомендована комп'ютерна томографія легенів. Обов'язкове часте пиття. Карбоцистеїн тричі на добу по 15 мл.
Дата							
04.12.2020							
29.11.2020							
01.11.2020							

Аналізи
Рецепти
Лікарняні
Додати запис

Рисунок 5.2 – Робота з медичною карткою пацієнта

Під час перегляду медичної картки лікар може обмежити дати, за якими відображається перелік записів, встановлюючи потрібний інтервал часу. За допомогою кнопки «Обмежити» можна задати відображення в потрібному інтервалі.

Обираючи з таблиці «Записи» запис за потрібну дату, можна переглянути за обраним записом правіше діагноз, встановлений лікарем, скарги пацієнта та прописане лікування. При цьому в правому куті вгорі відображаються дані лікаря, який вніс цей запис.

З цієї форми можна перейти за допомогою наступних кнопок на інші форми:

- кнопка «Аналізи» дозволяє переглянути аналізи за обраним пацієнтом;
- кнопка «Рецепти» дозволяє переглянути виписані пацієнту рецепти;
- кнопка «Лікарняні» дозволяє переглянути виписані лікарняні пацієнту;
- кнопка «Додати запис» дозволяє створити новий запис в медичній картці пацієнта.

На рис. 5.3 продемонстровано варіант роботи з рецептами, які були виписані пацієнту. Одразу видається весь перелік виписаних рецептів. Перелік відображається в таблиці. Для кожного рецепту вказується його дата і ліки з описом.

За допомогою наступних кнопок можна виконати інші дії:

- кнопка «Завершити» дозволяє завершити роботу з пацієнтом, коли прийом конкретного пацієнта завершений, відбудеться повернення до основної форми;
- кнопка «Додати» дозволяє створити новий рецепт;
- кнопка «Докладніше» дозволяє переглянути повністю рецепт у окремому вікні;
- кнопка «Медична картка» дозволяє повернутися до медичної картки пацієнта.

Пацієнт Колегін Артур Вікторович Додому

Дата	Ліки і опис
07.12.2020	Картопрекс: пігулки 25 мг тричі на добу протягом тижня
11.11.2020	Нурофен форте: пігулки 400 мг двічі на добу по 2 пігулки

Завершити Додати Докладніше Медична картка

Рисунок 5.3 – Робота з рецептами

На рис. 5.4 відображено приклад виписування лікарняного пацієнту.

Лікарняний Додому

Повернутися

Сімейний лікар Афталович Семен Йосипович

Пацієнт Семенко Семен Костянтинович

Дата початку

← грудень 2020 р. →

Пн	Вт	Ср	Чт	Пт	Сб	Нд
30	1	2	3	4	5	6
7	8	9	10	11	12	13
14	15	16	17	18	19	20
21	22	23	24	25	26	27
28	29	30	31	1	2	3
4	5	6	7	8	9	10

☐ Сьогодні: 10.12.2020

Дата завершення

← грудень 2020 р. →

Пн	Вт	Ср	Чт	Пт	Сб	Нд
30	1	2	3	4	5	6
7	8	9	10	11	12	13
14	15	16	17	18	19	20
21	22	23	24	25	26	27
28	29	30	31	1	2	3
4	5	6	7	8	9	10

☐ Сьогодні: 10.12.2020

Тип повторний

Діагноз

Covid

Виписати

Рисунок 5.4 – Лікарняний

За допомогою кнопки «Повернутися» лікар може повернутися на рівень нижче в роботі програми, тобто до переліку лікарняних.

Лікар може визначити дату початку дії лікарняного і дату завершення за допомогою календарів.

Після цього лікар має обрати тип лікарняного та задати діагноз, а далі натиснути на кнопку «Виписати» для збереження даних.

На рис. 5.5 продемонстровано приклад перегляду статистики і прогнозування кількості декларацій, укладених з лікарем.

Сімейний лікар Афталович Семен Йосипович Додому

Рік

Показати

Кількість декларацій

Дата	Кількість
01.2020	187 (+2)
02.2020	189 (+2)
03.2020	195 (+6)
04.2020	199 (+4)
05.2020	200 (+1)
06.2020	200
07.2020	202 (+2)
08.2020	205 (+3)
09.2020	209 (+4)
10.2020	210 (+1)
11.2020	212 (+2)
12.2020	~215 (+3)

Пацієнти

Рисунок 5.5 – Прогнозування кількості укладених декларацій

Дані відображаються для авторизованого лікаря в даному випадку. Біля значення прогнозу вказується символ «~».

Приклад роботи в режимі пацієнта показано на рис. 5.6.

У даному випадку пацієнт укладає декларацію з лікарем. Для цього йому запропоновано обрати лікаря спочатку. Він може обрати одного з лікарів з таблиці, а може спочатку обмежити свій вибір. Для цього пацієнту надаються засоби пошуку лікарів за прізвищем або вказуючи спеціалізацію, медичну установу шляхом обирання з випадаючого переліку.

Пацієнт Колегін Артур Вікторович

Додому

Прізвище лікаря

Ім'я

По батькові

Зоренко

Спеціалізація

Медична установа

Сімейний лікар

Знайти

Лікар	Медична установа
Зоренко Іван Миколайович	Запорізька обласна лікарня
Зоренко Ганна Пилипівна	Запорізька міська лікарня № 5

Докладніше

Укласти декларацію

Рисунок 5.6 – Укладання декларації

Після обрання лікаря можна натиснути на кнопку «Укласти декларацію» і буде виведено вікно з запитом підтвердження з повними

даними лікаря та пацієнта.

За допомогою кнопки «Докладніше» пацієнт може переглянути повні дані про лікаря.

Окрім того користувач-пацієнт може працювати з власною медичною карткою. Для цього він використовує форму, схожу на відповідну для лікаря. Це основна форма його роботи, звідси він може переглядати дані про власне лікування.

6 ЕКОНОМІКО-ОРГАНІЗАЦІЙНА ЧАСТИНА

Програмне забезпечення призначено для прогнозування кількості укладених декларацій з лікарем. Дана діяльність забезпечується на даний момент на підприємстві експертом, який має заробітну платню розміром у 100 гривень на годину.

Програмне забезпечення дозволить підвищити продуктивність праці експерта на 10%. Експерт працює цілий рік по даній тематиці. Таким чином економиться 180 годин машинного часу і оплата 180 годин експерта.

6.1 Планування розробки програмного продукту

Для розрахунку витрат необхідно визначити етапи робіт та їх тривалість [18]. У розробці беруть участь програміст протягом 3 місяців та консультант протягом 0,15 місяця. Розробка програмного забезпечення починається сьомого вересня та повинна бути закінчена до тридцятого листопада 2020 року.

Тривалість робіт визначають за формулою (6.1):

$$T_{\text{ц}} = \frac{Q}{R}, \quad (6.1)$$

де $T_{\text{ц}}$ – тривалість циклу, днів;

Q – трудомісткість, людино-днів;

R – кількість виконавців, людей.

У табл. 6.1 наведена трудомісткість проектування системи.

За даними табл. 6.1 складається зведений стрічковий графік розробки програмного продукту, який представляє собою таблицю, в першому стовпці якої розміщені в порядку збільшення строків початку виконання всі види роботи, а навпроти – календарний період їх виконання [18].

Даний графік наведений на рис. 6.1. Зеленим позначена робота програміста, а червоним – консультанта.

Таблиця 6.1 – Характеристика робіт з розробки програми

Найменування роботи	Трудомісткість		Виконавці	Тривалість, дні
	люд.- дні	% до підсумку		
Аналіз предметної області	3	4,76%	Програміст	3
Узгодження результатів аналізу	4	6,35%	Програміст Консультант	2
Визначення вимог до програмного продукту	5	7,94%	Програміст	5
Узгодження вимог	2	3,17%	Програміст Консультант	1
Документування вимог	2	3,17%	Програміст	2
Проектування структури програми	15	23,81%	Програміст	15
Розробка схеми функціонування програми	5	7,94%	Програміст	5
Створення програмного коду	15	23,81%	Програміст	15
Тестування та налагодження програмного продукту	5	7,94%	Програміст	5
Складання програмної документації	7	11,11%	Програміст	7
Підсумок	63	100		60

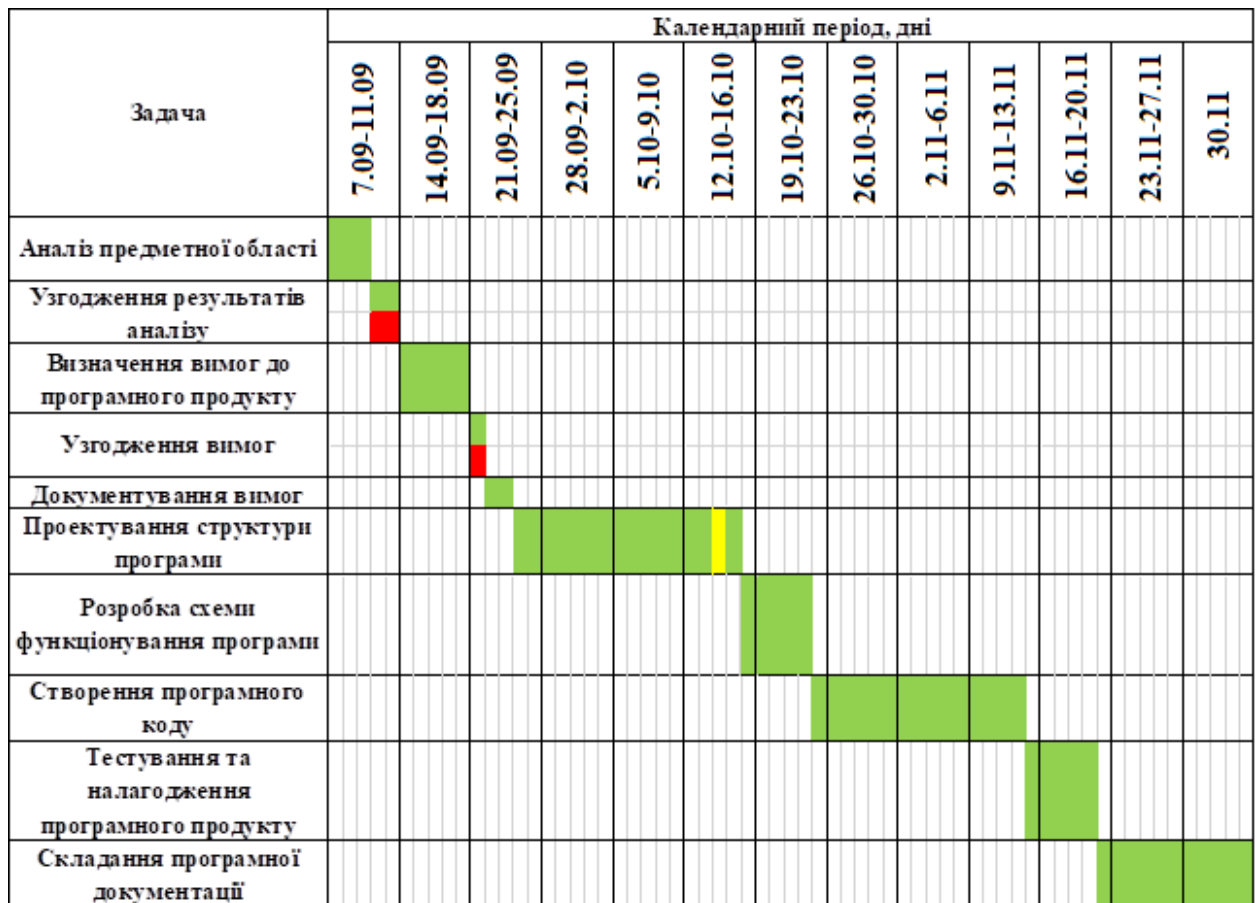


Рисунок 6.1 – Зведений стрічковий графік планування розробки програмного продукту

6.2 Розрахунок основної заробітної плати

Розрахунок зарплати ведеться на підставі даних про трудомісткість, поданих у табл. 6.2.

Таблиця 6.2 – Розрахунок основної заробітної плати

Посада виконавця	Чисельність, чоловік	Місячний оклад, грн.	Кількість місяців роботи	Сума заробітної платні, грн
Програміст	1	5000	3	15000
Консультант	1	8000	0,15	1200
Підсумок	2			16200

6.3 Розрахунок додаткової заробітної плати

Додаткову заробітну плату приймають рівною 10% від основної заробітної плати працівників і розраховують за формулою (6.2):

$$ЗП_{\text{доп}} = ЗП_{\text{осн}} \cdot 0,1 \quad [18]. \quad (6.2)$$

У нашому випадку отримаємо:

$$ЗП_{\text{доп}} = 16200 \cdot 0,1 = 1620 \text{ грн}.$$

6.4 Єдиний соціальний внесок

Єдиний соціальний внесок становить 22% і береться від основної та додаткової заробітної плати на основі формули (6.3).

$$BT_{\text{соц}} = (ЗП_{\text{осн}} + ЗП_{\text{доп}}) \cdot 0,22 \quad [18]. \quad (6.3)$$

Далі отримаємо:

$$BT_{\text{соц}} = (16200 + 1620) \cdot 0,22 = 3920,40 \text{ грн}.$$

6.5 Визначення витрат на матеріали

Використовується 3 найменування матеріалів: диск DVD+RW – 15 грн; картридж – 100 грн, і папір – 100 грн (1 упаковка).

Витрати на матеріали вираховуються за наступною формулою (6.4):

$$BT_{\text{мат}} = \sum_{i=1}^n (C_i \cdot N_i \cdot (1 + K_{\text{м.з.}}) - C_{\text{іо}} \cdot N_{\text{іо}}), \quad (6.4)$$

де BT_{mat} – це витрати на матеріали, покупні напівфабрикати та комплектуючі вироби, грн.;

$K_{m.з.}$ – коефіцієнт, що враховує транспортно-заготівельні витрати;

C_i – ціна i -го найменування матеріалу, напівфабрикату і комплектуючого, грн;

N_i – потреба в i -му матеріалі, напівфабрикати і комплектуючому;

C_{io} – ціна зворотних відходів i -го найменування матеріалу, грн;

N_{io} – кількість зворотних відходів i -го найменування;

n – кількість найменувань матеріалів, напівфабрикатів і комплектуючих [18].

Обчислимо:

$$C_{io} = 0, N_{io} = 0, K_{m.з.} = 0,05, \\ BT_{mat} = (1 + 0,05) \cdot (15 + 100 + 100) = 225.75 \text{ грн.}$$

Разом, витрати на матеріали становлять 225.75 грн.

6.6 Витрати на спеціальне обладнання

Балансова вартість персонального комп'ютера (ПК) – 3000 грн.

Амортизаційні відрахування визначають за формулою (6.5):

$$A = \Phi_{\phi} \cdot \frac{H_a}{100}, \quad (6.5)$$

де Φ_{ϕ} – балансова вартість обчислювальної техніки, грн;

H_a – норма амортизаційних відрахувань на повне відновлення обчислювальної техніки, % [18].

Норма амортизації для ПК становить 25%.

Балансова вартість обчислювальної техніки становить:

$$\Phi_{\phi} = 3000 \text{ грн}.$$

Тоді отримаємо амортизаційні відрахування:

$$A = 3000 \cdot 0,25 = 750 \text{ грн}.$$

Статтю «Експлуатація обладнання» розраховують підсумовуванням витрат на електроенергію та допоміжні матеріали на основі формули (6.6):

$$C_e = N_n \cdot \Phi_{ef} \cdot K_{зч} \cdot K_{зн} \cdot \Pi_{ел.е}, \quad (6.6)$$

де N_n – номінальна потужність електронно-обчислювальної машини (ЕОМ), кВт;

Φ_{ef} – річний ефективний фонд часу роботи ЕОМ, машино-год;

$K_{зч}$ – середній коефіцієнт завантаження по часу;

$K_{зн}$ – коефіцієнт завантаження по потужності;

$\Pi_{ел.е}$ – ціна одного кВт·год електроенергії, грн/(кВт·год) [18].

Номінальна потужність робочої машини – 0,2 кВт. Річний ефективний фонд часу роботи ЕОМ становить 1800 годин. Середні коефіцієнти завантаження за часом і по потужності рівні відповідно 0,9 і 0,6. Ціна однієї кіловат-години електроенергії становить 2,11 грн.

Отримуємо ціну експлуатації обладнання:

$$C_e = 0,2 \cdot 1800 \cdot 0,9 \cdot 0,6 \cdot 2,11 = 410,18 \text{ грн}.$$

Зарплата обслуговуючого персоналу розраховується за формулою (6.7):

$$ЗП_{обсл} = \Phi ЗП_p \cdot (1 + K_{відр}) \cdot \frac{t_{обсл}}{\Phi_{эф.обсл}}, \quad (6.7)$$

де $\Phi ЗП_p$ – річний фонд заробітної плати (основної та додаткової) обслуговуючих робітників, грн;

$K_{відр}$ – коефіцієнт, що враховує відрахування на соціальне страхування й в інші фонди;

$t_{обсл}$ – час протягом року, необхідне на технічне обслуговування ЕОМ, год/рік;

$\Phi_{эф.обсл}$ – річний ефективний фонд часу обслуговуючого персоналу, год/рік [18].

Місячна заробітна плата обслуговуючого персоналу становить 3723 грн., а річний фонд заробітної плати відповідно дорівнює 44676 грн. Річний ефективний фонд робочого часу обслуговуючого ПК працівника дорівнює 1750 год/рік. На обслуговування одного ПК витрачається по 1 годині на місяць, що в рік становить 12 годин.

Отримуємо:

$$ЗП_{обсл} = 44676 \cdot (1 + 0,22) \cdot 12 / 1750 = 373,75 \text{ грн.}$$

Стаття «Поточний ремонт обладнання» приймається рівною 3% від балансової вартості обладнання і становить 90 грн.

Стаття «Інші витрати» приймається рівною 5% від суми всіх попередніх статей витрат на утримання і експлуатацію обладнання. Сума всіх попередніх статей дорівнює 1623,93 грн., 5% від суми становлять 81.20 грн.

Розраховані статті витрат на утримання та експлуатацію обладнання внесено до табл. 6.3.

Таблиця 6.3 – Кошторис витрат на утримання та експлуатацію обладнання

Найменування статей витрат	Сума, грн.
Амортизація обладнання	750
Експлуатація обладнання (крім витрат на поточний ремонт)	410,18
Заробітна плата основна та додаткова обслуговуючих робітників з відрахуваннями на соціальні заходи	373,75
Поточний ремонт обладнання	90
Інші витрати	81,20
Разом	1705,13

Витрати на оплату машинного часу ЕОМ для написання і налагодження даної програми визначаються за формулою (6.8):

$$C_{mo} = B_{екс} \cdot t_{mo}, \quad (6.8)$$

де C_{mo} – витрати на оплату машинного часу, грн;

$B_{екс}$ – експлуатаційні витрати на одну годину машинного часу цієї цифрової ЕОМ, грн/машино-год;

t_{mo} – машинний час цифровий ЕОМ для написання і налагодження даного програмного продукту, машино-год. [18].

Річний ефективний фонд часу роботи ЕОМ дорівнює 1800 годин.

В результаті експлуатаційні витрати на одну годину машинного часу рівні:

$$B_{екс} = 1705,13 / 1800 = 0,95 \text{ грн} / \text{машино} - \text{год}.$$

ЕОМ експлуатується 60 днів в одну зміну, що становить в сумі 480 годин. Таким чином, витрати на оплату машинного часу складуть:

$$C_{mo} = 0,95 \cdot 480 = 456 \text{ грн.}$$

6.7 Розрахунок накладних витрат

Накладні витрати складають 50% до основної заробітної плати розробників, що становить 8100 грн.

Результати визначення витрат на розробку програми у вигляді калькуляції кошторисної вартості робіт наведені в табл. 6.4.

Таблиця 6.4 – Калькуляція кошторисної вартості робіт з розробки програми

Найменування статей витрат	Сума, грн.	Питома вага до підсумку, %
Основна заробітна плата	16200	53,08
Додаткова заробітна плата	1620	5,31
Єдиний соціальний внесок	3920,40	12,84
Матеріали та комплектуючі	225,75	0,73
Витрати на спеціальне обладнання	456	1,49
Накладні витрати	8100	26,54
Разом	30522,15	100

6.8 Розрахунок економічної ефективності програмного продукту

Річну економію від впровадження програми, що дозволяє знизити витрати машинного часу ЕОМ для рішення певного завдання в порівнянні із

програмою, що застосовувалася раніше, визначають за формулою (6.9):

$$E = (T_{M_1} - T_{M_2}) \cdot B_{\text{екс.кгз}} + E_{\text{н}}^a \cdot \frac{\Phi_{\text{б.кгз}} \cdot (T_{M_1} - T_{M_2})}{\Phi_{\text{еф.кгз}}} - \frac{S_{\text{рп}}}{T_{\text{с}}} + E_{\text{зп}}, \quad (6.9)$$

де T_{M_1}, T_{M_2} – машинний час, необхідний для рішення поставлених завдань, відповідно в старому та у новому варіантах, машино-год./рік;

$B_{\text{екс}}$ – експлуатаційні витрати, що доводяться на 1 год машинного часу ЕОМ, грн/ машино-год.;

$E_{\text{н}}^a$ – нормативний коефіцієнт ефективності капітальних вкладень у засоби автоматизації;

$\Phi_{\text{б.кгз}}$ – балансова вартість ЕОМ, грн.;

$\Phi_{\text{еф.кгз}}$ – річний ефективний фонд часу роботи ЕОМ, машино-год.;

$S_{\text{рп}}$ – сумарні витрати на розробку програми, грн.;

$T_{\text{с}}$ – термін служби впроваджуваної програми до її морального зношування, років;

$E_{\text{зп}}$ – економія на заробітній платні [18].

Програмне забезпечення дозволить підвищити продуктивність праці експерта з оплатою праці 100 гривень на годину на 10%. Експерт працює цілий рік по даній тематиці. Таким чином економиться 180 годин машинного часу і оплата 180 годин експерта:

$$E = (1800 \cdot 1,1 - 1800) \cdot 0,95 + 0,15 \cdot (3000 \cdot (1800 \cdot 1,1 - 1800)) / 1800 - 30522,15/5 + 180 \cdot (1 + 0,22) \cdot 100 = 16071,57 \text{ грн.}$$

Коефіцієнт ефективності капітальних вкладень E і строк їхньої окупності $T_{\text{ок}}$ розраховуються за формулами:

$$E\phi = \frac{E}{S_{\text{рп}}}, \quad (6.10)$$

$$T_{\text{ок}} = \frac{S_{\text{рп}}}{E} [18]. \quad (6.11)$$

Обчислимо:

$$E_{\phi} = 16071,57/30522,15 = 0,53,$$

$$T_{\text{ок}} = 30522,15/16071,57 = 1,9 \text{ року.}$$

Зведемо отримані техніко-економічні показники в табл. 6.5.

Таблиця 6.5 – Техніко-економічні показники

Показники	Сума	Одиниця виміру
Витрати на розробку програми	30522,15	грн.
Річна економія	1607,57	грн.
Строк окупності програми	1,9	рік

Строк окупності програми 1,9 року, що значно менше часу її нормативного строку окупності (6,6 року). Річна економія при використанні програми становить 16071,57 гривні. Розробка даного програмного продукту є економічно доцільною.

7 ОХОРОНА ПРАЦІ ТА БЕЗПЕКА В НАДЗВИЧАЙНИХ СИТУАЦІЯХ

7.1 Аналіз потенційних небезпек

В магістерській роботі проводиться дослідження і програмна реалізація методів динамічного визначення кількості укладених декларацій з лікарем.

При роботі фахівця постійно або періодично діють наступні небезпечні і шкідливі чинники:

- ураження електричним струмом, у наслідок несправності електроболаднання, невиконання правил техніки безпеки при користуванні електричним обладнанням, що може призвести до електротравм або летального наслідку;

- вірогідність загоряння, у зв'язку із несправністю електричного обладнання, недотримання, або порушення правил протипожежної безпеки обслуговуючим персоналом, що призводить до пожежі;

- неправильні дії персоналу в умовах надзвичайних ситуацій, які призводять до паніки та загибелі людей [19].

7.2 Заходи по забезпеченню безпеки

У приміщенні офісу застосовується широке різноманіття електроприладів: персональні комп'ютери, принтери, ксерокси, факси, освітлювальні прилади, кондиціонери, побутові електроприлади тощо, небезпека ураження електричним струмом при використанні цих приладів з'являється при недотриманні заходів обережності, а також при відмові або несправності цього обладнання, наслідки ураження електричним струмом залежать від багатьох факторів: опору організму, величини, тривалості дії, роду і частоти струму, шляхів його проходження через життєво важливі органи, умов зовнішнього середовища [19].

Для запобігання ураження електричним струмом встановлено електроустаткування, яке відповідає вимогам:

- правил улаштування електроустановок [19] та ДСТУ Б В.2.5-82:2016 «Електробезпека в будівлях і спорудах. Вимоги до захисних заходів від ураження електричним струмом» [20], величина опору захисного заземлення електрообладнання приміщення – 4 Ом, обладнання офісу має подвійну ізоляцію, яка складається з робочої та додаткової ізоляції;

- державного нормативного акту про охорону праці 40.1-1.32-01 «Правила улаштування електроустановок. Електрообладнання спеціальних установок», приміщення, в якому розташовуються ЕОМ, різноманітне устаткування, відноситься до класу пожежезабезпеченої зони П-Па, тому передбачений мінімальний ступінь захисту ізоляції обладнання IP44, ЕОМ, периферійні пристрої ЕОМ та устаткування для обслуговування, ремонту та налагодження ЕОМ по способу захисту людини від ураження електричним струмом, належать до I класу, оскільки мають подвійну ізоляцію, елемент для заземлення та провід для приєднання до джерела живлення, що має заземлюючу жилу і вилку з заземлюючим контактом [21].

Ймовірність механічного травмування може виникнути внаслідок не раціонального розташування робочих місць, захаращення робочих місць або у зв'язку з недбалістю та неуважністю обслуговуючого персоналу, для виключення травматизму зроблено більш зручне та раціональне розташування робочих місць, таким чином збільшена відстань між ними [20].

Для мінімізації негативного впливу оптичних випромінювань самого монітору, який може призвести до захворювань органів зору потрібно дотримуватися Державних санітарних норм і правил 3.3.2.007-98 «Гігієнічні вимоги до організації роботи з візуальними дисплейними терміналами електронно-обчислювальних машин» [22].

Передбачено такі профілактичні дії:

- оптимальний розподіл між періодом роботи та відпочинку – 10 хвилин на 1 годину;

- робочі місця з візуальними дисплейними терміналами (ВДТ) слід розташовувати так, щоб природнє світло падало з боку, переважно зліва;
- розташовувати ВДТ так, щоб було забезпечено зручність зорового забезпечення [21].

Для запобігання тунельного синдрому потрібно подбати про ергономічність робочого місця та правильне положення рук, при роботі з клавіатурою, кут згину руки в лікті повинен бути прямим (90 градусів), при роботі з мишкою кисть повинна бути прямою, і лежати на столі якнайдалі від краю, стілець або крісло повинно бути з підлокітниками [22].

Інші вимоги:

- наявність опорного пристрою, який дає змогу змінювати кут нахилу клавіатури в межах від 5° до 15° і виготовлений з матеріалу з великим коефіцієнтом тертя, що перешкоджає його переміщенню;
- виділення кольором на клавішах символів різних алфавітів (англійського, українського або російського);
- важливо підібрати мишу, відповідну індивідуальним розміром долоні, занадто велика або занадто маленька миша будуть створювати дискомфорт в кисті, пальці повинні зручно лежати на маніпуляторі;
- періодично потрібно робити перерви для проведення релаксійних вправ [23].

7.3 Заходи по забезпеченню виробничої санітарії та гігієни праці

Внаслідок роботи за ПК, на фізіологію людини негативно впливають електромагнітні випромінювання. Щоб зменшити наслідки впливу на людину та знизити негативні показники у робочій зоні до допустимих значень, згідно з Державним нормативним актом про охорону праці 0.00-7.15-18 «Вимоги щодо безпеки та захисту здоров'я працівників під час роботи з екранними пристроям» [24], вироби, які створюють електромагнітні поля, повинні мати захисні елементи (екрани, поглиначі і т.д.), вимоги до захисних елементів

повинні бути вказані в стандартах та технічних умовах на конкретні види виробів, згідно з Державним нормативним актом про охорону праці 0.00-7.15-18 [24] та Державними санітарними нормами і правилами 3.3.2.007-98 [22], на робочих місцях обладнаних ПК встановлені рідкокристалічні монітори, які не є джерелами рентгенівського та електромагнітного випромінювань [22].

Основними причинами недостатньої або нераціональної освітленості робочих місць є несправність або нераціональний вибір освітлювальних приладів [23].

У офісному приміщенні, згідно Державних будівельних норм В.2.5-28-2018 «Інженерне обладнання будинків і споруд. Природне і штучне освітлення» [25] передбачене природне та штучне освітлення, природне освітлення здійснено через світлові прорізи, які забезпечують коефіцієнт природної освітленості не нижче 1,5%, для захисту від прямих сонячних променів, які створюють прямі та відбиті відблиски на поверхні екранів і клавіатури, передбачено сонцезахисні пристрої, на вікнах встановлені жалюзі або штори, штучне освітлення в приміщенні, здійснено системою загального рівномірного освітлення, як джерела штучного освітлення в приміщенні застосовані люмінесцентні лампи типу ЛБ, при застосуванні яких дитримались наступних умов:

- температура навколишнього повітря не повинна бути нижче, ніж 5°C;
- напруга на освітлювальних приладах повинна бути не менше, ніж 90% номінальної [25].

Проведемо розрахунок штучного освітлення в офісному приміщенні розміри якого складають:

- довжина 18 м;
- ширина 10 м;
- висота 5 м.

Висота робочої поверхні $h_p = 0,7$ м, висота звисання світильника від стелі $h_z = 1,0$ м, рівень нормованого значення освітлення $E_H = 300$ лк, тип світильника – ЛПП, колір стелі, стін, підлоги $\rho_{cm} = 70\%$, $\rho_c = 50\%$, $\rho_n = 30\%$.

Визначимо висоту підвісу світильника h над робочою поверхнею за формулою (7.1):

$$h = H - h_p - h_z, \text{ м}, \quad (7.1)$$

де H – висота приміщення, м;

h_p – висота робочої поверхні, м;

h_z – висота звисання світильника від стелі, м.

Розрахуємо:

$$h = 5 - 0,7 - 1 = 3,3 \text{ м.}$$

Визначимо значення індексу приміщення i , що характеризує співвідношення розмірів освітлювального приміщення і висоти розміщення світильників за формулою (7.2):

$$i = \frac{A \cdot B}{h \cdot (A + B)}, \quad (7.2)$$

де A – довжина приміщення, м;

B – ширина приміщення, м;

h – висота підвісу світильника над робочою поверхнею, м.

Розрахуємо:

$$i = \frac{18 \cdot 10}{3,3 \cdot (18 + 10)} = 1,94.$$

Визначимо значення коефіцієнта використання світлового потоку η , створюваного світильниками вибраного типу. Вибирається в залежності від виду джерела світла, типу обраного світильника, коефіцієнтів відбиття поверхонь приміщення та індексу приміщення $\eta = 62 \%$

Визначимо сумарний світловий потік освітлювальної установки у даному приміщенні Φ_{Σ} за формулою (7.3):

$$\Phi_{\Sigma} = \frac{E_H \cdot A \cdot B \cdot k_z \cdot z}{\eta}, \text{ лк}, \quad (7.3)$$

де E_H – рівень нормованого загального освітлення, лк;

A – довжина приміщення, м;

B – ширина приміщення, м;

k_z – коефіцієнт запасу (для люмінесцентних ламп $k_z = 1,4 \dots 1,7$);

z – коефіцієнт нерівномірності (мінімальної) освітленості (відношення середньої освітленості до мінімальної освітленості), як правило дорівнює (для люмінесцентних ламп $z = 1,1$);

η – коефіцієнт використання світлового потоку.

Розрахуємо:

$$\Phi_z = \frac{300 \cdot 18 \cdot 10 \cdot 1,4 \cdot 1,1}{0,62} = 134129,03 \text{ лм.}$$

Таким чином, сумарний світловий потік освітлювальної установки у даному приміщенні Φ_{Σ} дорівнює 134129,03 лм.

Рівні звукового тиску в октавних смугах частот, рівні звуку та еквівалентні рівні звуку на робочих місцях приміщення відповідають вимогам Державних санітарних правил і норм 3.3.2.007-98 [22] та Державних санітарних норм 3.3.6.037-99 «Санітарні норми виробничого шуму,

ультразвуку та інфразвуку» [26]. Зниження рівня шуму в приміщенні здійснено за допомогою:

- використання більш сучасного обладнання;
- розташування принтерів та різноманітного устаткування колективного користування на значній відстані від більшості робочих місць працівників;
- переведення жорсткого диска в режим сну (Standby), якщо комп'ютер не працює протягом визначеного часу;
- використання блоків живлення ПК з вентиляторами на гумових підвісках;
- метеорологічні умови в приміщенні офісу – температура повітря, відносна вологість повітря й швидкість його переміщення відповідають встановленим санітарно-гігієнічним вимогам Державних санітарних норм 3.3.6.042-99 «Державні санітарні норми мікроклімату виробничих приміщень» [27].

7.4 Заходи по забезпеченню пожежної безпеки

Пожежі в обчислювальному центрі становлять особливу небезпеку, тому що пов'язані з великими матеріальними втратами, як відомо, пожежа може виникнути при взаємодії горючих речовин, окислення і джерел запалювання, у приміщеннях обчислювальних центрів присутні всі три основні чинники, необхідні для виникнення пожежі [28].

Для відводу теплоти від ПК у обчислювальному центрі постійно діє потужна система кондиціонування, тому кисень, як окислювач процесів горіння, є в будь-якій точці обчислювального центру [28].

Обчислювальний центр, згідно ДСТУ Б В.1.1-36:2016 «Визначення категорій приміщень, будинків та зовнішніх установок за вибухопожежною та пожежною небезпекою» [28] відноситься до категорії «П-Па», а клас можливої пожежі, згідно Державних будівельних норм В.1.1-7:2016

«Пожежна безпека об'єктів будівництва. Загальні вимоги» [29], визначається, як «А» [29].

До засобів гасіння пожежі, призначених для локалізації невеликих загорянь, відносяться пожежні стовбури, внутрішні пожежні водопроводи, вогнегасники, сухий пісок, азбестові ковдри і т. п. [28].

Оскільки приміщення (дослідницької лабораторії, конструкторського бюро, тощо) що обладнане ПК з ВДТ має площу 180 м², тому відповідно до вимог п. 3.8 розділу «Типові норми належності вогнегасників» ДСТУ 4297:2004 «Пожежна техніка. Технічне обслуговування вогнегасників. Загальні технічні вимоги» [23] для гасіння електроустановок, що знаходяться під напругою, передбачені вуглекислотні вогнегасники типу ВВК-3,5 у кількості 9 штук (з розрахунку один вогнегасник с величиною заряду вогнегасної речовини 3 кг і більше, на 20 м² площі приміщення), відстань між вогнегасниками та місцями можливих загорянь не перевищує 10 м [28].

Усі працівники обчислювального центру під час прийняття на роботу і в процесі праці повинні проходити протипожежний інструктаж, перевірку знань з питань пожежної безпеки, евакуаційні виходи і проходи утримуються постійно вільними та нічим не захащуються [29].

Відповідальний за протипожежний стан приміщень обчислювального центру повинен періодично (не рідше одного разу на місяць) перевіряти працездатність пристрою, який забезпечує автоматичне вимкнення системи вентиляції у разі пожежі, а також вогнедимозатримувальних пристроїв, під час ремонту слід не допускати руйнування або порушення цілісності негорючих діафрагм у фальш-підлоговому просторі [28].

Відповідальний за пожежну безпеку приміщень обчислювального центру зобов'язаний негайно сповістити технічну службу у разі виявлення незадовільного технічного стану димових та теплових сповіщувачів автоматичної пожежної сигналізації, а також про несправності автоматичної установки об'ємного (газового) пожежогасіння [23].

Усі працівники обчислювального центру повинні вміти привести в дію (в разі потреби) первинні засоби пожежогасіння, регулярно, але не рідше одного разу на квартал, слід очищати від пилу ЕОМ, контрольновимірну апаратуру, кабельні траншеї та фальшпідлоговий простір за допомогою пиловсмоктувача [28].

7.5 Заходи безпеки у надзвичайних ситуаціях

Особливістю хімічних надзвичайних ситуацій є висока швидкість формування і дія факторів ураження, що викликає необхідність прийняття цілого ряду оперативних і попереджувальних заходів, які спрямовані на захист населення і сил цивільної оборони під час ліквідації їх наслідків, з урахуванням специфіки хімічних небезпечних надзвичайних ситуацій при локалізації і ліквідації їх наслідків приймаються заходи, які спрямовані спочатку на обмеження і призупинення викиду (випливу) сильнодіючих отруйних речовин, локалізації хімічного ураження, попередження зараження ґрунту і джерел водозабезпечення населення [30].

З урахуванням специфіки хімічних небезпечних надзвичайних ситуацій при локалізації і ліквідації їх наслідків приймаються заходи, які спрямовані спочатку на обмеження і призупинення викиду (випливу) сильнодіючих отруйних речовин, локалізації хімічного ураження, попередження зараження ґрунту і джерел водозабезпечення населення [30].

При частковій дегазації і дезинфекції з використанням дегазаційних комплектів насамперед оброблюються ті частини і поверхні техніки і транспорту, з якими необхідний контакт при виконанні роботи (поставленої задачі) [30].

Повна дегазація складається з повного обеззаражування або видалення з всієї поверхні техніки і транспорту отруйних речовин шляхом протирання заражених поверхонь розчинами для дегазації; при відсутності їх можуть бути використані розчинники і розчини для дезактивації [30].

Головна вимога до організації першої медичної допомоги – надати її максимальній кількості потерпілих в короткі строки і доставити їх в лікарняні заклади, при цьому легкопоранені можуть евакуйовуватися невеликими групами пішим порядком, потерпілі, які знаходяться у важкому стані, вивозяться транспортом [30].

ВИСНОВКИ

У роботі розглянуто проблему організації медичної системи. Виділено основні вимоги та особливості взаємодії з користувачами, якими характеризуються системи керування охороною здоров'я. Розглянуто програмні аналоги програми. Проаналізовано функціональні можливості українського сервісу Helsi.me та американського сервісу Cerner, що виконує подібні дії в даній предметній області.

Проаналізовано задачу динамічного визначення кількості укладених декларацій з лікарем. Обґрунтовано вибір LSTM-мереж в якості засобу вирішення задачі. Розглянуто особливості LSTM-мереж і проблеми, якими характеризується їх використання.

Обрано засоби реалізації програмного забезпечення, що включають мову C# для реалізації програмного забезпечення, Microsoft Visual Studio 2019 в якості інтегрованого середовища розробки, Microsoft SQL Server 2019 в якості системи керування базами даних.

Функціональні вимоги до програми виділені на основі розподілу основних функціональних ролей між лікарем та пацієнтом. Для кожного з них визначено сценарії їх роботи з програмою, в результаті чого побудовано відповідні діаграми прецедентів.

Запроєктовано структуру бази даних зі створенням схеми бази даних та докладним описом таблиць, що були створені, та їх складу.

Створено структурну схему програмного забезпечення, що виходить з використання мов XAML для створення інтерфейсу та C# для оброблення дій користувача.

Розроблено програмне забезпечення, що дозволяє виконувати оформлення декларацій з лікарем, отримувати лікарям та пацієнтам єдиний доступ до медичної інформації, вести електронні медичні картки.

Наукова новизна роботи полягає в запропонованому методі динамічного визначення кількості укладених декларацій з лікарем, що

характеризується використанням нейронної мережі архітектури LSTM для прогнозування кількості декларацій на майбутню послідовність періодів часу, що дозволяє менеджменту лікарні приймати рішення про планування своєї діяльності.

Усі завдання, які було поставлено на дипломну кваліфікаційну роботу магістра, виконано.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Poluliakh, Y. Effective EHR Development [Electronic resource] : How an EHR May Streamline Your Practice / Yana Poluliakh, Daria Bulatovych. – Access mode : <https://yalantis.com/blog/medical-app-development-how-to-build-a-logbook-app-for-electronic-health-records/>.
2. Doctors give electronic health records an 'F' [Electronic resource]. – Access mode : <https://www.sciencedaily.com/releases/2019/11/191114141238.htm>.
3. Foster, C. University of New Mexico Studies Physician Burnout Related to Electronic Records [Electronic resource] / C. Foster. – Access mode : <https://hsc.unm.edu/news/news/university-of-new-mexico-studies-physician-burnout-related-to-electronic-records.html>.
4. Half, R. 3 Things to Know About Healthcare Management Systems [Electronic resource] / R. Half. – Access mode : <https://www.roberthalf.com/blog/management-tips/3-things-to-know-about-healthcare-management-systems>.
5. What are the Different Types of Healthcare Software? [Electronic resource]. – Access mode : <https://www.policymedical.com/different-types-of-healthcare-software/>.
6. Saini, D. Software Design Framework for Healthcare Systems [Text] / D. Saini // International Journal of Computer Applications. – 2015. – № 116 (4). – Pp. 47-55.
7. Crotty, B. H. Adopting Agile Principles In Health Care [Electronic resource] / Bradley H. Crotty Melek Somai Narath Carlile. – Access mode : <https://www.healthaffairs.org/doi/10.1377/hblog20190813.559504/full/>.
8. Hedges, L. What Is Cerner EMR? A Guide to the Vendor's Solutions [Electronic resource] / L. Hedges. – Access mode : <https://www.softwareadvice.com/resources/what-is-cerner-emr/>.
9. Li, W. Exploring the best sequence LSTM modeling architecture for

flood prediction [Electronic resource] / Wei Li, Amin Kiaghadi, Clint Dawson // Neural Computing and Applications. – 2020. – Vol. 192. – Access mode : <https://link.springer.com/article/10.1007%2Fs00521-020-05334-3>.

10. Николенко, С. Глубокое обучение [Текст] / С. Николенко, А. Кадури, Е. Архангельская. – СПб. : Питер, 2018. – 480 с. : ил.

11. Гудфеллоу, Я. Глубокое обучение [Текст] / Я. Гудфеллоу, И. Бенджио, А. Курвилль. – 2-е изд., испр. – М. : ДМК Пресс, 2018. – 652 с.

12. Sherstinsky, A. Fundamentals of Recurrent Neural Network (RNN) and Long Short-Term Memory (LSTM) network [Electronic resource] / A. Sherstinsky // Physica D. – 2020. – № 404. – Access mode : https://www.researchgate.net/publication/338723814_Fundamentals_of_Recurrent_Neural_Network_RNN_and_Long_Short-Term_Memory_LSTM_network.

13. Houdt, G.V. A review on the long short-term memory model [Text] / Greg Van Houdt, Carlos Mosquera, Gonzalo Napoles // Artificial Intelligence Review. – 2020. – Vol. 53. – Pp. 5929–5955.

14. Sanchez-Sanchez, P. A. Encountered Problems of Time Series with Neural Networks [Electronic resource] : Models and Architectures / Paola Andrea Sanchez-Sanchez, Jose Rafael Garcia-Gonzalez, Leidy Haidy Perez Coronell. – Access mode : <https://www.intechopen.com/books/recent-trends-in-artificial-neural-networks-from-training-to-prediction/encountered-problems-of-time-series-with-neural-networks-models-and-architectures>.

15. Srivastava, P. Essentials of Deep Learning [Electronic resource] : Introduction to Long Short Term Memory / P. Srivastava. – Access mode : <https://www.analyticsvidhya.com/blog/2017/12/fundamentals-of-deep-learning-introduction-to-lstm/>.

16. Прата, С. Язык программирования C++ (C++ 11). Лекции и упражнения [Текст] / С. Прата. – 6-е изд. – М. : Вильямс, 2012. – 1248 с.

17. Скит, Дж. С# для профессионалов [Текст] : тонкости программирования / Дж. Скит. – М. : Вильямс, 2014. – 608 с.

18. Методичні вказівки по економічному обґрунтуванню дипломних

проектів для студентів спеціальностей 8.05010301 «Програмне забезпечення автоматизованих систем» та 8.05010201 «Інформаційні технології проектування» для усіх форм навчання [Текст] / В.В. Остапенко, Є.М. Сердюк. – Запоріжжя : ЗНТУ, 2015. – 22 с.

19. Правила улаштування електроустановок [Текст] : Нормативне виробничо-практичне видання : вид. 5-те, перероблене й доповнене : затв. М-вом енергетики та вугільної промисловості України 20.06.14 : введення в дію з 20.11.14. – Х. : Міненерговугілля України, 2014. – 793 с.

20. Електробезпека в будівлях і спорудах. Вимоги до захисних заходів від ураження електричним струмом [Текст] : ДСТУ Б В.2.5-82:2016. – На заміну ДБН В.2.5-27-2006 ; чинний від 2017-04-01. – К. : ДП «УкрНДНЦ», 2016. – 109 с. – (Державний Стандарт України).

21. Охорона праці [Текст] : навч. посіб. / З. М. Яремко, С. В. Тимошук, О. І. Третяк, Р. М. Ковтун ; за ред. проф. З. М. Яремка. – Львів : Видавничий центр ЛНУ імені Івана Франка, 2010. – 374 с.

22. Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин [Електронний ресурс] : ДСанПіН 3.3.2.007-98. – Чинний від 1998-12-10. – К. : МОЗ України, 1998. – Режим доступу : <http://mozdocs.kiev.ua/view.php?id=2445>.

23. Жидецький, В. Ц. Основи охорони праці [Текст] : підручник / В. Ц. Жидецький. – 5-те вид., доп. – К. : Знання, 2014. – 373 с.

24. Вимоги щодо безпеки та захисту здоров'я працівників під час роботи з екранними пристроям [Текст] : НПАОП 0.00-7.15-18. – На заміну НПАОП 0.00-1.28-10; чинний від 2018-05-18. – К. : Мінсоцполітики України, 2018. – 6 с.

25. ДБН В.2.5-28:2018. Природне та штучне освітлення [Текст] : Посібник: проблеми природного і штучного освітлення. – К. : Етін, 2019. – 180 с.

26. ДСН 3.3.6.037-99 Санітарні норми виробничого шуму,

ультразвуку та інфразвуку [Текст] : Законодавство про охорону праці. Том 1. – Харків : Форт, 2013. – 368 с.

27. ДСН 3.3.6.042-99 Санітарні норми мікроклімату виробничих приміщень [Текст] – Харків : Форт, 2000. – 24 с.

28. Визначення категорій приміщень, будинків та зовнішніх установок за вибухопожежною та пожежною небезпекою [Текст] : ДСТУ Б В.1.1-36:2016. – На заміну НАПБ Б.03.002-2007 ; чинний від 2017-01-01. – К. : Мінрегіонбуд України, 2016. – 66 с. – (Державний Стандарт України).

29. Пожежна безпека об'єктів будівництва. Загальні вимоги [Текст] : ДБН В.1.1-7:2016. – На заміну ДБН В.1.1.7-2002 ; чинний від 2017-06-01. – К. : Мінрегіон України, 2017. – 47 с. – (Державні будівельні норми).

30. Повышение устойчивости работы объектов народного хозяйства в военное время [Текст] : учебное пособие / Г. П. Демиденко, Е. П. Кузьменко, П. П. Орлов, В. А. Пролыгин, И. А. Сидоренко ; под ред. Г. П. Демиденко. – К. : Вища шк. Головное изд-во, 1984. – 232 с.

ДОДАТОК А
Технічне завдання

Вступ

Програмне забезпечення, що розробляється, має накопичувати дані щодо стану здоров'я пацієнта, основою для чого є його медична картка. Для забезпечення ведення медичної картки пацієнти фіксують свої відносини з лікарями шляхом укладання декларації.

Декларації укладаються не тільки з сімейним лікарем, а і з будь-яким іншим лікарем, відповідно тільки за наявності декларації лікар може працювати з медичною картою пацієнта: переглядати її дані та вносити нові дані.

A.1 Підстави для розробки

Розробка виконується на основі завдання на дипломну кваліфікаційну роботу магістра, яке було визначене за темою “Дослідження та програмна реалізація методів динамічного визначення кількості укладених декларацій з лікарем” і затверджене наказом № 301 від 3 листопада 2020 р. за Національним університетом «Запорізька політехніка».

A.2 Призначення розробки

Програмне забезпечення, що розробляється, направлене на зберігання в зручній формі для подальшого аналізу даних, що належать сфері охорони здоров'я і включають накопичені дані щодо взаємодії пацієнта та лікаря.

В якості засобів аналізу основним є визначення кількості укладених декларацій з лікарем, що дозволяє приймати пацієнтам рішення щодо вибору лікаря з одного боку, а з іншого боку – лікарням та лікарям планувати власну діяльність.

A.3 Основні вимоги до програми

A.3.1 Вимоги до функціональних характеристик

У відповідності з описаною логікою за результатами аналізу програмних аналогів та проблеми загалом, в роботі має бути забезпечено реалізацію таких функціональних характеристик програми:

- укладання декларації з лікарем;
- перегляд лікарем власних пацієнтів;
- прогнозування кількості укладених декларацій з лікарем на період;
- перегляд кількості фактично укладених декларацій з лікарем за період;
- додавання записів до медичинської карти пацієнта;
- внесення результатів аналізу крові;
- внесення файлів інших аналізів;
- перегляд результатів аналізів пацієнта;
- прив'язування результату аналізу до запису в медичній картці;
- оформлення лікарняного листа;
- перегляд лікарняних листів пацієнта;
- пошук лікарняних листів;
- пошук пацієнтів;
- пошук лікарів;
- внесення даних пацієнта;
- оформлення рецепту на лікування;
- перегляд рецептів, виданих пацієнту;
- перегляд записів медичної картки;
- авторизація пацієнтів та лікарів.

Вхідні дані визначаються даними акаунту користувача, даними медичних карток, лікарів, пацієнтів, оглядів пацієнтів, проведених аналізів, лікарняних листів.

Вихідні дані визначаються результатами, збереженими в базі даних.

A.3.2 Вимоги до надійності

Дані, які вводяться користувачем, повинні в формах програми перевірятися перед виконанням запитів до бази даних, тобто перед збереженням даних. Мають оброблятися дані, які мають відповідний параметру тип. Програма повинна контролювати даний процес і виводити повідомлення у протилежних випадках.

A.3.3 Умови експлуатації

Для експлуатації програми повинен бути виділений сервер або персональний комп'ютер з наявним мережним з'єднанням, який буде відігравати роль сервера. Доступ до сервера може виконуватися зі звичайних комп'ютерів. На всіх має бути інстальовано операційну систему Windows. Окрім того вимагається наявність встановленого фреймворку .NET Framework версії 4.7 або пізнішої.

A.3.4 Вимоги до складу та параметрів технічних засобів

Для забезпечення коректної роботи програми потрібно мати наступний набір технічних засобів:

- процесор з частотою не менше 1 ГГц;
- 8 Гб оперативної пам'яті;
- 50 Гб вільного місця на жорсткому диску з обсягом жорсткого диску не менше 100 Гб.

A.3.5 Вимоги до транспортування та збереження

Транспортування програми повинно виконуватися шляхом мережного підключення. Тільки за певної необхідності програма може зберігатися на

носії, за допомогою якого виконується транспортування програми.

A.3.6 Вимоги до програмної документації

Не передбачається розробка повного набору програмної документації. У тексті програми наводиться фрагмент програми для її ідентифікації. У технічному завданні визначаються основні початкові параметри розробки.

A.4 Порядок контролю та приймання

Контроль виконання дипломної кваліфікаційної роботи магістра реалізується на основі календарного плану, визначеного в завданні на роботу. Для кожного етапу роботи зазначено строки її виконання. За результатом тринадцятого тижня має бути виконано, підготовлено до захисту та захищено роботу, тоді виконання розробки можна вважати успішним.

ДОДАТОК Б
Текст програми

Б.1 Текст файла Patient.cs

```
using System;
using System.Collections.Generic;
using System.Data;
using System.Data.SqlClient;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace FedMedSystem
{
    public class Patient
    {
        private int id;
        private Account acc;
        private string surname;
        private string name;
        private string fname;
        private byte gender;
        private string birthday;
        private string job;
        private string nation;
        private float weight;
        private float height;
        private string idcode;
        private string passport;
        private string phone;
        private string email;
        private string address;
        SqlConnection sqc;
        SqlDataAdapter adapt;
        DataSet tableinfo;

        public Patient ()
        {
            this.Id = 0;
            this.Acc = new Account();
            this.Surname = "";
            this.Name = "";
            this.Fname = "";
        }
    }
}
```

```

        this.Gender = 0;
        this.Birthday = "";
        this.Job = "";
        this.Nation = "";
        this.Weight = 0;
        this.Height = 0;
        this.Idcode = "";
        this.Passport = "";
        this.Phone = "";
        this.Email = "";
        this.Address = "";
    }

    public int Id
    {
        get => id;
        set => id = value;
    }

    public Account Acc
    {
        get => acc;
        set => acc = value;
    }

    public string Surname
    {
        get => surname;
        set => surname = value;
    }

    public string Name
    {
        get => name;
        set => name = value;
    }

    public string Fname
    {
        get => fname;
        set => fname = value;
    }

```

```
public string SNF
{
    get => surname+""+name+""+fname;
}

public byte Gender
{
    get => gender;
    set => gender = value;
}

public string Birthday
{
    get => fname;
    set => fname = value;
}

public string Job
{
    get => job;
    set => job = value;
}

public string Nation
{
    get => nation;
    set => nation = value;
}

public float Weight
{
    get => weight;
    set => weight = value;
}

public float Height
{
    get => height;
    set => height = value;
}

public string Idcode
{
    get => idcode;
```

```

        set => idcode = value;
    }

    public string Passport
    {
        get => passport;
        set => passport = value;
    }

    public string Phone
    {
        get => phone;
        set => phone = value;
    }

    public string Email
    {
        get => email;
        set => email = value;
    }

    public string Address
    {
        get => address;
        set => address = value;
    }

    public void Fill()
    {
        sqc = new SqlConnection(DBInfo.Parameter);
        adapt = new SqlDataAdapter("SELECT * FROM Patient
WHERE Patient.id = " + this.Id, sqc);
        tableinfo = new DataSet();
        adapt.Fill(tableinfo, "Patient");
        foreach (DataRow row in
tableinfo.Tables["Patient"].Rows)
        {
            this.Acc = new Account();
            this.Acc.Id = row["id_account"];
            this.Acc.Fill();
            this.Surname = row["surname"];
            this.Name = row["name"];
            this.Fname = row["fname"];

```

```

        this.Gender = row["gender"];
        this.Birthday = row["birthday"];
        this.Job = row["job"];
        this.Nation = row["nation"];
        this.Weight = row["weight"];
        this.Height = row["height"];
        this.Idcode = row["idcode"];
        this.Passport = row["passport"];
        this.Phone = row["phone"];
        this.Email = row["email"];
        this.Address = row["address"];
    }

}

}
}

```

Б.2 Текст файла PatientList.cs

```

using System;
using System.Collections.Generic;
using System.Data;
using System.Data.SqlClient;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace FedMedSystem
{
    public class PatientList
    {
        private List<Patient> lp;
        SqlConnection sqc;
        SqlDataAdapter adapt;
        DataSet tableinfo;

        public void PatientList ()
        {
            sqc = new SqlConnection(DBInfo.Parameter);

```

```

        adapt = new SqlDataAdapter("SELECT * FROM Patient",
sqc);

        tableinfo = new DataSet ();
        adapt.Fill(tableinfo, "Patient");
        foreach (DataRow row in
tableinfo.Tables["Patient"].Rows)
        {
            Patient p = new Patient();
            p.Id = row["id"];
            p.Acc = new Account();
            p.Acc.Id = row["id_account"];
            p.Acc.Fill();
            p.Surname = row["surname"];
            p.Name = row["name"];
            p.Fname = row["fname"];
            p.Gender = row["gender"];
            p.Birthday = row["birthday"];
            p.Job = row["job"];
            p.Nation = row["nation"];
            p.Weight = row["weight"];
            p.Height = row["height"];
            p.Idcode = row["idcode"];
            p.Passport = row["passport"];
            p.Phone = row["phone"];
            p.Email = row["email"];
            p.Address = row["address"];
            pl.Add(p);
        }
    }

```

```

        public void AddPatient (int accid, string surname, string
name, string fname, byte gender, string birthday, string job, string
nation, float weight, float height, string idcode, string passport,
string phone, string email, string address)
        {
            Patient p = new Patient();
            p.Acc = new Account();
            p.Acc.Id = accid;
            p.Acc.Fill();
            p.Surname = surname;
            p.Name = name;
            p.Fname = fname;
            p.Gender = gender;

```

```

p.Birthday = birthday;
p.Job = job;
p.Nation = nation;
p.Weight = weight;
p.Height = height;
p.Idcode = idcode;
p.Passport = passport;
p.Phone = phone;
p.Email = email;
p.Address = address;
lp.Add(p);
sqc = new SqlConnection(DBInfo.Parameter);
var q = "select * from Patient where 0 = 1";
adapt = new SqlDataAdapter(q, sqc);
tableinfo = new DataSet();
adapt.Fill(tableinfo);
var r = tableinfo.Tables["Patient"].NewRow();
newRow["id_account"] = p.Acc.Id;
tableinfo.Tables["Patient"].Rows.Add(r);
r = tableinfo.Tables["Patient"].NewRow();
newRow["surname"] = p.Surname;
tableinfo.Tables["Patient"].Rows.Add(r);
r = tableinfo.Tables["Patient"].NewRow();
newRow["name"] = p.Name;
tableinfo.Tables["Patient"].Rows.Add(r);
r = tableinfo.Tables["Patient"].NewRow();
newRow["fname"] = p.Fname;
tableinfo.Tables["Patient"].Rows.Add(r);
r = tableinfo.Tables["Patient"].NewRow();
newRow["gender"] = p.Gender;
tableinfo.Tables["Patient"].Rows.Add(r);
r = tableinfo.Tables["Patient"].NewRow();
newRow["birthday"] = p.Birthday;
tableinfo.Tables["Patient"].Rows.Add(r);
r = tableinfo.Tables["Patient"].NewRow();
newRow["job"] = p.Job;
tableinfo.Tables["Patient"].Rows.Add(r);
r = tableinfo.Tables["Patient"].NewRow();
newRow["nation"] = p.Nation;
tableinfo.Tables["Patient"].Rows.Add(r);
r = tableinfo.Tables["Patient"].NewRow();
newRow["weight"] = p.Weight;
tableinfo.Tables["Patient"].Rows.Add(r);

```

```

        r = tableinfo.Tables["Patient"].NewRow();
        newRow["height"] = p.Height;
        tableinfo.Tables["Patient"].Rows.Add(r);
        r = tableinfo.Tables["Patient"].NewRow();
        newRow["idcode"] = p.Idcode;
        tableinfo.Tables["Patient"].Rows.Add(r);
        r = tableinfo.Tables["Patient"].NewRow();
        newRow["passport"] = p.Passport;
        tableinfo.Tables["Patient"].Rows.Add(r);
        r = tableinfo.Tables["Patient"].NewRow();
        newRow["phone"] = p.Phone;
        tableinfo.Tables["Patient"].Rows.Add(r);
        r = tableinfo.Tables["Patient"].NewRow();
        newRow["email"] = p.Email;
        tableinfo.Tables["Patient"].Rows.Add(r);
        r = tableinfo.Tables["Patient"].NewRow();
        newRow["address"] = p.Address;
        tableinfo.Tables["Patient"].Rows.Add(r);
        new SqlCommandBuilder(adapt);
        adapt.Insert(tableinfo);
    }

    public void Update (int pid)
    {
        sqc = new SqlConnection(DBInfo.Parameter);
        var q = "select * from Patient where 0 = 1";
        adapt = new SqlDataAdapter(q, sqc);
        tableinfo = new DataSet();
        adapt.Fill(tableinfo);
        var r = tableinfo.Tables["id"].NewRow();
        newRow["id"] = pid;
        tableinfo.Tables["Patient"].Rows.Add(r);
        r = tableinfo.Tables["Patient"].NewRow();
        newRow["id_account"] = p.get(pid).Acc.Id;
        tableinfo.Tables["Patient"].Rows.Add(r);
        r = tableinfo.Tables["Patient"].NewRow();
        newRow["surname"] = p.get(pid).Surname;
        tableinfo.Tables["Patient"].Rows.Add(r);
        r = tableinfo.Tables["Patient"].NewRow();
        newRow["name"] = p.get(pid).Name;
        tableinfo.Tables["Patient"].Rows.Add(r);
        r = tableinfo.Tables["Patient"].NewRow();
        newRow["fname"] = p.get(pid).Fname;
    }

```

```

tableinfo.Tables["Patient"].Rows.Add(r);
r = tableinfo.Tables["Patient"].NewRow();
newRow["gender"] = p.get(pid).Gender;
tableinfo.Tables["Patient"].Rows.Add(r);
r = tableinfo.Tables["Patient"].NewRow();
newRow["birthday"] = p.get(pid).Birthday;
tableinfo.Tables["Patient"].Rows.Add(r);
r = tableinfo.Tables["Patient"].NewRow();
newRow["job"] = p.get(pid).Job;
tableinfo.Tables["Patient"].Rows.Add(r);
r = tableinfo.Tables["Patient"].NewRow();
newRow["nation"] = p.get(pid).Nation;
tableinfo.Tables["Patient"].Rows.Add(r);
r = tableinfo.Tables["Patient"].NewRow();
newRow["weight"] = p.get(pid).Weight;
tableinfo.Tables["Patient"].Rows.Add(r);
r = tableinfo.Tables["Patient"].NewRow();
newRow["height"] = p.get(pid).Height;
tableinfo.Tables["Patient"].Rows.Add(r);
r = tableinfo.Tables["Patient"].NewRow();
newRow["idcode"] = p.get(pid).Idcode;
tableinfo.Tables["Patient"].Rows.Add(r);
r = tableinfo.Tables["Patient"].NewRow();
newRow["passport"] = p.get(pid).Passport;
tableinfo.Tables["Patient"].Rows.Add(r);
r = tableinfo.Tables["Patient"].NewRow();
newRow["phone"] = p.get(pid).Phone;
tableinfo.Tables["Patient"].Rows.Add(r);
r = tableinfo.Tables["Patient"].NewRow();
newRow["email"] = p.get(pid).Email;
tableinfo.Tables["Patient"].Rows.Add(r);
r = tableinfo.Tables["Patient"].NewRow();
newRow["address"] = p.get(pid).Address;
tableinfo.Tables["Patient"].Rows.Add(r);
new SqlCommandBuilder(adapt);
adapt.Update(tableinfo);

```

```

}

```

```

}

```

```

}

```

Б.3 Текст файла Physician.cs

```

using System;
using System.Collections.Generic;
using System.Data;
using System.Data.SqlClient;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace FedMedSystem
{
    public class Physician
    {
        private int id;
        private Account acc;
        private string surname;
        private string name;
        private string fname;
        private Speciality spec;
        private string department;
        private string phone;
        private string email;
        SqlConnection sqc;
        SqlDataAdapter adapt;
        DataSet tableinfo;

        public Physician ()
        {
            this.Id = 0;
            this.Acc = new Account();
            this.Surname = "";
            this.Name = "";
            this.Fname = "";
            this.Department = "";
            this.Spec = new Speciality();
            this.Phone = "";
            this.Email = "";
        }

        public int Id
        {

```

```

        get => id;
        set => id = value;
    }

    public Account Acc
    {
        get => acc;
        set => acc = value;
    }

    public string Surname
    {
        get => surname;
        set => surname = value;
    }

    public string Name
    {
        get => name;
        set => name = value;
    }

    public string Fname
    {
        get => fname;
        set => fname = value;
    }

    public string SNF
    {
        get => spec.name+" "+surname+" "+name+" "+fname;
    }

    public string Department
    {
        get => gender;
        set => gender = value;
    }

    public Speciality Spec
    {
        get => fname;
        set => fname = value;
    }

```

```

        public string Phone
        {
            get => phone;
            set => phone = value;
        }

        public string Email
        {
            get => email;
            set => email = value;
        }

        public void Fill()
        {
            sqc = new SqlConnection(DBInfo.Parameter);
            adapt = new SqlDataAdapter("SELECT * FROM Physician
WHERE Physician.id = " + this.Id, sqc);
            tableinfo = new DataSet();
            adapt.Fill(tableinfo, "Physician");
            foreach (DataRow row in
tableinfo.Tables["Physician"].Rows)
            {
                this.Acc = new Account();
                this.Acc.Id = row["id_account"];
                this.Acc.Fill();
                this.Surname = row["surname"];
                this.Name = row["name"];
                this.Fname = row["fname"];
                this.Spec = new Speciality();
                this.Spec.Id = row["id_speciality"];
                this.Spec.Fill();
                this.Department = row["department"];
                this.Phone = row["phone"];
                this.Email = row["email"];
            }
        }
    }
}

```

Б.4 Текст файла Card.cs

```

using System;
using System.Collections.Generic;
using System.Data;
using System.Data.SqlClient;
using System.Globalization;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace FedMedSystem
{
    public class Card
    {
        private int id;
        private Declaration dec;
        private Physician phys;
        private DateTime dateinfo;
        private List<string> description;
        SqlConnection sqc;
        SqlDataAdapter adapt;
        DataSet tableinfo;

        public Card ()
        {
            this.Id = 0;
            this.Dec = new Account();
            this.Phys = "";
            this.Dateinfo = DateTime.Now;
            this.Description = "";
        }

        public int Id
        {
            get => id;
            set => id = value;
        }

        public Declaration Dec
        {
            get => dec;

```

```

        set => dec = value;
    }

    public int DecId
    {
        get => dec.id;
    }

    public Physician Phys
    {
        get => phys;
        set => phys = value;
    }

    public string PhysData
    {
        get => phys.SNF;
    }

    public DateTime Dateinfo
    {
        get => dateinfo;
        set => dateinfo = value;
    }

    public List<string> Description
    {
        get => description;
        set => description = value;
    }

    public void Fill()
    {
        sqc = new SqlConnection(DBInfo.Parameter);
        adapt = new SqlDataAdapter("SELECT * FROM Card
WHERE Card.id = " + this.Id, sqc);
        tableinfo = new DataSet();
        adapt.Fill(tableinfo, "Card");
        foreach (DataRow row in
tableinfo.Tables["Card"].Rows)
        {
            this.Dec = new Declaration();
            this.Dec.Id = row["id_declaration"];
            this.Dec.Fill();

```

```

        this.Phys = new Physician();
        this.Phys.Id = row["id_physician"];
        this.Phys.Fill();
        this.Dateinfo = row["dateinfo"];
        this.Description = row["description"];
    }

}

}
}

```

Б.5 Текст файла BloodAnalysis.cs

```

using System;
using System.Collections.Generic;
using System.Data;
using System.Data.SqlClient;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace FedMedSystem
{
    public class BloodAnalysis
    {
        private int id;
        private Card medcard;
        private string laboratory;
        private DateTime dateinfo;
        private float eritrocity;
        private float gemoglobin;
        private float leikocity;
        private float color;
        private float gematokrit;
        private float retikulocity;
        private float trombocity;
        private float soe;
        private float basofily;
        private float eozinofily;
        private float meilocity;
    }
}

```

```

private float metamielocity;
private float neitrofilyp;
private float neitrofilys;
private float limfocity;
private float monocity;
SqlConnection sqc;
SqlDataAdapter adapt;
DataSet tableinfo;

public BloodAnalysis ()
{
    this.Id = 0;
    this.Medcard = new Card();
    this.Laboratory = "";
    this.Dateinfo = DateTime.Now;
    this.Eritrocity = 0.0;
    this.Gemoglobin = 0.0;
    this.Leikocity = 0.0;
    this.Color = 0.0;
    this.gematokrit = 0.0;
    this.Retikulocity = 0.0;
    this.Trombocity = 0.0;
    this.Soe = 0.0;
    this.Basofily = 0.0;
    this.Eozinofily = 0.0;
    this.Meilocity = 0.0;
    this.Metamielocity = 0.0;
    this.Neutrofilyp = 0.0;
    this.Neutrofilys = 0.0;
    this.Limfocity = 0.0;
    this.Monocity = 0.0;
}

public int Id
{
    get => id;
    set => id = value;
}

public Card Medcard
{
    get => medcard;
    set => medcard = value;
}

```

```
}

public string Laboratory
{
    get => laboratory;
    set => laboratory = value;
}

public DateTime Dateinfo
{
    get => dateinfo;
    set => dateinfo = value;
}

public float Eritrocity
{
    get => eritrocity;
    set => eritrocity = value;
}

public float Gemoglobin
{
    get => gemoglobin;
    set => gemoglobin = value;
}

public float Leikocity
{
    get => leikocity;
    set => leikocity = value;
}

public float Color
{
    get => color;
    set => color = value;
}

public float Gematokrit
{
```

```
        get => gematokrit;  
        set => gematokrit = value;  
    }  
  
    public float Retikulocity  
    {  
        get => retikulocity;  
        set => retikulocity = value;  
    }  
  
    public float Trombocity  
    {  
        get => trombocity;  
        set => trombocity = value;  
    }  
  
    public float Soe  
    {  
        get => soe;  
        set => soe = value;  
    }  
  
    public float Basofily  
    {  
        get => basofily;  
        set => basofily = value;  
    }  
  
    public float Eozinofily  
    {  
        get => eozinofily;  
        set => eozinofily = value;  
    }  
  
    public float Meilocity  
    {  
        get => meilocity;  
        set => meilocity = value;  
    }
```

```

public float Metamielocity
{
    get => metamielocity;
    set => metamielocity = value;
}

public float Neitrofilyp
{
    get => neitrofilyp;
    set => neitrofilyp = value;
}

public float Neitrofilys
{
    get => neitrofilys;
    set => neitrofilys = value;
}

public float Limfocity
{
    get => limfocity;
    set => limfocity = value;
}

public float Monocity
{
    get => monocity;
    set => monocity = value;
}

public void Fill()
{
    sqc = new SqlConnection(DBInfo.Parameter);
    adapt = new SqlDataAdapter("SELECT * FROM
BloodAnalysis WHERE BloodAnalysis.id = " + this.Id, sqc);
    tableinfo = new DataSet();
    adapt.Fill(tableinfo, "BloodAnalysis");
    foreach (DataRow row in
tableinfo.Tables["BloodAnalysis"].Rows)
    {
        this.Medcard = new Card();
        this.Medcard.Id = row["id_card"];
    }
}

```

```

        this.Medcard.Fill();
        this.Laboratory = row[""];
        this.Dateinfo = row["dateinfo"];
        this.Eritrocity = row["eritrocity"];
        this.Gemoglobin = row["gemoglobin"];
        this.Leikocity = row["leikocity"];
        this.Color = row["color"];
        this.Gematokrit = row["gematokrit"];
        this.Retikulocity = row["retikulocity"];
        this.Trombocity = row["trombocity"];
        this.Soe = row["soe"];
        this.Basofily = row["basofily"];
        this.Eozinofily = row["eozinofily"];
        this.Meilocity = row["meilocity"];
        this.Metamielocity = row["metamielocity"];
        this.Neitrofilyp = row["neitrofilyp"];
        this.Neitrofilys = row["neitrofilys"];
        this.Limfocity = row["limfocity"];
        this.Monocity = row["monocity"];
    }

}

}
}

```

Б.6 Текст файлу MedCertificate.cs

```

using System;
using System.Collections.Generic;
using System.Data;
using System.Data.SqlClient;
using System.Globalization;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace FedMedSystem
{
    public class MedCertificate

```

```

{
    private int id;
    private Declaration dec;
    private string laboratory;
    private DateTime dateinfo;
    private DateTime begdate;
    private DateTime enddate;
    private string diagnosis;
    private byte type;
    SqlConnection sqc;
    SqlDataAdapter adapt;
    DataSet tableinfo;

    public MedCertificate ()
    {
        this.Id = 0;
        this.Dec = new Declaration ();
        this.Laboratory = "";
        this.Dateinfo = DateTime.Now;
        this.Begdate = DateTime.Now;
        this.Enddate = DateTime.Now;
        this.Diagnosis = "";
        this.Type = 0;
    }

    public int Id
    {
        get => id;
        set => id = value;
    }

    public Declaration Dec
    {
        get => dec;
        set => dec = value;
    }

    public int DecId
    {
        get => dec.Id;
    }

    public Physician Phys

```

```

{
    get => dec.Phys;
}

public string PhysInfo
{
    get => dec.Phys.SNF;
}

public Patient pat
{
    get => dec.Pat;
}

public string PatInfo
{
    get => dec.Pat.SNF;
}

public string Laboratory
{
    get => laboratory;
    set => laboratory = value;
}

public DateTime Dateinfo
{
    get => dateinfo;
    set => dateinfo = value;
}

public DateTime Begdate
{
    get => begdate;
    set => begdate = value;
}

public DateTime Enddate
{
    get => enddate;
    set => enddate = value;
}

```

```

public string Diagnosis
{
    get => diagnosis;
    set => diagnosis = value;
}

public byte Type
{
    get => type;
    set => type = value;
}

public void Fill()
{
    sqc = new SqlConnection(DBInfo.Parameter);
    adapt = new SqlDataAdapter("SELECT * FROM
MedCertificate WHERE MedCertificate.id = " + this.Id, sqc);
    tableinfo = new DataSet();
    adapt.Fill(tableinfo, "MedCertificate");
    foreach (DataRow row in
tableinfo.Tables["MedCertificate"].Rows)
    {
        this.Dec = new Account();
        this.Dec.Id = row["id_declaration"];
        this.Dec.Fill();
        this.Laboratory = row["laboratory"];
        this.Dateinfo = row["dateinfo"];
        this.Begdate = row["begdate"];
        this.Enddate = row["enddate"];
        this.Diagnosis = row["diagnosis"];
        this.Type = row["type"];
    }
}

}

}

```

ДОДАТОК В
Слайди презентації

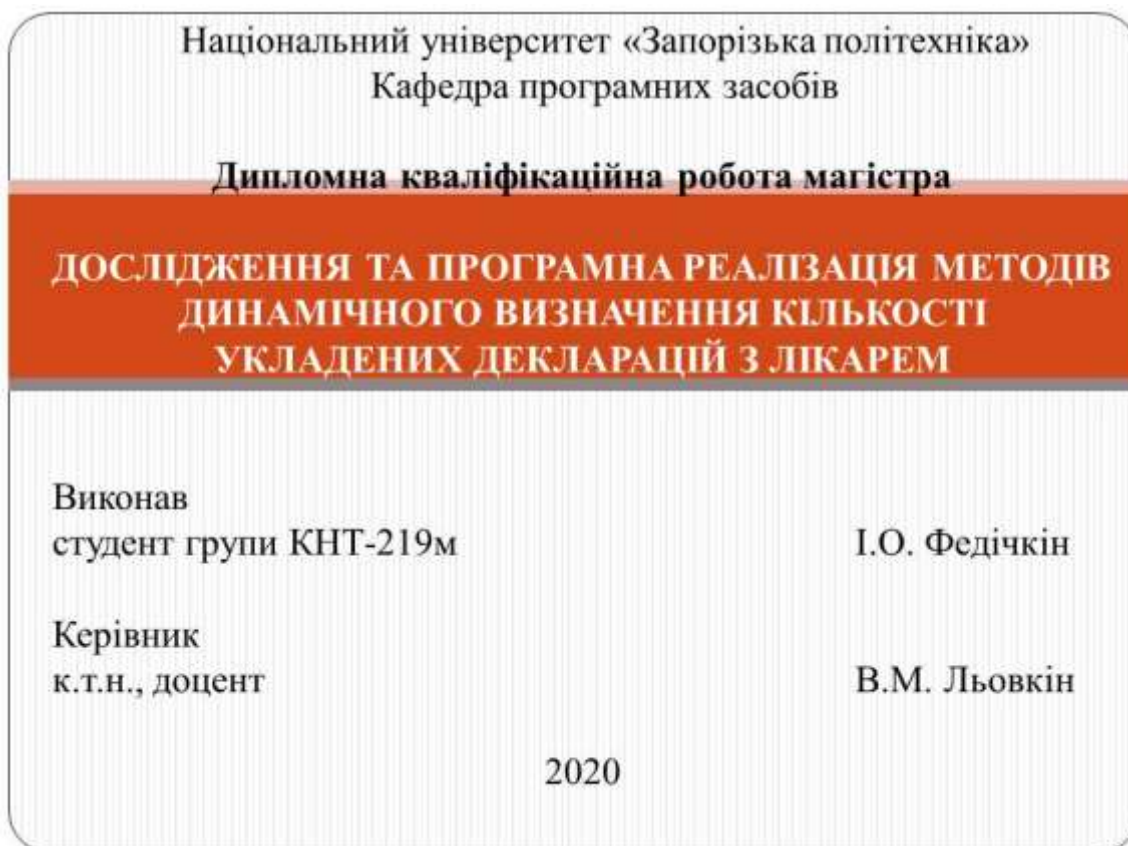


Рисунок В.1 – Слайд 1

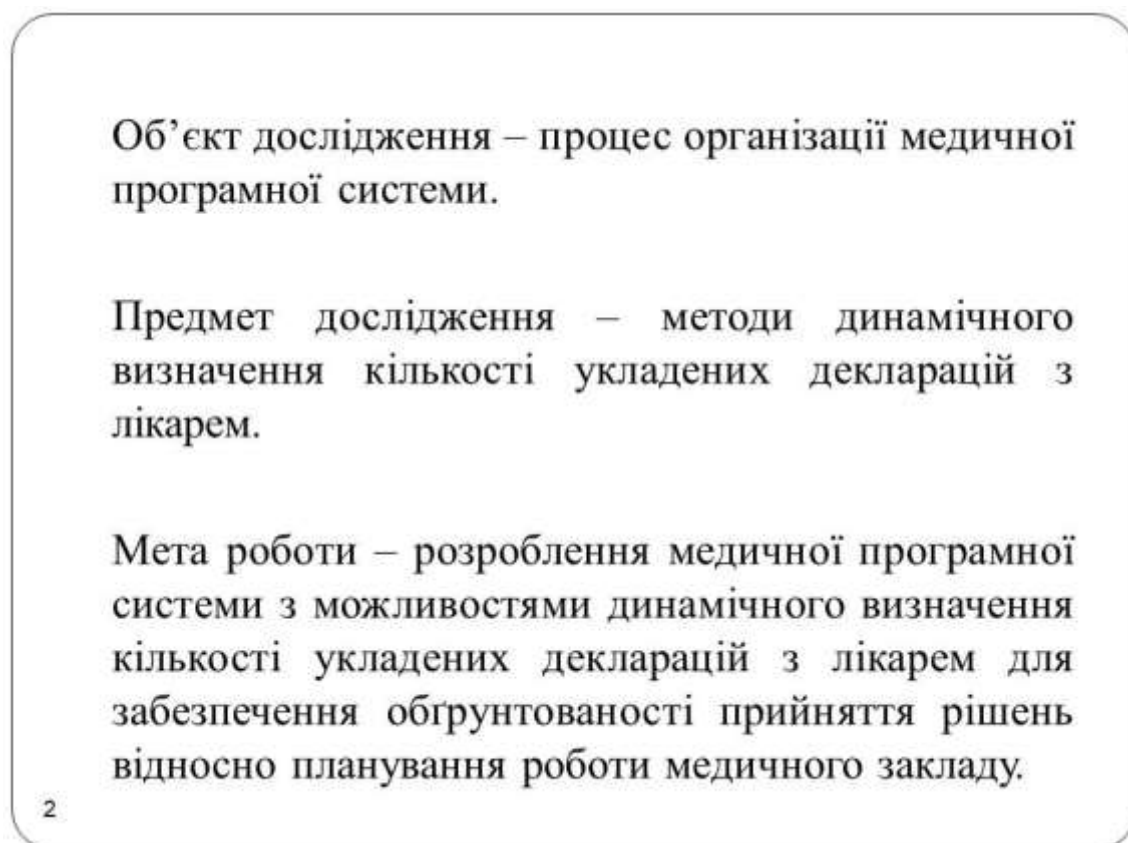


Рисунок В.2 – Слайд 2

Завдання

- ✓ розроблення методу динамічного визначення кількості укладених декларацій з лікарем;
- ✓ розробка технічного завдання;
- ✓ проєктування програмного забезпечення;
- ✓ розробка програмного забезпечення організації медичного сервісу;
- ✓ тестування та відлагодження програми;
- ✓ економічне обґрунтування доцільності розробки.

3

Рисунок В.3 – Слайд 3

Метод динамічного визначення кількості укладених декларацій з лікарем

Етап 1. Витягання даних з бази даних, акумуляція статистики щодо кожного лікаря в системі і формування вибірки даних.

Кожен екземпляр вибірки даних представлений сумарною кількістю декларацій d_j^i , які були укладені з заданим лікарем i за заданий базовий період j .

Кожен екземпляр вибірки має складатися з M значень $d_j^i, j = r, \dots, r + M - 1$ в якості вхідних та $d_{r+M}^i, d_{r+M+1}^i, \dots, d_{r+M+p}^i$ в якості вихідної ознаки.

Етап 2. Створити модель прогнозування кількості укладених декларацій з лікарем на основі використання нейронної мережі з архітектурою LSTM.

Етап 3. Виконати навчання моделі.

Вся навчальна вибірка, отримана в результаті виконання першого етапу, має бути розбита на навчальну і тестову вибірки. Навчальна вибірка складає 80 %.

Етап 4. Виконати тестування моделі.

На даному етапі має бути прийняте рішення, чи отримана в результаті модель є практично застосовною.

Етап 5. Виконати оновлення кількості очікуваних для укладення декларацій з лікарем на наступні базові періоди на основі поданих значень про попередні періоди.

Відповідно для цього реалізується переобчислення кількості декларацій, укладених за місяць.

4

Рисунок В.4 – Слайд 4

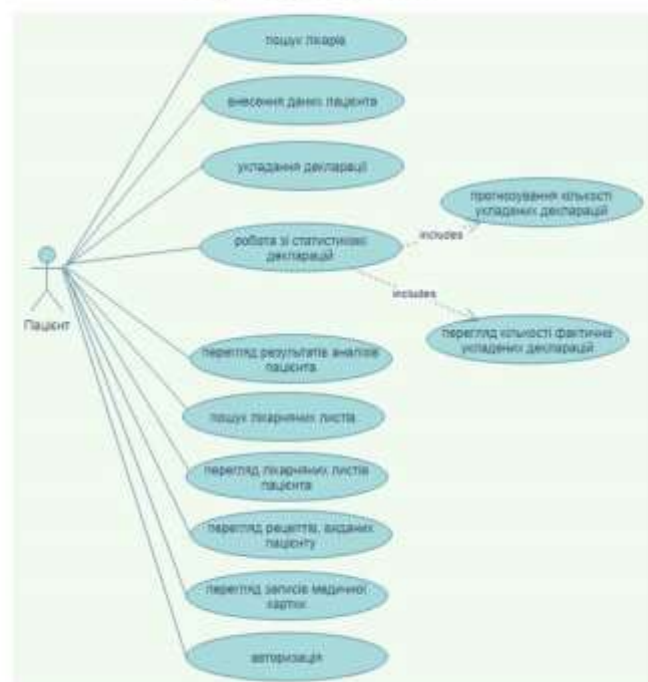
Вибір мови програмування

Критерій	C#	C++
Тип мови	Компільована в байт-код	Компільована
Парадигма об'єктно-орієнтованого програмування	+	+
Робота з пам'яттю	Автоматично	Вручну
Створення застосунків для настільних систем	Зручне	Вимагає більших витрат порівняно з C#

5

Рисунок В.5 – Слайд 5

Діаграма прецедентів роботи пацієнта з програмою



6

Рисунок В.6 – Слайд 6

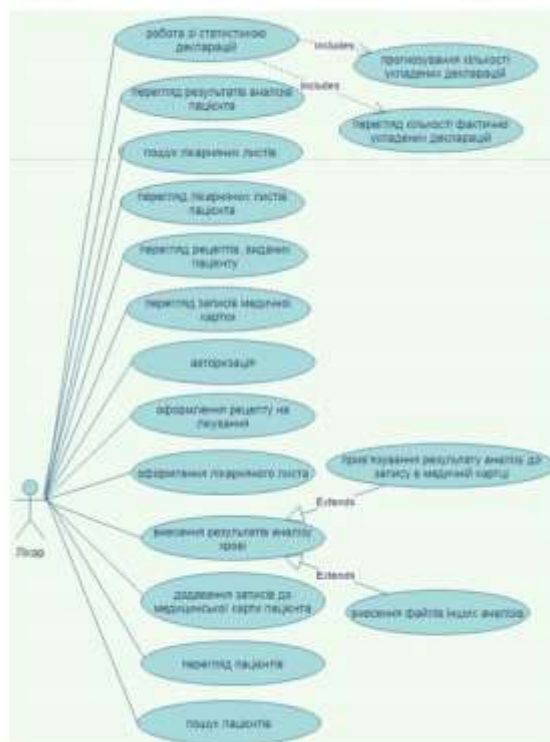


Рисунок В.7 – Слайд 7



Рисунок В.8 – Слайд 8

Структурна схема програмного проєкту

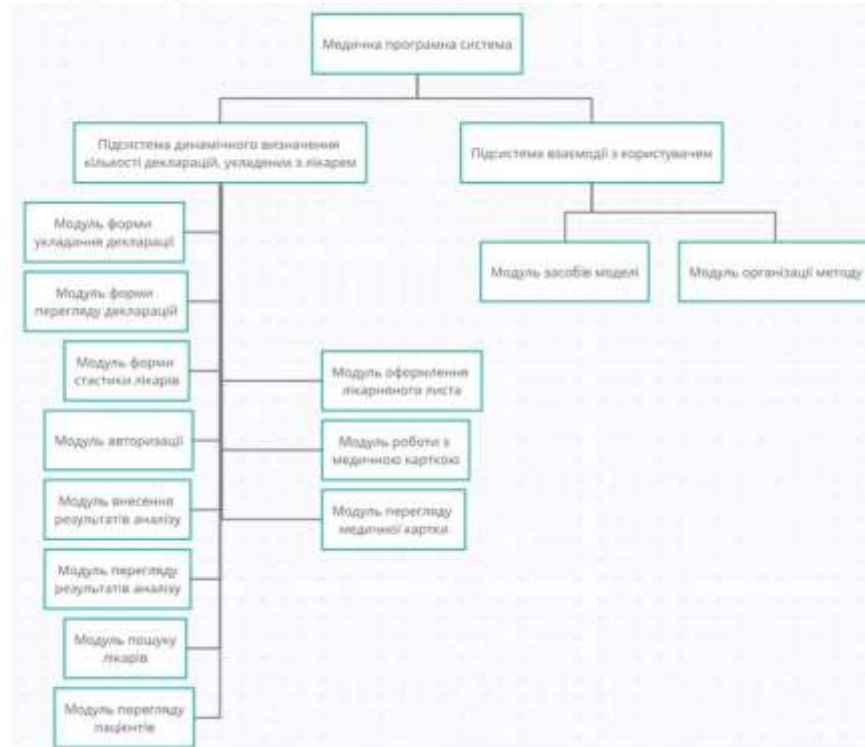


Рисунок В.9 – Слайд 9

Функціональна схема програми

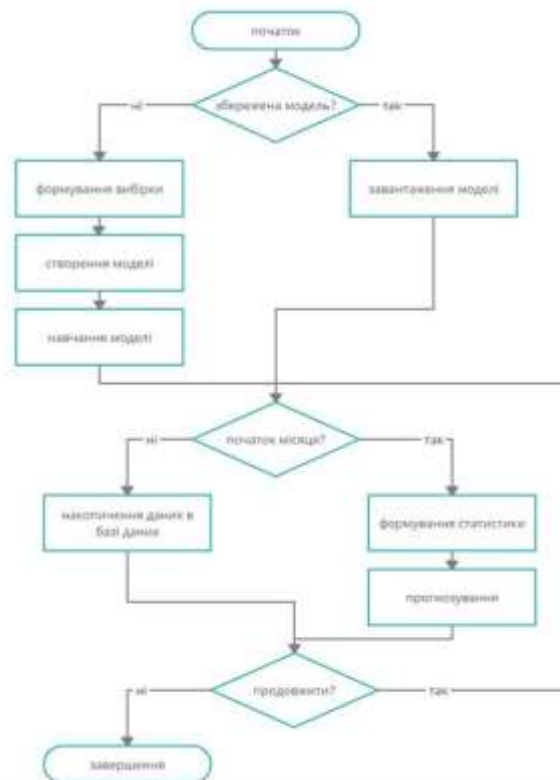


Рисунок В.10 – Слайд 10

Робота з деклараціями

Сімейний лікар Зоренко Іван Миколайович

Прізвище пацієнта: Ім'я: По батькові:

Номер телефону:

Пацієнт	Телефон	Укладено
Архипов Руслан Петрович	+380505326281	15.02.2020
Асмаков Іван Сергійович	+380501346219	18.10.2019
Зоренко Микола Олександрович	+380996021428	21.03.2020
Пут Анатолій Михайлович	+380671692438	18.04.2020
Венедиктов Кирило Андрійович	+380674278291	15.02.2020
Віра Гарига Анатолій	+380504632910	20.06.2020
Зоренко Олександр Германович	+380957218439	02.03.2020
Вустий Іван Миколайович	+380977215639	05.07.2020
Зоренко Софія Іванович	+380673026528	02.09.2020
Гудінов Анатолій Сергійович	+380678261927	01.02.2020

Пацієнт Колесін Артур Вікторович

Прізвище лікаря: Ім'я: По батькові:

Спеціалізація: Медична установа:

Лікар	Медична установа
Зоренко Іван Миколайович	Запорізька обласна лікарня
Зоренко Ганна Пилипівна	Запорізька міська лікарня № 5

Рисунок В.11 – Слайд 11

Робота над даними пацієнта

Медична картка

Пацієнт Мисюков Петро Андрійович

Сімейний лікар Іваненко Леонід Григорович

Інтервал часу з: по:

Записи

Дата
04.12.2020
29.11.2020
01.11.2020

Діагнози

Запальний легітис

Скарги

Віль у грудях під час дихання, важке дихання, слабкість

Лікування

Рекомендована комбінована терапія: томографія легень. Обсяг ліжковий частіше п'яти. Карбоцистеїн тричі на добу по 15 мл.

Лікарняний

Сімейний лікар Афталович Семен Васильович

Пацієнт Семених Семен Костянтинович

Дата початку

1 грудня 2020 р.

Пн	Вт	Ср	Чт	Пт	Сб	Нд
01	02	03	04	05	06	07
08	09	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30	31	1	2	3	4
5	6	7	8	9	10	11

10.12.2020

Дата завершення

1 грудня 2020 р.

Пн	Вт	Ср	Чт	Пт	Сб	Нд
01	02	03	04	05	06	07
08	09	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30	31	1	2	3	4
5	6	7	8	9	10	11

10.12.2020

Тип:

Діагноз:

12

Рисунок В.12 – Слайд 12

Статистика і результати прогнозування

Сімейний лікар Афталонович Семен Йосипович Додому

Рік Показати

Кількість декларацій

Дата	Кількість
01.2020	187 (+2)
02.2020	189 (+2)
03.2020	195 (+6)
04.2020	199 (+4)
05.2020	200 (+1)
06.2020	200
07.2020	202 (+2)
08.2020	205 (+3)
09.2020	209 (+4)
10.2020	210 (+1)
11.2020	212 (+2)
12.2020	~215 (+3)

Пацієнти

13

Рисунок В.13 – Слайд 13

Висновки

У роботі розглянуто проблему організації медичної програмної системи.

Розроблено програмне забезпечення, що дозволяє виконувати оформлення декларацій з лікарем, отримувати лікарям та пацієнтам єдиний доступ до медичної інформації, вести електронні медичні картки.

Наукова новизна роботи полягає в запропонованому методі динамічного визначення кількості укладених декларацій з лікарем, що характеризується використанням нейронної мережі архітектури LSTM для прогнозування кількості декларацій на майбутню послідовність періодів часу, що дозволяє менеджменту лікарні приймати рішення про планування своєї діяльності.

14

Рисунок В.14 – Слайд 14