

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет комп'ютерних наук і технологій

(повне найменування факультету)

Кафедра програмних засобів

(повне найменування кафедри)

Пояснювальна записка

до дипломного проєкту (роботи)

магістр

(ступінь вищої освіти)

на тему ДОСЛІДЖЕННЯ ТА ПРОГРАМНА РЕАЛІЗАЦІЯ ПЛАТФОРМИ
ДЛЯ ВЕДЕННЯ ЩОДЕННИКА ТА УПРАВЛІННЯ ЗАВДАННЯМИ
RESEARCH AND SOFTWARE IMPLEMENTATION OF
A PLATFORM FOR DIARY AND TASK MANAGEMENT

Виконав(ла): студент(ка) 2 курсу, групи КНТ-114м
Спеціальності 121 Інженерія програмного
(код і найменування спеціальності)

забезпечення

Освітня програма (спеціалізація)

Інженерія програмного забезпечення

ЛЮБАС С.О.

(ПРІЗВИЩЕ та ініціали)

Керівник КОЛПАКОВА Т.О.

(ПРІЗВИЩЕ та ініціали)

Рецензент ІЛЬЯШЕНКО М.Б.

(ПРІЗВИЩЕ та ініціали)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

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

ЗАТВЕРДЖУЮ

Завідувач кафедри ПЗ, д.т.н, проф.
Сергій СУББОТІН
“ ” 2025 року

З А В Д А Н Н Я

НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА(КИ)

ЛЮБАСА Сергія Олексійовича

(ПРИЗВИЩЕ, ім'я, по батькові)

1. Тема проєкту (роботи) Дослідження та програмна реалізація платформи для ведення щоденника та управління завданнями. Research and Software Implementation of a Platform for Diary and Task Management

керівник проєкту (роботи) к.т.н., доцент, КОЛПАКОВА Тетяна Олексіївна,
(науковий ступінь, вчене звання, ПРИЗВИЩЕ, ім'я, по батькові)

затверджені наказом закладу вищої освіти від “ 30 ” вересня 2025 року № 447

2. Строк подання студентом проєкту (роботи) 02 грудня 2025 року

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

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1. Аналіз предметної області. 2. Матеріали і методи. 3. Опис програми. 4. Експлуатація, тестування та експериментальне дослідження програми.

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

6. Консультанти розділів проєкту (роботи)

Розділ	ПРИЗВИЩЕ, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-4 Основна частина	КОЛПАКОВА Т.О., доцент		
Нормоконтроль	ДЕЙНЕГА Л.Ю., ст. викладач		

7. Дата видачі завдання “ 30 ” вересня 2025 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1	Постановка завдання роботи.	1 тиждень	Завдання, ТЗ
2	Аналіз предметної області.	1 тиждень	Розділ 1
3	Вибір мови програмування та інших технологій розробки.	2 тиждень	Розділ 2
4	Розробка структури програми.	3 тиждень	Розділ 3
5	Розробка програми.	4-7 тижні	Розділи 3,4
6	Тестування та експериментальне дослідження програмного забезпечення.	8 тиждень	Розділ 4
7	Оформлення пояснювальної записки та документів до неї.	9-10 тижні	Додатки
8	Нормоконтроль та рецензування.	11 тиждень	
9	Захист роботи.	12 тиждень	

Студент(ка)

(підпис) Сергій ЛЮБАС
(Ім'я ПРИЗВИЩЕ)

Керівник проєкту (роботи)

(підпис) Тетяна КОЛПАКОВА
(Ім'я ПРИЗВИЩЕ)

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи магістра:
88 с., 3 табл., 39 рис., 3 дод., 17 джерел.

PHP, PHPSTORM, ДІАГРАМА, МОВА ПРОГРАМУВАННЯ,
ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ, УПРАВЛІННЯ ЗАВДАННЯМИ,
ЩОДЕННИК.

Об'єкт дослідження – процес розробки програмного забезпечення для ведення щоденника та управління завданнями.

Предмет дослідження – програмні засоби для ведення щоденника та управління завданнями.

Мета роботи – дослідження та розробка програмної платформи для ведення щоденника та управління завданнями.

Матеріали, методи та технічні засоби: мова програмування PHP, середовище розробки PhpStorm.

Результати. Розроблено програмну платформу для ведення щоденника та управління завданнями за допомогою мови програмування PHP та середовища розробки PhpStorm.

Практична цінність роботи полягає у розробці програмного забезпечення для ведення щоденника та управління завданнями.

Висновки. Виконано проєктування програмного забезпечення для ведення щоденника та управління завданнями. Розроблено програмну платформу для ведення щоденника та управління завданнями. Здійснено тестування розробленої програмної платформи для ведення щоденника та управління завданнями.

Галузь використання – планування завдань.

ABSTRACT

Explanatory note to the diploma qualifying work of the master: 88 pages, 3 tables, 39 figures, 3 appendixes, 17 sources.

PHP, PHPSTORM, DIAGRAM, PROGRAMMING LANGUAGE, SOFTWARE, TASK MANAGEMENT, DIARY.

The object of research is the process of developing software for diary and task management.

The subject of the research is software for diary and task management.

The purpose of this work is to develop the software platform for diary and task management.

Materials, methods and technical tools: PHP programming language, PhpStorm development environment.

Results. The software platform for diary and task management using the PHP programming language and the PhpStorm development environment has been developed.

The practical value of the work lies in the development of software platform for diary and task management.

Conclusions. The software platform for diary and task management has been designed. The software platform for diary and task management has been developed. The developed software platform for diary and task management has been tested.

Scope of use – task planning.

ЗМІСТ

	С.
Перелік скорочень та умовних позначок	8
Вступ	9
1 Аналіз предметної області	11
1.1 Програмне забезпечення для ведення персональних щоденників та управління завданнями	11
1.2 Аналіз програмного забезпечення для ведення персональних щоденників та управління завданнями	19
1.3 Висновки за розділом 1	27
2 Матеріали і методи	29
2.1 Вибір мови програмування	29
2.2 Вибір середовища розробки для створення програмної платформи для ведення щоденника та управління завданнями	31
2.3 Розробка моделі взаємодії користувача з програмним забезпеченням для ведення щоденника та управління завданнями	34
2.4 Висновки за розділом 2	37
3 Опис програми	38
3.1 Структура програмної платформи для ведення щоденника та управління завданнями	38
3.2 Функціонування програмної платформи для ведення щоденника та управління завданнями	41
3.3 Архітектура вебзастосунку програмного забезпечення для ведення щоденника та управління завданнями	46
3.4 Проектування інтерфейсу програмної платформи для ведення щоденника та управління завданнями	48
3.5 Висновки за розділом 3	52
4 Експлуатація, тестування та експериментальне дослідження програми	53
4.1 Призначення й умови застосування програми	53

4.2 Характеристики програми для ведення щоденника та управління завданнями	54
4.3 Інструкція по експлуатації програми.....	55
4.3.1 Звернення до програми.....	55
4.3.2 Вхідні й вихідні дані.....	55
4.4 Виконання програмної платформи для ведення щоденника та управління завданнями	55
4.5 Тестування програмної платформи для ведення щоденника та управління завданнями	60
4.6 Висновки за розділом 4	60
Висновки.....	61
Перелік джерел посилання	64
Додаток А Технічне завдання.....	66
Додаток Б Фрагмент тексту програми	70
Додаток В Слайди презентації	81

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

MVC – Model-View-Controller;

БД – база даних;

ДК – діаграма класів;

ПЗ – програмне забезпечення;

СУБД – системи управління базами даних;

УЗ – управління завданнями.

ВСТУП

Програмне забезпечення для ведення щоденника та управління завданнями залишається надзвичайно актуальним у сучасному світі, де ефективне планування часу та організація діяльності мають вирішальне значення. Воно допомагає структурувати щоденні справи, ставити пріоритети і контролювати виконання завдань, що сприяє підвищенню продуктивності та зменшенню стресу [1], [2].

Такі програмні інструменти часто інтегруються з іншими сервісами, що полегшує комунікацію і спільну роботу, особливо у віддалених командах. З огляду на зростаючу кількість інформації та швидкість її обробки, програмне забезпечення для планування залишається необхідним засобом для підтримки особистої та професійної організації [3], [4].

Крім того, розробка програмного забезпечення для ведення щоденника та управління завданнями відповідає потребам сучасного ритму життя, який характеризується високою динамічністю та великим обсягом інформації. В умовах постійного зростання навантажень та багатозадачності, такі інструменти стають незамінними для підтримки балансу між роботою, навчанням і особистим життям. Вони сприяють підвищенню самоорганізації та дисципліни користувачів, допомагають уникати проблем, пов'язаних з забуванням важливих справ [5], [6].

Проте деякі програми для ведення персональних щоденників та управління завданнями характеризуються складністю інтерфейсу або мати надлишкові функції, які ускладнюють швидке освоєння і викликають перевантаження користувача, особливо якщо програму використовують люди без технічного досвіду. По-друге, залежність від Інтернет з'єднання для синхронізації та збереження даних може створювати проблеми у разі нестабільного доступу до мережі. Також існує ризик втрати конфіденційної інформації через недостатній рівень захисту або витік даних. Важливим недоліком може бути обмежена сумісність з іншими застосунками або

платформами, що ускладнює інтеграцію у вже існуючі робочі процеси. Крім того, деякі програми можуть не враховувати індивідуальні потреби користувачів, пропонуючи надто універсальні рішення без гнучкості налаштувань. У певних випадках вартість підписки або ліцензії на програмне забезпечення може бути перешкодою для широкого використання. Таким чином, незважаючи на численні переваги, ці недоліки можуть впливати на загальну ефективність і зручність використання такого програмного забезпечення [1]-[6].

Тому актуальною є розробка програмного забезпечення для ведення щоденника та управління завданнями. У дипломній кваліфікаційній роботі магістра розв'язується актуальне завдання розробки та дослідження програмної платформи для ведення щоденника та управління завданнями.

Для досягнення поставленої мети у кваліфікаційній роботі магістра необхідно розв'язати такі задачі:

- виконати аналіз предметної області та програмних засобів для ведення персональних щоденників та управління завданнями;
- розробити модель взаємодії користувача з програмним забезпеченням для ведення щоденника та управління завданнями;
- здійснити проєктування програмного забезпечення для ведення щоденника та управління завданнями;
- створити програмної платформи для ведення щоденника та управління завданнями;
- виконати тестування розробленої програмної платформи для ведення щоденника та управління завданнями.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Програмне забезпечення для ведення персональних щоденників та управління завданнями

Програмне забезпечення для ведення щоденника та управління завданнями слугує цифровим середовищем, яке дозволяє організовувати особисті справи, контролювати поточні та майбутні події, а також зберігати нотатки й важливі записи у впорядкованому вигляді. Такий інструмент об'єднує різні функції, що допомагають стежити за виконанням планів і водночас фіксувати деталі, які можуть знадобитися в роботі чи повсякденному житті. Завдяки подібним рішенням зникає необхідність вести окремі паперові щоденники чи створювати нагадування в різних додатках, адже вся інформація збирається в одному місці й залишається доступною у зручному форматі [1]-[3].

Зазвичай подібні програмні платформи дозволяють створювати записи про зустрічі, події, завдання або цілі з можливістю вказувати деталі й встановлювати часові рамки для кожного пункту. При цьому багато сучасних рішень передбачають синхронізацію з календарями чи іншими сервісами, що значно спрощує планування і дозволяє уникати дублювання інформації. Окрему увагу приділено гнучкості в управлінні даними: користувачі можуть редагувати записи, переносити їх у часі, об'єднувати з контактами чи групувати за темами [4], [5].

Ще однією важливою властивістю такого програмного забезпечення є система сповіщень, яка дає змогу своєчасно нагадувати про кінцеві терміни виконання або важливі справи, мінімізуючи ймовірність пропуску подій. Для спільної роботи часто реалізовано функції взаємодії між кількома користувачами, що відкриває можливості для планування командних завдань чи спільного внесення змін у загальний розклад [2]-[4].

Завдяки поєднанню гнучкого налаштування, простоти доступу й зручного інтерфейсу таке програмне рішення допомагає ефективніше

використовувати час, впорядковувати справи та формувати чітку структуру повсякденного чи професійного життя. Усе це робить програмне забезпечення для ведення щоденника та управління завданнями корисним інструментом як для особистого користування, так і для роботи у групах, де важливо підтримувати чіткість і злагодженість дій [5], [6].

З метою виявлення та аналізу вимог до розробки відповідного програмного забезпечення (ПЗ) проаналізуємо процес ведення щоденника та управління завданнями [1]-[6].

Сутність організації записів у щоденнику за допомогою відповідного програмного забезпечення полягає в упорядкуванні власного часу так, щоб усі справи розміщувалися у потрібній послідовності й не створювали перевантаження чи хаосу. Такий підхід дозволяє і окремим користувачам, і цілим колективам грамотно формувати щоденні та довгострокові плани, завдяки чому досягається рівновага між роботою, особистими обов'язками й періодами відпочинку. Особливо у робочій сфері грамотне ведення розкладу перетворюється з рутинного записування подій на важливий важіль керування ефективністю та дисципліною команди [1], [2].

Коли календар за допомогою відповідного програмного забезпечення використовується свідомо, результатом стає значно продуктивніша організація часу. Чіткий розподіл справ запобігає плутанині та невиправданим затримкам, а також знижує напруженість, бо власник розкладу завжди розуміє, коли і що необхідно виконати. Крім того, таке планування допомагає уникати виснаження й дозволяє збалансувати завантаженість так, щоб залишався простір для особистих справ або відпочинку [1], [2].

Реалізувати такі підходи допомагають сучасні календарні сервіси та відповідне програмне забезпечення, що забезпечують швидкий доступ до записів із різних пристроїв, дозволяють розмічати записи мітками й кольорами, ставити нагадування та тримати всі важливі справи під рукою. Коли в офісі використовується централізований електронний розклад, він стає спільним орієнтиром для всієї команди та полегшує колективну роботу [1], [2].

Основою будь-якої подібної програмної системи стає єдиний календар, у якому відображаються як особисті, так і спільні обов'язки. Додавання командних записів допомагає уникнути накладок і зайвих непорозумінь між працівниками. Грамотне виставлення пріоритетів і розподіл часу між важливими та другорядними справами дозволяє зосередитися на головному. Такий підхід передбачає виділення певних відрізків часу для різних типів діяльності та уникнення зайвого відволікання [1], [2].

Особливу увагу вартує приділяти так званим буферним проміжкам, які закладаються між запланованими справами. Вони захищають від ризику потрапити в ситуацію, коли кілька справ стикаються між собою, не залишаючи простору на перепочинок або вирішення раптових питань. Гнучкий розклад завжди враховує, що не все йде за сценарієм, і залишає місце для швидкого реагування [1], [2].

Корисно час від часу переглядати весь розклад за допомогою відповідного програмного забезпечення, оцінювати, наскільки він відповідає поточним завданням, і вносити потрібні зміни. Завдяки цьому календар залишається дієвим інструментом контролю часу й не перетворюється на формальний список, який втрачає свою практичну користь [1], [2].

Організація ділового розпорядку керівника складається не лише зі звичайних записів подій, а й охоплює значно ширші обов'язки, що вимагають уваги до кожної дрібниці. Ведення щоденника передбачає постійне узгодження численних зустрічей, перевірку термінів, контроль за змінами у планах та миттєве коригування всього розкладу, коли виникають нові завдання чи обставини. Асистент, який відповідає за такі процеси, мусить подбати про те, щоб навіть при раптових змінах розпорядок дня лишався логічним, а часових проміжків вистачало на дорогу, обговорення чи короткий перепочинок [3], [4].

Для безперебійного виконання обов'язків важливо з'єднувати електронну пошту з календарем, адже чимало завдань делегується іншим і потребує контролю. Така синхронізація дає змогу швидко стежити, як ідуть

справи з виконанням доручених пунктів. Крім того, важливо забезпечити, щоб кожна робоча чи ділова зустріч відповідала поставленій меті та була ретельно підготовлена — тільки так вдається уникати марних витрат часу [3], [4].

Той, хто відповідає за ці обов'язки, має вміти користуватися найновішими програмними інструментами планування. Це можуть бути як сучасні сервіси для складання розкладів, так і допоміжні засоби для автоматичного надсилання нагадувань. Продумано встановлені сигнали нагадувань значно спрощують контроль, адже про важливі події не дозволяють забути push-повідомлення чи електронні листи. Дрібниці мають велике значення — саме через них легко допустити промахи, яких завжди потрібно уникати [3], [4].

Вчасне інформування про зміни та підтвердження зустрічей гарантує, що ніякі деталі не загубляться. Якщо виникають сумніви, чи варто витрачати час на певну подію, доцільно дізнатися всі обставини заздалегідь. Це допомагає гнучко налаштувати графік і не допустити конфлікту інтересів. Також важливо тримати постійний діалог із керівником, щоб залишатися обізнаним у його вподобаннях і швидко коригувати плани відповідно до нових обставин [3], [4].

Дуже зручно виділяти події різними кольорами за допомогою відповідного програмного забезпечення. Такий візуальний поділ робить план більш зрозумілим, дозволяючи з першого погляду відрізнити типи зустрічей — від внутрішніх нарад до переговорів із зовнішніми партнерами. Це не лише зручно, а й знижує ризик помилок у щоденних планах [3], [4].

Якщо діяльність передбачає регулярні подорожі, важливо брати до уваги часові пояси. Запис цих деталей допомагає уникати непорозумінь при організації дзвінків чи зустрічей з колегами, які працюють в іншій часовій зоні. Коли цей момент враховується, графік залишається реалістичним і точним [3], [4].

Окрім календаря, корисно завжди мати під рукою окремий список справ. Такий перелік дозволяє відстежувати дрібні завдання та поступово їх

закривати. Це може бути цифровий додаток чи простий паперовий блокнот — головне, щоб нічого не губилося й було завжди під рукою [3], [4].

Варто фіксувати всі кінцеві терміни виконання, які мають критичне значення для проєктів або роботи всієї команди. Як тільки про них стає відомо, доцільно одразу внести їх до розкладу й тримати в полі зору. Це допоможе не допустити накладання завдань і уникнути перевантаження у найвідповідальніший момент [3], [4].

Щоб уникнути перенасичення розкладу, важливо передбачати відрізки часу, вільні від зустрічей. Вони потрібні для виконання завдань, які не терплять відкладень. Такі робочі проміжки мають бути захищені від раптових змін, щоб справи рухались без затримок [3], [4].

Корисно підходити до планування зустрічей максимально точно. Іноді замість години вистачає коротшого проміжку, тож краще залишати невеликий запас часу, щоб уникнути затягування. Грамотне складання послідовності зустрічей допомагає уникнути поспіху й дозволяє зосередитися на найважливішому [5], [6].

В умовах динамічного ритму сучасності раціональне планування часу стає необхідністю для тих, хто прагне тримати під контролем власний день, знижувати рівень напруги й не втрачати з поля зору ключові цілі. Незалежно від сфери діяльності чи ролі, вміння вести особистий або робочий щоденник допомагає врівноважити всі обов'язки й знайти час для справ, які дійсно важливі. Саме тому докладне розуміння підходів до ведення щоденника дозволяє ефективніше розподіляти навантаження і робити кожен день більш передбачуваним та продуктивним [5], [6].

Пошук відповідного програмного інструмента для фіксації планів — один із основних етапів упорядкування розкладу. Серед варіантів можна знайти як класичні паперові блокноти, так і цифрові платформи або багатофункціональні програми. Кожен формат має власні плюси й недоліки. Паперові варіанти створюють відчуття фізичного контакту й можуть бути стилістично приємними, але поступаються мобільності та швидкості

редагування. Онлайн календарі дають змогу синхронізувати дані між різними пристроями й легко ділитися ними, проте потребують наявності інтернету та уваги до кібербезпеки. Сучасні застосунки часто об'єднують можливості різних рішень: дозволяють планувати, фіксувати нагадування й навіть вести облік досягнутих звичок, проте можуть бути перевантажені зайвими опціями, що іноді заважає фокусуватись на головному [5], [6].

Коли обрано спосіб ведення записів, наступним кроком стає усвідомлення того, що справді потребує уваги в першу чергу. Для цього важливо чітко визначати, які завдання мають найбільшу вагу та як вони співвідносяться з життєвими цінностями. Обґрунтоване формулювання цілей допомагає не втрачати фокус і не розпорошувати енергію на другорядне. Коли завдання набувають вимірюваної форми й мають конкретні орієнтири, простіше оцінювати поступ і своєчасно вносити зміни [5], [6].

Зрозуміле структурування щоденного графіка починається з того, що для найважливіших подій відводиться чітко визначений проміжок часу. Розклади часто складаються не лише з робочих зборів чи термінів виконання проєктів, а й з особистих подій, зустрічей, моментів для відпочинку або занять, які підтримують фізичне й психічне здоров'я. Такі фіксовані часові блоки захищають важливе від випадкових змін і допомагають уникати хаотичного перенесення справ [5], [6].

Завдання, які здаються надто масштабними чи заплутаними, зазвичай доцільно розбивати на невеликі кроки. Це дозволяє уникати відкладання й поступово рухатися вперед, не відчуваючи перевантаження. Для прикладу, при підготовці звіту можна спершу опрацювати інформацію, скласти структуру, написати вступні розділи й так далі, крок за кроком просуваючись до фінального варіанта [5], [6].

Щоб уникати перенапруги, варто передбачати у розкладі паузи. Короткі перерви протягом дня дають змогу перезавантажити увагу й залишатись зібраним, а проміжки між подіями допомагають впоратися із затримками або непередбаченими завданнями. Гнучкість теж відіграє

неабияку роль: навіть найбільш продуманий графік не застрахований від раптових змін, тож здатність швидко коригувати час і розставляти нові акценти є запорукою стійкості до стресу [5], [6].

Насамкінець, реалістична оцінка власних можливостей запобігає розчаруванню і перенавантаженню. Уміння визнавати, скільки справ реально виконати за день, а що можна перенести, зберігає баланс і допомагає не втрачати контроль навіть у найбільш насичені періоди. Навіть якщо деякі зустрічі або плани скасовуються, цей час завжди можна використати на інші, не менш важливі справи [5], [6].

У повсякденному потоці справ нагадування й різноманітні сповіщення відіграють ключову роль у тому, щоб не втрачати з уваги важливі моменти та уникати ситуацій, коли терміни раптово підходять до завершення без підготовки. Коли у записах фіксуються сигнали про майбутні зустрічі, терміни виконання чи необхідність виконати завдання, підтримується впорядкованість розкладу, а кожен наступний крок планується з урахуванням актуальної ситуації. Попередньо задані сигнали дозволяють не забувати про суттєві події, нагадуючи заздалегідь, щоб була змога зреагувати вчасно [5], [6].

Ведення будь-якого календаря не обмежується одноразовим заповненням — це процес, що потребує постійного оновлення. Щоденні чи щотижневі перевірки допомагають відстежувати, що вже вдалося зробити, а що потребує корекції або додаткової уваги. Регулярне оновлення записів підтримує точність у відображенні обов'язків та пріоритетів, а також полегшує оцінку дій і планування наступних кроків. Так зберігається чітке уявлення про заплановане та фактично виконане [5], [6].

Часто трапляються справи, які можна довірити іншим — це доцільно робити, щоб звільнити час для більш значущих завдань. Передача деяких питань іншим людям або залучення зовнішніх виконавців відкриває можливість зосередитися на тому, що справді потребує особистої уваги. Таким чином знижується ризик перевантаження й підвищується якість виконання ключових обов'язків [5], [6].

Іноді варто свідомо відмовлятися від зайвих ініціатив, які не мають значної користі чи суперечать головним напрямкам діяльності. Це непросто, проте здатність відмовлятися від другорядного дозволяє тримати час під контролем і не втрачати ресурсів на справи, що не ведуть до поставлених результатів [5], [6].

Підвищити ефективність роботи допомагають і різні підходи до організації часу. Застосування технік із чітким розподілом проміжків для кожної дії, робота короткими серіями із перервами або розподіл завдань за рівнем нагальності та важливості — усе це допомагає працювати більш усвідомлено й з меншими втратами ресурсів. Кожен може випробувати кілька варіантів і залишити ті, що найкраще підходять до особистого ритму [5], [6].

Водночас важливо подбати про усунення всього, що може відволікати під час виконання складних або об'ємних завдань. Іноді достатньо прибрати зайві сповіщення, обмежити доступ до ресурсів, які не мають стосунку до справи, або забезпечити тишу й усамітнення. Створення умов, де ніщо не відволікає, позитивно впливає на швидкість і якість роботи [5], [6].

Щоб краще зрозуміти, як насправді витрачаються години протягом дня, корисно вести облік витраченого часу. Таке спостереження допомагає відстежити шаблони поведінки, знайти точки, де час іде не туди, куди треба, і своєчасно внести правки у звички чи розклад [5], [6].

Визначено, що програмне забезпечення для ведення щоденника та управління завданнями слугує цифровим середовищем, яке дозволяє організовувати особисті справи, контролювати поточні та майбутні події, а також зберігати нотатки й важливі записи у впорядкованому вигляді. Такий інструмент об'єднує різні функції, що допомагають стежити за виконанням планів і водночас фіксувати деталі, які можуть знадобитися в роботі чи повсякденному житті. Завдяки подібним програмним рішенням зникає необхідність вести окремі паперові щоденники чи створювати нагадування в різних додатках, адже вся інформація збирається в одному місці й залишається доступною у зручному форматі.

1.2 Аналіз програмного забезпечення для ведення персональних щоденників та управління завданнями

Проаналізуємо програмне забезпечення для ведення щоденників та управління завданнями [7]-[13].

Програмне забезпечення Clockify (рис. 1.1) являє собою універсальний інструмент, що дозволяє організувати облік особистого часу, керувати поточними справами та планувати робочі або особисті події завдяки гнучкому поєднанню з іншими сервісами. Його використання передбачає повноцінну інтеграцію з календарями, що полегшує перенесення вже запланованих зустрічей чи нагадувань і підтримує порядок у щоденних графіках. Завдяки функціоналу платформи можна не просто фіксувати заплановане, а й доповнювати записи деталями, вносити описи та коментарі, об'єднувати записи з різними учасниками чи ділитися підготовленими матеріалами із співробітниками або близькими [7], [11].

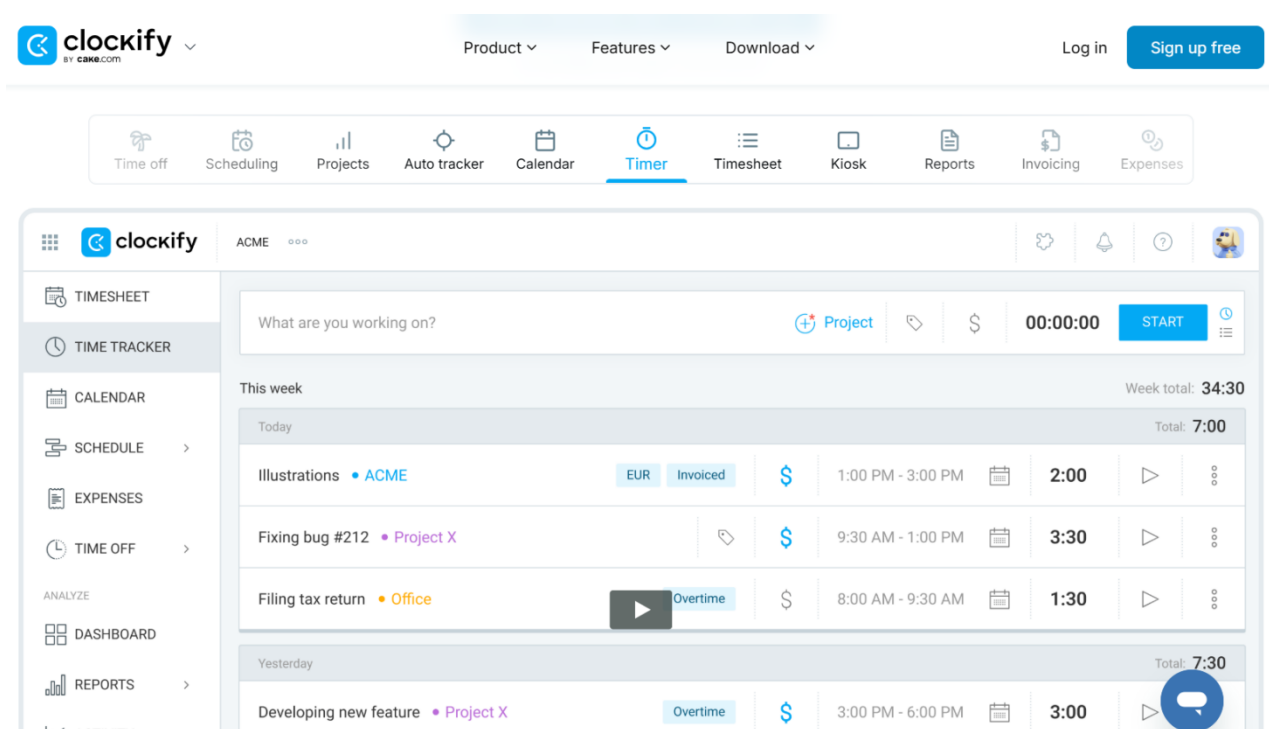


Рисунок 1.1 – Програмне забезпечення Clockify [11]

Кожне нове завдання чи подія всередині системи має бути гнучко налаштоване: у користувача залишається можливість додавати іншу людину до виконання певного плану або надсилати запрошення на спільну участь у заході. Важливою перевагою є нагадування, що дозволяють не пропускати важливе й своєчасно повертатися до актуальних завдань. Окрім роботи з календарями, програма підтримує просте управління контентом, що робить її корисною як у професійному середовищі для командної взаємодії, так і в індивідуальному користуванні. Продукт зберігає свою функціональність незалежно від пристрою, з якого здійснюється доступ, що означає можливість працювати в браузері або через мобільний додаток, не втрачаючи даних і прогресу. Усе це робить Clockify практичним помічником для тих, хто прагне впорядкувати розклад, стежити за виконанням поставлених завдань і ефективно розподіляти час протягом дня чи тижня [7], [11].

Основними характеристиками програмного забезпечення Clockify є такі [7], [11]:

- інтеграція з календарями: програмне забезпечення Clockify дозволяє синхронізувати події та завдання з популярними календарними сервісами, такими як Google Calendar чи Apple Calendar [7], [11];

- створення нотаток і завдань: програмне забезпечення Clockify забезпечує можливість фіксувати події, завдання та супроводжувати їх короткими коментарями [7], [11];

- налаштування нагадувань: програмне забезпечення Clockify підтримує функцію встановлення сповіщень для запланованих справ у підключених календарях [7], [11];

- спільна робота: програмне забезпечення Clockify дає змогу додавати інших користувачів до завдань і подій для командної взаємодії [7], [11];

- можливість поширення: програмне забезпечення Clockify дозволяє позначати власний контент для зручного обміну інформацією користувача зі своїми колегами чи друзями [7], [11];

– кросплатформенність: доступний через браузер та мобільні застосунки, що робить програмне забезпечення Clockify універсальним для користувачів різних пристроїв [7], [11].

Серед недоліків програмного забезпечення Clockify можна відзначити обмеження у функціоналі деяких розширених можливостей, що можуть вимагати переходу на платні версії, а також залежність від стабільного Інтернет з'єднання для повної синхронізації між пристроями. Іноді користувачі звертають увагу на складність освоєння всіх функцій одразу, особливо якщо потрібно налаштувати інтеграції з кількома календарями чи додатковими сервісами. Деяким може не вистачати більш гнучких параметрів персоналізації інтерфейсу, що ускладнює налаштування застосунку під індивідуальні вимоги чи специфіку роботи команди [7], [11].

Програмне забезпечення Habitica (рис. 1.2) представляє собою інструмент для організації особистих завдань і формування корисних звичок, який поєднує щоденне планування зі стилістикою гри [7], [12].

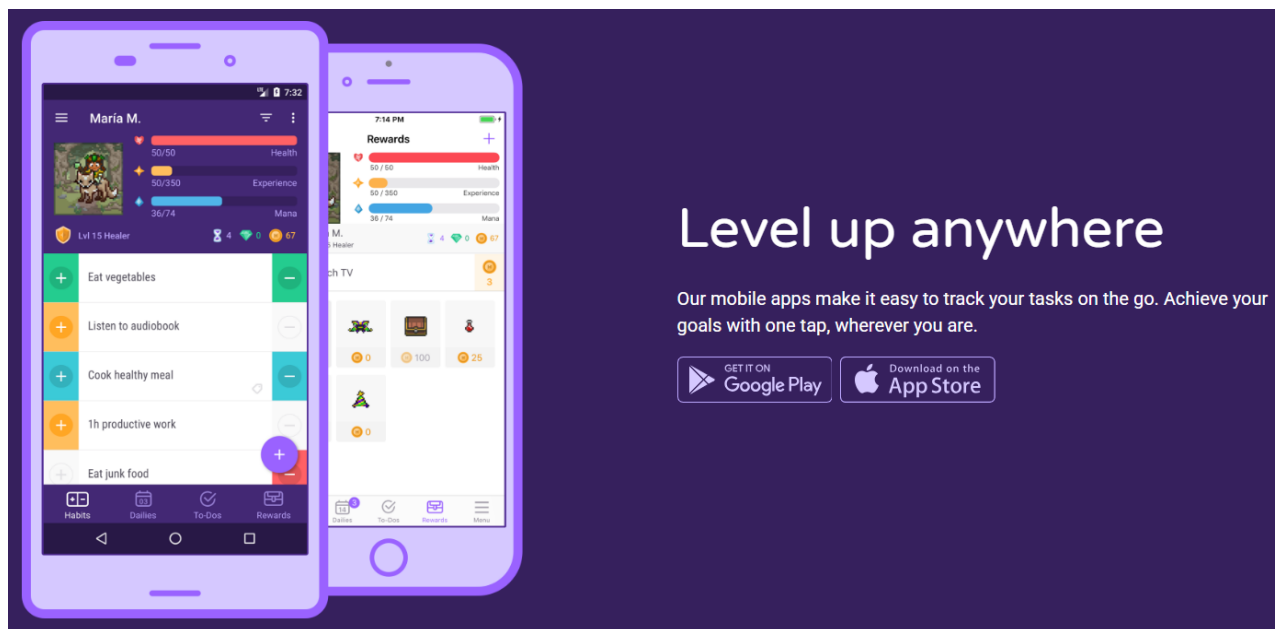


Рисунок 1.2 – Програмне забезпечення Habitica [12]

Завдяки особливому підходу платформа Habitica мотивує користувача стежити за виконанням завдань не лише через звичні списки чи нагадування, а

й через систему досягнень, рівнів і винагород. В основі роботи платформи Habitica лежить принцип, коли щоденні справи, великі цілі чи звички перетворюються на завдання всередині фантазійного середовища, де кожен учасник створює власного персонажа й прокачує його, виконуючи звичайні справи з реального життя [7], [12].

Habitica працює через браузер і мобільні додатки, тому всі дані залишаються доступними, незалежно від того, з якого пристрою користувач заходить. Паралельно можна налаштувати нагадування, щоб не забувати про регулярні завдання, і стежити за прогресом у зручній візуальній формі. Такий формат робить управління звичками й плануванням не нудною рутиною, а своєрідною грою, де результат у щоденнику безпосередньо впливає на розвиток користувача. Усе це перетворює програму Habitica на незвичайний інструмент, що поєднує дисципліну з елементами розваги та дозволяє організувати особистий розклад і контроль завдань у більш легкій і захопливій формі [7], [12].

Програмне забезпечення Habitica має такі особливості [7], [12]:

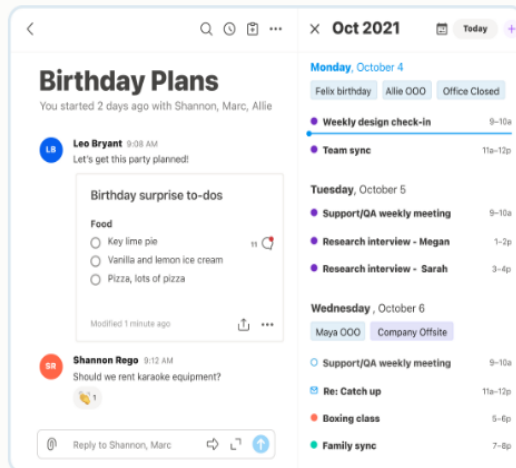
- ігровий підхід: усі завдання та звички перетворюються на частину гри з персонажем, що мотивує виконувати справи [7], [12];
- система нагород: за виконання завдань користувач програмного забезпечення Habitica отримує досвід, предмети чи бонуси, що стимулює підтримувати продуктивність [7], [12];
- гнучке планування: програмне забезпечення Habitica дозволяє створювати щоденні, регулярні або одноразові завдання зі зручним налаштуванням [7], [12];
- командна взаємодія: програмне забезпечення Habitica передбачає можливість об'єднуватися з іншими користувачами для спільних завдань та обміну досягненнями [7], [12];
- багатоплатформений доступ: програмне забезпечення Habitica працює через вебінтерфейс і мобільні версії, що забезпечує синхронізацію даних [7], [12];

- вбудовані нагадування: програмне забезпечення Habitica допомагає не забувати про звички та завдання завдяки системі сповіщень [7], [12];
- рольова персоналізація: користувач програмного забезпечення Habitica розвивається разом із виконанням поставлених цілей [7], [12].

Серед недоліків програмного забезпечення Habitica можна відзначити, що гейміфікований формат підходить не всім і може здаватися надто ігровим для тих, хто шукає суто діловий інструмент планування без зайвих декоративних елементів. Деяким користувачам може не вистачати гнучкості у деталізації завдань або розширених можливостей аналітики прогресу. Також для повноцінного залучення іноді потрібен час, щоб звикнути до специфічного інтерфейсу й зрозуміти всі механіки взаємодії. Інтеграція з іншими календарями чи сторонніми сервісами може здаватися обмеженою, що ускладнює використання Habitica як єдиної системи для комплексного керування всіма аспектами щоденних справ [7], [12].

Програмне забезпечення TwoBird (рис. 1.3) є інструментом, що об'єднує функції поштового клієнта, планувальника завдань і нотатника, створюючи простір, де можна керувати листуванням та одночасно відстежувати справи й записи. Основна ідея програмного забезпечення TwoBird полягає в тому, щоб поєднати електронну пошту з функціями ведення списків справ і особистих нотаток у межах одного інтерфейсу, завдяки чому зникає потреба постійно перемикатися між кількома окремими програмами чи сервісами. Користувач отримує можливість не тільки перевіряти вхідні листи чи відповідати на повідомлення, а й перетворювати листи на завдання або доповнювати їх нагадуваннями, щоб нічого не загубилося серед великої кількості повідомлень. Важливою особливістю програмного забезпечення TwoBird є підтримка спільної роботи: можна залишати коментарі прямо в нотатках чи подіях, обговорювати робочі моменти або уточнювати деталі без потреби створювати окремі листи. [7], [13].

Context is everything.



Constantly switching from an email to a document, or a chat app to a calendar wastes time. Twobird helps by giving you context where you already are.

Edit a note right inside of a conversation. Set a reminder for anything in your inbox, whether it's an email or a quick note. And see your reminders and events side-by-side in the same calendar, not separate lists or tabs.

Рисунок 1.3 – Програмне забезпечення TwoBird [13]

Програмне забезпечення TwoBird значно спрощує командну взаємодію та дозволяє працювати над одним завданням кільком людям одночасно. Окрім пошти та списків справ, у межах платформи зручно вести окремі записи, зберігати важливі думки чи робочі чернетки, що потім можна перетворювати на конкретні кроки у вигляді завдань чи нагадувань [7], [13].

Інтерфейс програмного забезпечення TwoBird створено так, аби користувачі могли легко зосередитися на найголовнішому — для цього реалізовано функції приховування зайвого й зручне групування важливих повідомлень, подій чи нотаток в одному просторі. Такий підхід допомагає зберігати порядок у пошті та водночас тримати під рукою всі ключові справи без потреби відкривати кілька різних вкладок чи застосунків. Використовувати TwoBird можна як через браузер, так і через мобільні пристрої, що робить його доступним у будь-якій ситуації — у дорозі, на роботі чи вдома. Ця платформа орієнтована на тих, хто хоче мати під рукою пошту,

завдання та нотатки в єдиному вікні, де вся інформація завжди впорядкована та готова до роботи без зайвих зусиль [7], [13].

Програмне забезпечення TwoBird має такі особливості [7], [13]:

- інтеграція пошти і завдань: програмне забезпечення TwoBird дозволяє працювати з електронною поштою та списками справ в одному просторі без потреби перемикатися між додатками [7], [13];

- коментування всередині нотаток: програмне забезпечення TwoBird надає можливість обговорювати деталі завдань або нотаток без створення додаткових листів [7], [13];

- створення нагадувань: програмне забезпечення TwoBird дозволяє перетворювати листи або записи на завдання з повідомленнями про дотримання строків [7], [13];

- мінімалістичний інтерфейс: програмне забезпечення TwoBird спрощує зосередження на головному, прибираючи все зайве і групуючи важливе [7], [13];

- багатоплатформенна підтримка: програмне забезпечення TwoBird забезпечує доступ як через браузер, так і через мобільні пристрої, зберігаючи синхронізацію [7], [13];

- гнучке планування: дозволяє створювати списки справ, нотатки і події в одному інтерфейсі [7], [13];

- спільна робота: програмне забезпечення TwoBird підтримує участь кількох людей у роботі над нотатками чи завданнями [7], [13].

Серед недоліків програмного забезпечення TwoBird можна відзначити обмеженість інтеграцій з деякими зовнішніми сервісами, що може стати перешкодою для користувачів, які звикли працювати з великою кількістю сторонніх інструментів. Іноді виникає потреба у більш розширених можливостях сортування або фільтрації пошти й завдань, чого може не вистачати для тих, хто обробляє великий обсяг повідомлень щодня. Деякі користувачі звертають увагу на брак глибоких функцій автоматизації процесів, які є у більш складних платформах для управління проектами.

Також інтерфейс, хоч і простий, може здатися надто мінімалістичним, що не всім підходить для роботи з великою кількістю паралельних справ і нагадувань [7], [13].

Порівняльну характеристику програмного забезпечення для ведення щоденника та управління завданнями наведено у таблиці 1.1.

Таблиця 1.1 – Порівняння програмного забезпечення для ведення щоденника та управління завданнями

Критерій порівняння	Clockify [11]	Habitica [12]	TwoBird [13]
Інтеграція з календарями	+	+—	+
Робота з поштою	+—	—	+
Мотиваційні механізми	+—	+	—
Простота інтерфейсу	+	+—	+—
Зручність для списків справ	+—	+—	+
Можливість нагадувань	+—	+	+

За результатами проведеного аналізу можна зробити висновок, що у наш час існує досить багато програмних засобів для ведення персональних щоденників та управління завданнями. Проте деякі програми для ведення персональних щоденників та управління завданнями характеризуються складністю інтерфейсу або мати надлишкові функції, які ускладнюють швидке освоєння і викликають перевантаження користувача, особливо якщо програму використовують люди без технічного досвіду. По-друге, залежність від Інтернет з'єднання для синхронізації та збереження даних може створювати проблеми у разі нестабільного доступу до мережі. Також існує ризик втрати конфіденційної інформації через недостатній рівень захисту або витік даних. Важливим недоліком може бути обмежена сумісність з іншими застосунками або платформами, що ускладнює інтеграцію у вже існуючі робочі процеси. Крім того, деякі програми можуть не враховувати індивідуальні потреби

користувачів, пропонуючи надто універсальні рішення без гнучкості налаштувань. У певних випадках вартість підписки або ліцензії на програмне забезпечення може бути перешкодою для широкого використання. Таким чином, незважаючи на численні переваги, ці недоліки можуть впливати на загальну ефективність і зручність використання такого програмного забезпечення. Тому актуальною є розробка програмного забезпечення для ведення щоденника та управління завданнями.

При розробці програмної платформи для ведення щоденника та управління завданнями необхідно забезпечити такі функціональні вимоги:

- підтримка можливості ведення щоденника завдань та зустрічей користувача;
- підтримка можливості відображення завдань та зустрічей користувача у вигляді календаря подій;
- можливість отримання інформації про контакти користувача;
- можливість редагування інформації про зустрічі користувача;
- підтримка можливості доступу до адміністративної панелі програмної платформи для редагування даних системи.

1.3 Висновки за розділом 1

Визначено, що програмне забезпечення для ведення щоденника та управління завданнями слугує цифровим середовищем, яке дозволяє організовувати особисті справи, контролювати поточні та майбутні події, а також зберігати нотатки й важливі записи у впорядкованому вигляді. Такий інструмент об'єднує різні функції, що допомагають стежити за виконанням планів і водночас фіксувати деталі, які можуть знадобитися в роботі чи повсякденному житті. Завдяки подібним програмним рішенням зникає необхідність вести окремі паперові щоденники чи створювати нагадування в різних додатках, адже вся інформація збирається в одному місці й залишається доступною у зручному форматі.

За результатами проведеного аналізу зроблено висновок, що у наш час існує досить багато програмних засобів для ведення щоденників та управління завданнями. Проте деякі програми для ведення персональних щоденників та управління завданнями характеризуються складністю інтерфейсу або мати надлишкові функції, які ускладнюють швидке освоєння і викликають перевантаження користувача, особливо якщо програму використовують люди без технічного досвіду. По-друге, залежність від Інтернет з'єднання для синхронізації та збереження даних може створювати проблеми у разі нестабільного доступу до мережі. Також існує ризик втрати конфіденційної інформації через недостатній рівень захисту або витік даних. Важливим недоліком може бути обмежена сумісність з іншими застосунками або платформами, що ускладнює інтеграцію у вже існуючі робочі процеси. Крім того, деякі програми можуть не враховувати індивідуальні потреби користувачів, пропонуючи надто універсальні рішення без гнучкості налаштувань. У певних випадках вартість підписки або ліцензії на програмне забезпечення може бути перешкодою для широкого використання. Таким чином, незважаючи на численні переваги, ці недоліки можуть впливати на загальну ефективність і зручність використання такого програмного забезпечення. Тому актуальною є розробка програмного забезпечення для ведення щоденника та управління завданнями.

Сформульовано функціональні вимоги до програмної платформи для ведення щоденника та управління завданнями.

2 МАТЕРІАЛИ І МЕТОДИ

2.1 Вибір мови програмування

При розробці програмної платформи для ведення щоденника та управління завданнями було використано мову програмування PHP [14], [15].

PHP вирізняється як мова, що часто обирається для створення інтерактивних вебсистем, оскільки забезпечує простий старт і дозволяє швидко будувати робочі зв'язки між клієнтом і сервером. Широка база хостингів, які підтримують цю технологію без додаткових ускладнень, спрощує запуск і перенесення проєктів у робоче середовище. Розвинута система сумісності з багатьма популярними системами управління базами даних (СУБД) відкриває можливість створення рішень різної складності — від невеликих особистих сервісів до масштабних багатокористувацьких платформ із великими обсягами інформації. Сфера PHP розвивається завдяки значній кількості розробників і постійному оновленню супутніх бібліотек та фреймворків, що помітно скорочує час реалізації типових компонентів. Наявність перевірених шаблонів дозволяє зосередитися саме на ключовій бізнес-логіці, уникаючи зайвих витрат часу на тривіальні рішення. З огляду на це, PHP часто розглядається як практичний вибір для тих випадків, коли необхідно швидко розгорнути і підтримувати вебплатформу з оптимальним співвідношенням ресурсів і функціоналу [14], [15].

PHP залишається однією з тих мов програмування, які найчастіше обираються для створення вебплатформ, що потребують постійного доступу з різних пристроїв через звичайний браузер. Завдяки своїй архітектурі вона ефективно виконує роль рушія серверної логіки, обробляючи запити користувачів, керуючи даними у сховищах та виконуючи процедури автентифікації й контролю доступу. Сучасні PHP-фреймворки на кшталт Laravel чи Symfony дозволяють уникнути зайвої рутинної роботи та організувати розробку у чіткій і безпечній структурі, що підвищує якість і стабільність готового рішення [14], [15].

Завдяки тому, що розгортання PHP-застосунків не потребує значних фінансових витрат і складних технічних умов, запуск сервісу навіть у невеликих проєктах часто стає швидким і вигідним. Ця мова легко взаємодіє з найпоширенішими СУБД, що робить її зручною для збереження даних користувачів, нотаток або завдань, а при зростанні обсягів інформації система залишається придатною до масштабування. Не варто недооцінювати й той факт, що довкола PHP сформувалася велика база готових інструментів і рішень, що істотно полегшує обслуговування платформи й подальше вдосконалення її функцій [14], [15].

Обґрунтування вибору мови програмування для реалізації програмної платформи для ведення щоденника та управління завданнями наведено у таблиці 2.1.

Таблиця 2.1 – Обґрунтування вибору мови програмування для реалізації програмної платформи для ведення щоденника та управління завданнями

Критерій порівняння мов програмування	Мова програмування		
	PHP	Python	JavaScript
Швидкість розробки	+	+—	+—
Легкість інтеграції з СУБД	+	+	+—
Простота розгортання на більшості хостингів	+	+—	+—
Зручність для створення ПЗ для ведення щоденника та управління завданнями	+	—	—
Велика спільнота і кількість рішень	+	+	+—

Якщо постає завдання реалізувати онлайн систему з гнучким особистим кабінетом, можливістю керування повідомленнями, календарями чи спільною роботою над завданнями, мова програмування PHP надає для цього достатній набір технічних засобів, залишаючись зрозумілим і доступним для широкого кола розробників, що забезпечує безперервну підтримку й розвиток таких рішень [14], [15].

Отже, для реалізації програмної платформи для ведення щоденника та управління завданнями обрано мову програмування PHP, яка є однією з найпоширеніших мов для створення вебзастосунків та добре підходить для обробки запитів, роботи з базами даних і управління авторизацією користувачів. Крім того, завдяки великій кількості готових фреймворків можна значно пришвидшити розробку, забезпечуючи при цьому високу структурованість коду, безпеку та масштабованість.

2.2 Вибір середовища розробки для створення програмної платформи для ведення щоденника та управління завданнями

В якості середовища розробки для створення програмної платформи для ведення щоденника та управління завданнями було обрано середовище PhpStorm [16], [17].

Середовище розробки PhpStorm поєднує у собі потужність сучасного середовища для серверної розробки та розширену підтримку всіх мов і технологій, які зазвичай використовуються у вебсфері. Наявність розумної системи автодоповнення та глибокого аналізу коду дає змогу значно знизити кількість помилок ще до етапу тестування, що особливо цінно під час роботи з великими чи комплексними вебплатформами. Робота у цьому середовищі дозволяє швидко орієнтуватися у структурі багатокomпонентного проєкту, знаходити залежності та виконувати зміни у коді без ризику порушити цілісність логіки застосунку [16], [17].

PhpStorm легко поєднується з різними інструментами, які забезпечують контроль версій, автоматизовані збірки або тестування, що створює сприятливі умови для налагоджених командних процесів і гнучкого управління циклом розробки. Крім того, це середовище пропонує зручні засоби для виявлення «вузьких місць» у продуктивності, даючи змогу завчасно усувати критичні проблеми. Завдяки такому підходу до організації робочого простору досягається висока якість підсумкового продукту й знижується ризик появи прихованих помилок [16], [17].

Завдяки комплексному підходу до підтримки не лише PHP, а й суміжних мов розмітки та стилів, це середовище дозволяє тримати під контролем усі етапи розробки вебзастосунків, що робить його особливо доречним для реалізації рішень з багатим функціоналом і високими вимогами до стабільності роботи [16], [17].

Використання PhpStorm під час розробки програмної системи для нотаток і менеджменту завдань обґрунтовується цілим рядом технічних переваг і практичних зручностей, які воно надає розробнику. Це середовище спроектоване так, аби максимально відповідати вимогам проєктів, що базуються на PHP, а також охоплювати супутні технології, які традиційно застосовуються у вебсфері. Повноцінний набір інструментів для редагування, ретельного тестування і точного налагодження коду дозволяє тримати якість проєкту на високому рівні з мінімальними витратами часу на рутинні технічні процеси [16], [17].

Ефективне поєднання з популярними фреймворками та зручна взаємодія з базами даних спрощують робочий цикл і дають змогу швидше перейти від етапу підготовки середовища до безпосередньої реалізації логіки сервісу. У межах одного проєкту можна легко орієнтуватися між різними файлами, відстежувати зв'язки, знаходити потрібні фрагменти коду і бачити можливі помилки ще до моменту їх появи у фінальній збірці. Підсвітка синтаксису, інтелектуальне автодоповнення та статичний аналіз скорочують кількість помилок і суттєво підвищують швидкість роботи [16], [17].

Розробникам також зручно налаштовувати під PhpStorm локальні сервери чи підключатися до віддалених середовищ, що особливо цінно, коли над однією платформою працює кілька учасників команди одночасно. Це середовище сприяє впорядкованій структурі коду і допомагає без зайвих складнощів впроваджувати нові функції або масштабувати вже наявні рішення. Не останню роль відіграє і доступність якісної документації разом з активною спільнотою, що спрощує вирішення специфічних технічних питань. Завдяки такому поєднанню можливостей і гнучкій орієнтації на PHP, PhpStorm залишається одним із найзручніших інструментів для створення потужної та стійкої системи для особистого чи командного планування [16], [17].

Обґрунтування вибору середовища розробки для створення програмної платформи для ведення щоденника та управління завданнями наведено у таблиці 2.2.

Таблиця 2.2 – Обґрунтування вибору середовища розробки для створення програмної платформи для ведення щоденника та управління завданнями

Критерій порівняння середовищ розробки	Середовища розробки		
	PhpStorm	Visual Studio Code	Sublime Text
Інтеграція з фреймворками	+	+—	—
Автоматизація та підказки при написанні коду	+	+	+—
Простота інтерфейсу	+	—	+
Наявність вбудованих інструментів налагодження	+	+	—
Зручність роботи з базами даних	+	+—	—

Таким чином, для створення програмної платформи для ведення щоденника та управління завданнями обрано середовище розробки PhpStorm, що розроблене спеціально для роботи з PHP і суміжними вебтехнологіями, пропонує широкий спектр інструментів для написання, налагодження та тестування коду. Завдяки глибокій інтеграції з популярними фреймворками та системами управління базами даних вдається значно знизити час налаштування середовища та сконцентруватися безпосередньо на розробці функціоналу. PhpStorm забезпечує зручну навігацію по проєкту, автоматичне доповнення коду, статичний аналіз та миттєву підсвітку помилок, що допомагає уникати багатьох помилок на ранніх етапах.

2.3 Розробка моделі взаємодії користувача з програмним забезпеченням для ведення щоденника та управління завданнями

З метою можливості розробки програмної платформи для ведення щоденника та управління завданнями створимо відповідну модель у вигляді діаграми прецедентів.

При розробленні моделі взаємодії користувача з програмним забезпеченням для ведення щоденника та управління завданнями враховуємо те, що користувачі програми мають можливості авторизації та реєстрації, перегляду розділів щоденника, редагування інформації у щоденнику, планування зустрічей.

Основний успішний сценарій користувача при роботі з програмним забезпеченням для ведення щоденника та управління завданнями можна описати таким чином:

- користувач програмного забезпечення для ведення щоденника та управління завданнями авторизується;
- після авторизації користувача сервер програмного забезпечення для ведення щоденника та управління завданнями у відповідь повертає основну сторінку вебсайту для ведення щоденника та управління завданнями;

- користувач бачить відповідну сторінку програмного забезпечення для ведення щоденника та управління завданнями та переходить до розділу Зустрічі;

- користувач за необхідності шукає потрібну зустріч;
- користувач натискає відповідну кнопку для редагування зустрічі та змінює інформацію про зустрічі, або створює нову зустріч;
- користувач завершує роботу з програмним забезпеченням для ведення щоденника та управління завданнями.

Враховуючи сценарії роботи користувачів програмної платформи для ведення щоденника та управління завданнями (гість, користувач та адміністратор) при роботі з програмою, модель взаємодії користувача з програмним забезпеченням для ведення щоденника та управління завданнями подамо у вигляді діаграми прецедентів, зображеної на рис. 2.1.

Діаграма прецедентів програмної платформи для ведення щоденника та управління завданнями відображає три ролі з різними наборами можливостей. На діаграмі окремо позначається гість, який має мінімальний доступ і здатен пройти реєстрацію або здійснити вхід до облікового запису. Після авторизації користувач отримує розширений функціонал — з'являється можливість створювати записи у різних тематичних розділах, працювати з календарем, планувати зустрічі та зберігати контактну інформацію. Крім того, користувачеві доступні дії з перегляду і внесення змін до зазначених розділів, що дозволяє підтримувати записи у актуальному стані та вчасно оновлювати важливі дані. Окремо на діаграмі виділяється адміністратор, який володіє правами для управління обліковими записами інших користувачів, зокрема, перегляду даних користувачів та блокування профілів. Кожен з цих учасників програмної платформи для ведення щоденника та управління завданнями пов'язаний з відповідними прецедентами, що описують послідовність дій та взаємодію з платформою відповідно до наданих прав.

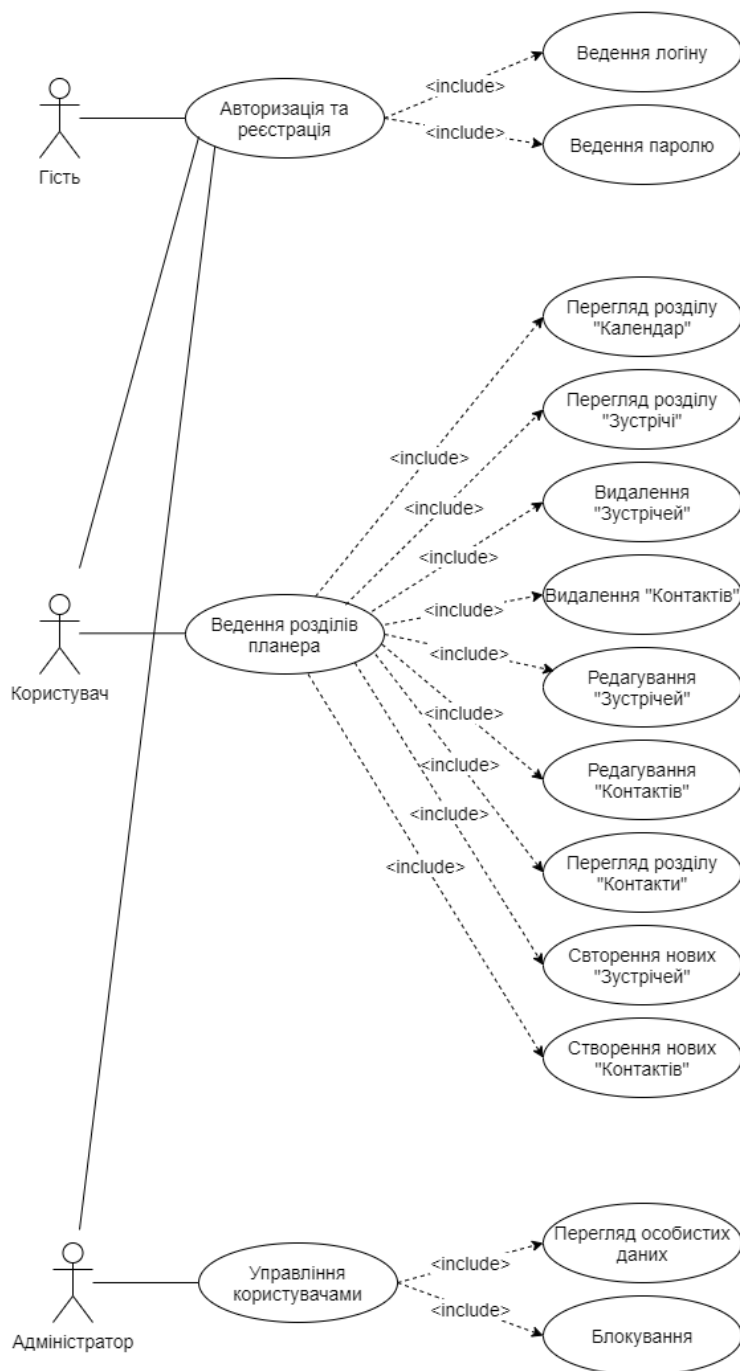


Рисунок 2.1 – Модель взаємодії користувача з програмним забезпеченням для ведення щоденника та управління завданнями

Таким чином, створено модель взаємодії користувача з програмним забезпеченням для ведення щоденника та управління завданнями, яка подана у вигляді діаграми прецедентів, та дозволяє розроблювати програмне забезпечення для ведення щоденника та управління завданнями.

2.4 Висновки за розділом 2

Для реалізації програмного забезпечення для ведення щоденника та управління завданнями обрано мову програмування PHP, яка є однією з найпоширеніших мов для створення вебзастосунків та добре підходить для обробки запитів, роботи з базами даних і управління авторизацією користувачів. Крім того, завдяки великій кількості готових фреймворків можна значно пришвидшити розробку, забезпечуючи при цьому високу структурованість коду, безпеку та масштабованість.

Для створення програмного забезпечення для ведення щоденника та управління завданнями обрано середовище розробки PhpStorm, що розроблене спеціально для роботи з PHP і суміжними вебтехнологіями, пропонує широкий спектр інструментів для написання, налагодження та тестування коду. Завдяки глибокій інтеграції з популярними фреймворками та системами управління базами даних вдається значно знизити час налаштування середовища та сконцентруватися безпосередньо на розробці функціоналу. PhpStorm забезпечує зручну навігацію по проєкту, автоматичне доповнення коду, статичний аналіз та миттєву підсвітку помилок, що допомагає уникати багатьох помилок на ранніх етапах.

Створено модель взаємодії користувача з програмним забезпеченням для ведення щоденника та управління завданнями, яка подана у вигляді діаграми прецедентів, та дозволяє розроблювати програмне забезпечення для ведення щоденника та управління завданнями.

3 ОПИС ПРОГРАМИ

3.1 Структура програмної платформи для ведення щоденника та управління завданнями

Структуру програмного забезпечення для ведення щоденника та управління завданнями наведено на рис. 3.1.

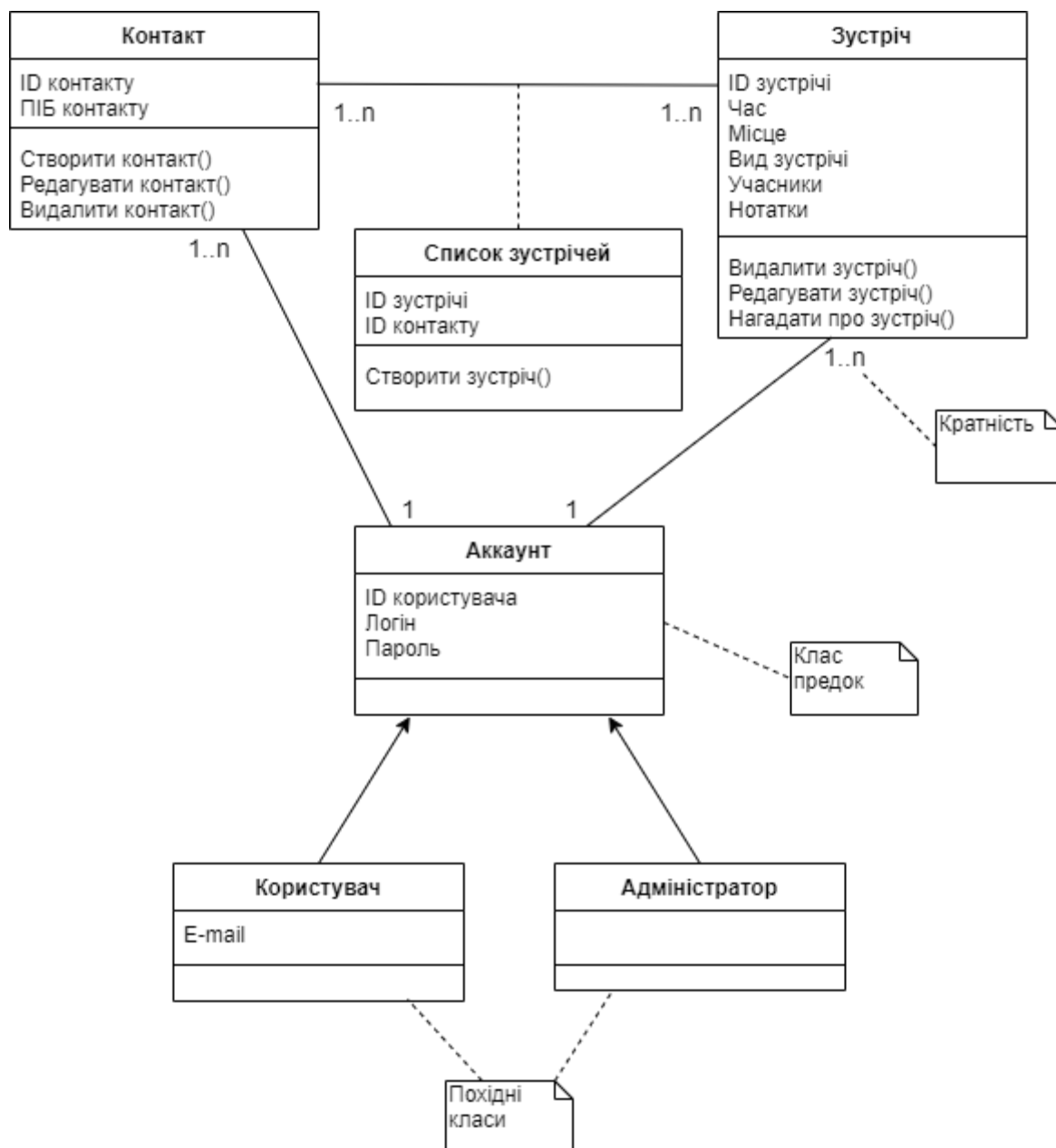


Рисунок 3.1 – Структура програмного забезпечення для ведення щоденника та управління завданнями

Як видно з рис. 3.1, основними класами програмної платформи для ведення щоденника та управління завданнями є Контакт, Аккаунт, Зустріч, Список зустрічей, Користувач, Адміністратор.

Діаграма класів для системи, що призначена для організації особистого щоденника та управління списком завдань, включає декілька ключових об'єктів, кожен з яких виконує власну роль у структурі платформи. Основою виступає клас Аккаунт, що містить базову інформацію про обліковий запис і слугує спільною ланкою для похідних сутностей. Від цього класу успадковуються Користувач і Адміністратор, які розширюють базову функціональність — звичайний користувач отримує доступ до записів і особистих нотаток, тоді як адміністратор наділяється правами для управління всіма обліковими записами у системі.

Окремим блоком представлено клас Контакт, який зберігає дані про окремі особи або організації, що можуть бути пов'язані із записами чи подіями. Для організації планування у структурі діаграми передбачено клас Зустріч, що описує окрему подію з визначеними параметрами — датою, часом, темою чи місцем проведення. Групування подій реалізується за допомогою класу Список зустрічей, що забезпечує зберігання декількох записів зустрічей та дозволяє оперувати ними як єдиним масивом: додавати нові, редагувати наявні чи вилучати непотрібні.

Усі класи взаємодіють між собою через чітко визначені зв'язки: користувачі створюють записи, прикріплюють контакти до зустрічей і формують власні списки для планування справ. Така структура дозволяє чітко розмежувати функції зберігання особистих даних, організації подій і адміністрування системи, підтримуючи логічну послідовність і взаємозв'язки між усіма компонентами.

Діаграму компонентів програмної платформи для ведення щоденника та управління завданнями наведено на рис. 3.2.

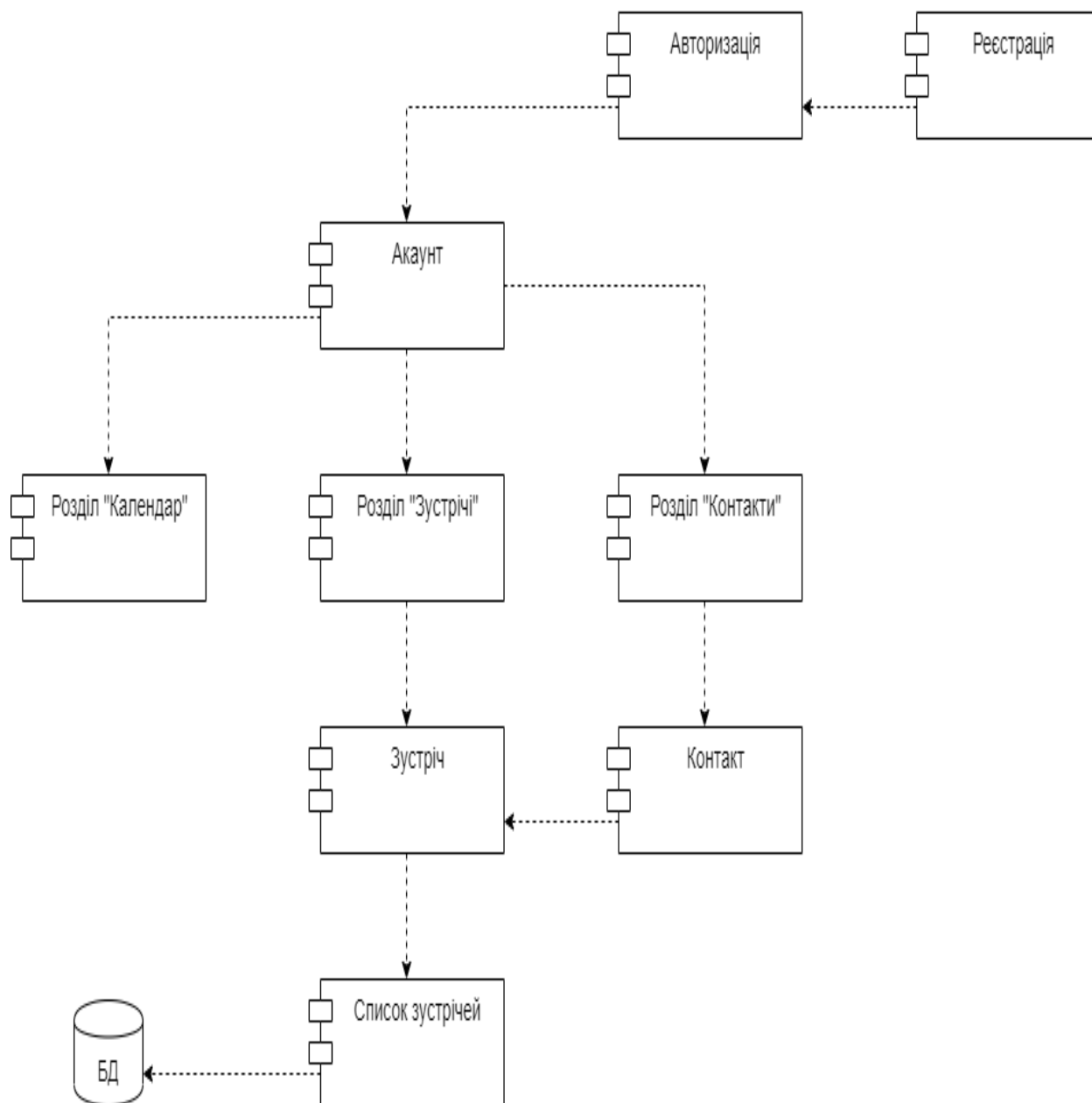


Рисунок 3.2 – Діаграма компонентів програмної платформи для ведення щоденника та управління завданнями

Діаграма компонентів для програмної платформи, яка забезпечує ведення щоденника й контроль завдань, демонструє, як різні частини системи об'єднуються для виконання окремих функцій. Основу складає компонент Акаунт, що відповідає за зберігання та обробку ключової інформації про профілі користувачів. З ним безпосередньо взаємодіють Авторизація та

Реєстрація — окремі модулі, що реалізують механізми перевірки даних при вході та створенні нових облікових записів.

Для організації особистих записів передбачено кілька тематичних модулів: Розділ Календар надає функції для планування подій за датами, Розділ Зустрічі фокусується на збереженні інформації про заплановані домовленості та зустрічі, тоді як Розділ Контакти містить дані про людей чи організації, з якими може бути пов'язана діяльність користувача. Усі ці частини об'єднані спільною логікою доступу до Бази даних, що виступає як центральне сховище для всієї інформації платформи — від облікових даних до списків подій і контактів.

Кожен із цих компонентів взаємодіє через чітко визначені інтерфейси, що дозволяє легко підтримувати систему та додавати нові функції. Додаткові службові модулі забезпечують роботу сповіщень, безпечне зберігання сесій і підтримку зв'язку між частинами програми.

Така структура дозволяє розмежувати відповідальність кожного модуля та забезпечити стабільність роботи платформи незалежно від кількості користувачів чи обсягу даних.

3.2 Функціонування програмної платформи для ведення щоденника та управління завданнями

Функціонування програмної платформи для ведення щоденника та управління завданнями подамо за допомогою діаграм, зображених на рис. 3.3-3.6.

На рис. 3.3 зображено діаграму діяльності програмної платформи для ведення щоденника та управління завданнями. Діаграма діяльності для цієї платформи відображає послідовність кроків, що виконуються під час створення нової зустрічі. Початковий етап передбачає ініціювання дії — користувач переходить до створення запису про зустріч. Далі перевіряється, чи прив'язано контакт до майбутньої події: у разі відсутності відповідного

запису система одразу інформує про неможливість продовження і завершує спробу, сповіщаючи користувача про відсутність контакту.

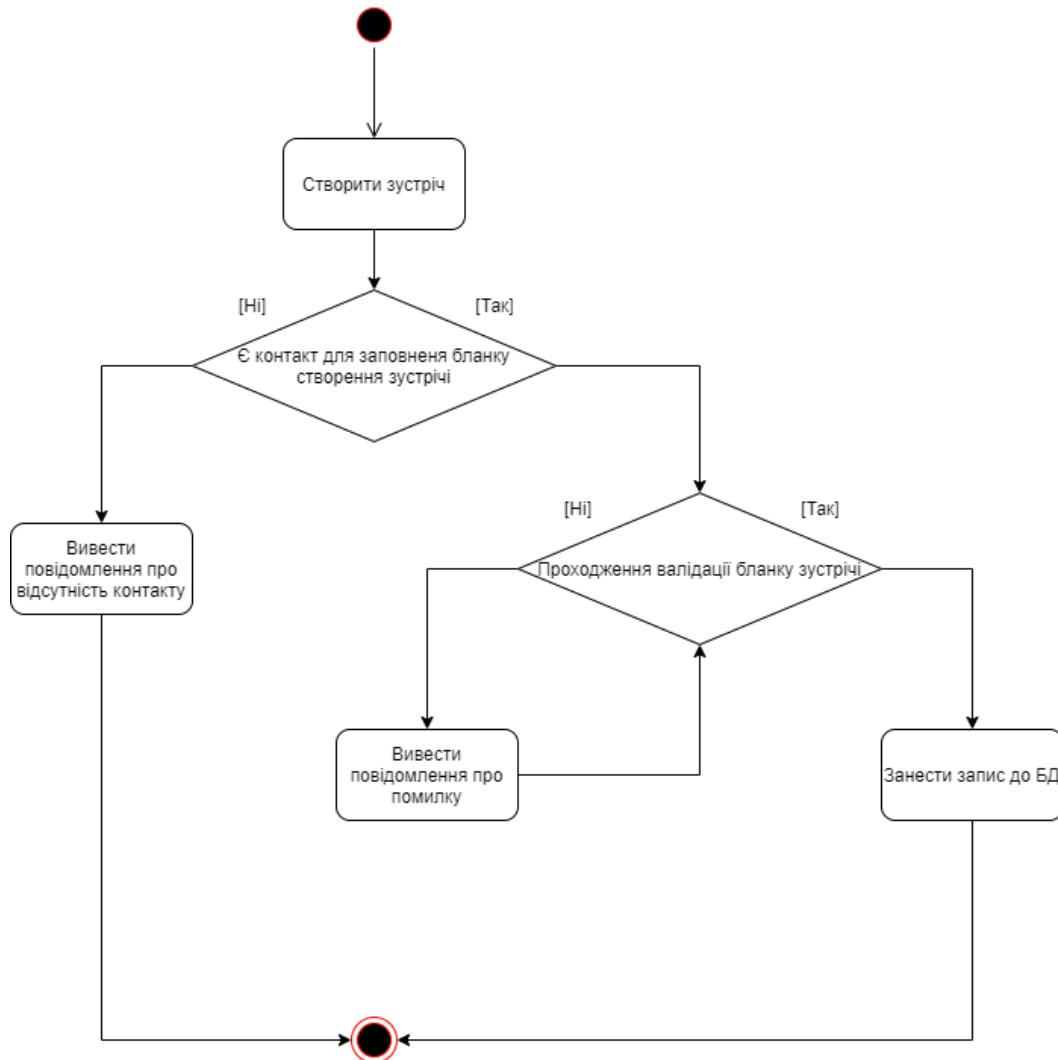


Рисунок 3.3 – Діаграма діяльності програмної платформи для ведення щоденника та управління завданнями

Якщо контакт знайдено, процес переходить до етапу перевірки правильності введених даних. Ця валідація визначає, чи всі обов’язкові поля заповнено коректно. Якщо під час перевірки виявляється помилка, користувач бачить повідомлення про недоліки і повертається до етапу виправлення форми для повторної перевірки. Коли всі дані успішно проходять перевірку, система завершує операцію внесенням інформації про зустріч до бази даних, що фіксує

заплановану подію та робить її доступною для подальшого перегляду або редагування. Така схема описує чіткий порядок взаємодії користувача з платформою та демонструє, як система контролює послідовність кроків, мінімізуючи ймовірність помилок.

На рис. 3.4 зображено діаграму станів програмної платформи для ведення щоденника та управління завданнями.

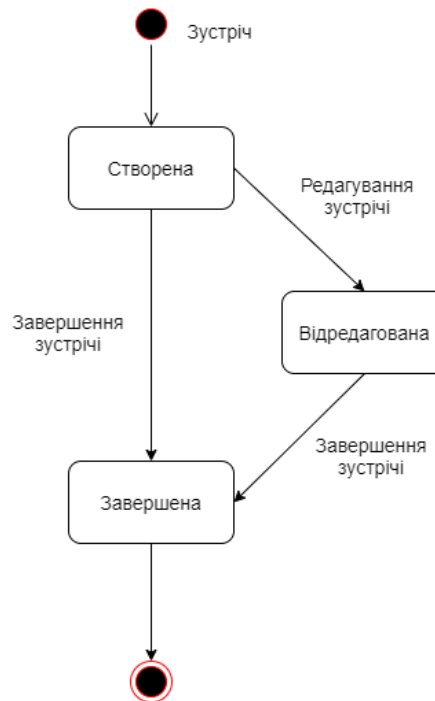


Рисунок 3.4 – Діаграма станів програмної платформи для ведення щоденника та управління завданнями

Діаграма станів програмної платформи для ведення щоденника та управління завданнями ілюструє, як змінюється статус зустрічі протягом усього її життєвого циклу. На початку подія перебуває у стані «Створена», що означає, що всі необхідні дані внесено й запис збережено у системі. З цього стану можливі два шляхи: за потреби можна внести зміни до деталей — тоді зустріч переходить у стан «Відредагована», що фіксує факт оновлення інформації. Після цього вона знову готова до виконання або очікує завершення. Другий варіант розвитку передбачає безпосереднє завершення — після того, як запланована подія відбулася або була скасована, її статус

змінюється на «Завершена», що вказує на закінчення циклу. Таким чином, діаграма відображає послідовність змін стану однієї зустрічі — від моменту створення через можливі виправлення до фінальної фіксації факту виконання чи закриття події.

Діаграму кооперацій програмної платформи для ведення щоденника та управління завданнями наведено на рис. 3.5.

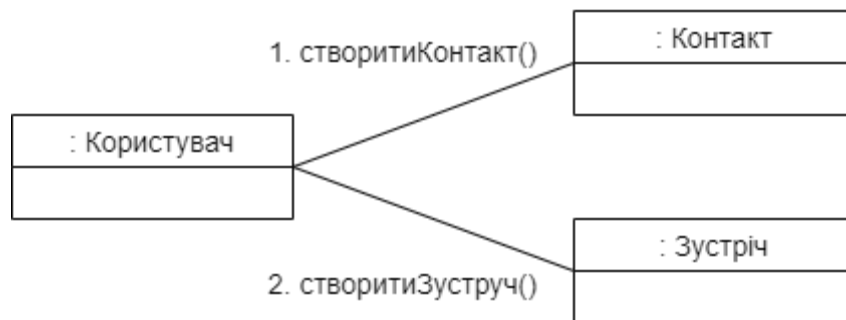


Рисунок 3.5 – Діаграма кооперацій програмної платформи для ведення щоденника та управління завданнями

Діаграма кооперацій демонструє, як взаємодіють об'єкти програмної платформи для ведення щоденника та управління завданнями під час створення ключових записів. В центрі зображено Користувача, який ініціює дві окремі дії. Перша з них — передача запиту на формування нового запису у компонент Контакт, що дозволяє додати до системи інформацію про людину чи організацію для подальшого використання під час планування подій. Друга взаємодія реалізується через створення нового запису про майбутню подію: користувач надсилає команду до об'єкта Зустріч, що відповідає за фіксацію даних про заплановану активність у щоденнику. Обидва зв'язки відображають послідовність взаємодії між роллю користувача та відповідними складовими системи, підкреслюючи, що саме людина визначає момент створення контактів і зустрічей, а система у відповідь виконує обробку та збереження цих даних.

На рис. 3.6 зображено діаграму послідовності програмної платформи для ведення щоденника та управління завданнями.

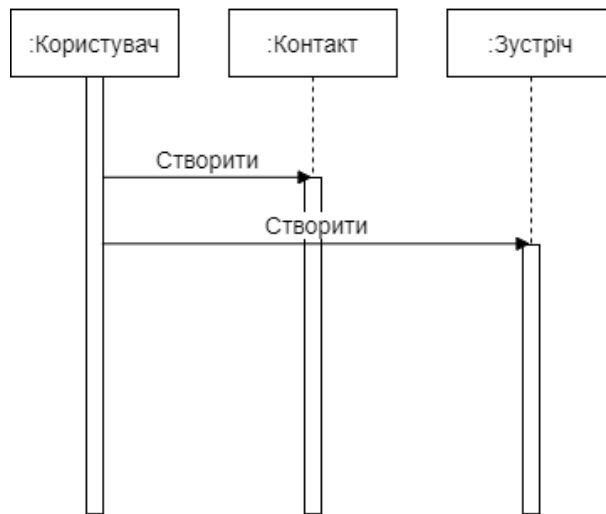


Рисунок 3.6 – Діаграма послідовності програмної платформи для ведення щоденника та управління завданнями

Діаграма послідовності описує, як саме користувач взаємодіє із системою під час додавання нових даних. На схемі видно, що ініціатором усіх дій виступає Користувач, який спершу надсилає запит на створення запису про зустріч — відповідний запит спрямовується до об'єкта Зустріч, де формується потрібний запис із заданими параметрами. Паралельно або одразу після цього користувач ініціює ще одну дію — формує команду для додавання інформації у Контакт, що дозволяє зберегти дані про людину чи організацію, які можуть бути пов'язані з цією подією. Така послідовність ілюструє, як користувач керує наповненням щоденника: спочатку фіксується подія, а потім або одночасно створюється контакт, щоб пов'язати його з відповідною зустріччю. Обидві взаємодії відображають прямий зв'язок між діями користувача й відповіддю системи, забезпечуючи чіткий ланцюг обробки та збереження введеної інформації.

3.3 Архітектура вебзастосунку програмного забезпечення для ведення щоденника та управління завданнями

Архітектура вебзастосунку програмного забезпечення для ведення щоденника та управління завданнями побудована на основі шаблону Model-View-Controller (MVC).

Схему архітектури програмного забезпечення для ведення щоденника та управління завданнями побудована на основі шаблону MVC наведено на рис. 3.7.

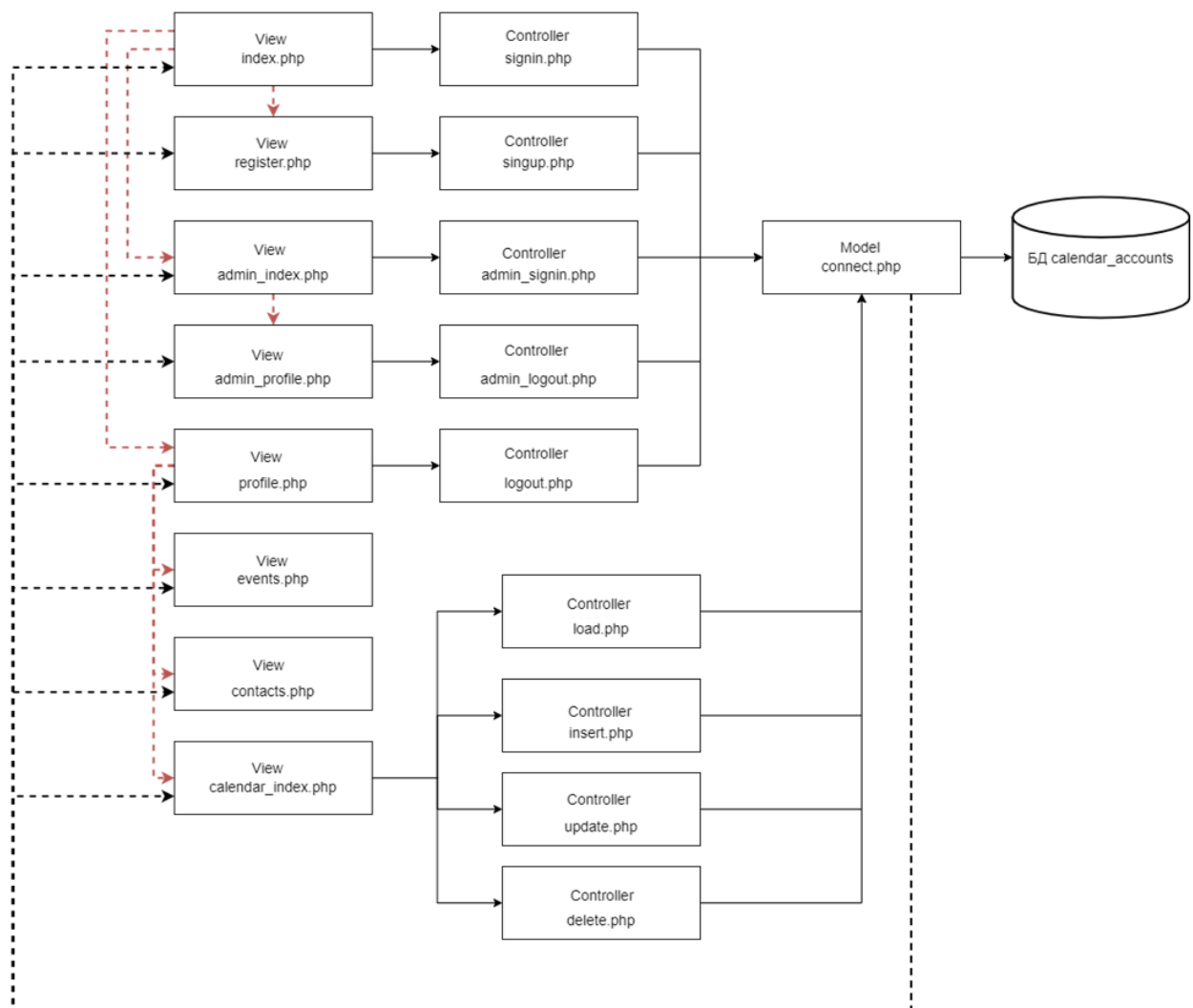


Рисунок 3.7 – Схема архітектури програмного забезпечення для ведення щоденника та управління завданнями

Архітектура такого програмного забезпечення для ведення щоденника та управління завданнями, побудована на принципах Model-View-Controller, забезпечує чітке розділення обов'язків між обробкою даних, логікою взаємодії та відображенням інформації. У цій структурі вся робота з записами щоденника та організацією завдань зосереджена у моделях, які відповідають за збереження стану, обробку інформації та зв'язок із сховищем даних. Завдяки їм платформа для ведення щоденника та управління завданнями утримує відомості про події, нотатки, списки справ чи контактні деталі у впорядкованому вигляді.

Контролери програмного забезпечення для ведення щоденника та управління завданнями виступають посередниками, які приймають запити користувача, перевіряють правильність вхідних даних, координують взаємодію між моделями й визначають, які дії мають бути виконані. Саме через контролери реалізується створення нових записів, редагування чи видалення зустрічей, перевірка прав доступу й інші етапи, пов'язані з обробкою дій користувача.

Відповідь на запити формується за допомогою представлень програмного забезпечення для ведення щоденника та управління завданнями, які відповідають за вивід підготовлених даних у зручній формі — користувач бачить сторінки з календарем, переліком контактів або списками завдань так, як це передбачено інтерфейсом. Такий поділ дозволяє розвивати та підтримувати програмне забезпечення для ведення щоденника та управління завданнями більш структуровано: зміни у візуальній частині не зачіпають логіку бізнес-процесів, а вдосконалення моделі не потребує переписування контролерів чи шаблонів відображення. Усе це робить програмну платформу для ведення щоденника та управління завданнями зрозумілою для команди розробників і зручною для подальшого розширення функцій.

На основі розробленої архітектури та діаграм, запропонованих у попередніх розділах, створено програмну платформу для ведення щоденника та управління завданнями, файлову структуру якої зображено на рис. 3.8.

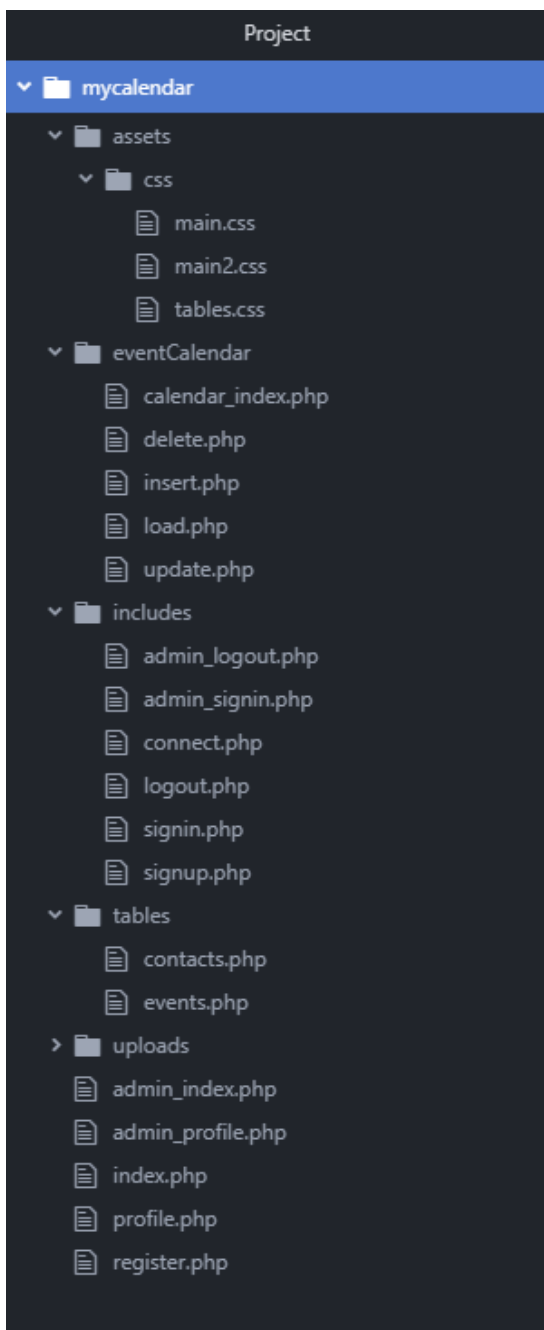


Рисунок 3.8 – Файлова структура програмної платформи для ведення щоденника та управління завданнями

3.4 Проєктування інтерфейсу програмної платформи для ведення щоденника та управління завданнями

Проєктування інтерфейсу взаємодії користувача з програмним забезпеченням для ведення щоденника та управління завданнями виконано з урахуванням вимог технічного завдання. При розробленні програмної

платформи для ведення щоденника та управління завданнями враховані функціональні вимоги до програми:

- підтримка можливості ведення щоденника завдань та зустрічей користувача;
- підтримка можливості відображення завдань та зустрічей користувача у вигляді календаря подій;
- можливість отримання інформації про контакти користувача;
- можливість редагування інформації про зустрічі користувача;
- підтримка можливості доступу до адміністративної панелі програмної платформи для редагування даних системи.

Інтерфейс сторінки аутентифікації користувача програмної платформи для ведення щоденника та управління завданнями наведено на рис. 3.9.

The image shows a login interface for a web application. At the top, the title 'Програмна платформа для ведення щоденника та управління завданнями' is displayed in bold black text. Below the title, the subtitle 'Вхід до платформи' is centered. The login form consists of two input fields: 'Логін' (Login) with the placeholder text 'Введіть свій Логін' and 'Пароль' (Password) with the placeholder text 'Введіть свій Пароль'. Below these fields is a grey button labeled 'Увійти' (Login). At the bottom, there is a link that reads 'Немає акаунту? Тоді необхідно [зареєструватися](#)' (Don't have an account? Then you need to [register](#)).

Рисунок 3.9 – Інтерфейс головної форми програмної платформи для ведення щоденника та управління завданнями

Інтерфейс прототипу сторінки реєстрації користувача програми для ведення щоденника та управління завданнями наведено на рис. 3.10.

Сторінка профілю користувача програми для ведення щоденника та управління завданнями наведена на рис. 3.11.

**Програмна платформа для ведення
щоденника та управління завданнями**

Реєстрація

Ім'я
Введіть Ім'я

Логін
Введіть Логін

Електронна пошта
Введіть свою електронну пошту

Файл з зображенням
Файл Файл не обрано

Пароль
Введіть пароль

Перевірка паролю
Введіть пароль (ще раз)

Зареєструватися

Вже є акаунт? [Заходь сюди](#)

Рисунок 3.10 – Сторінка реєстрації користувача програми для ведення щоденника та управління завданнями



My name: Chik
My email: Chik59354@gmail.com

Мій календар

Контакти

Зустрічі

Вихід

Рисунок 3.11 – Сторінка профілю користувача програми для ведення щоденника та управління завданнями

Сторінку календаря щоденника подій та завдань користувача наведено на рис. 3.12.

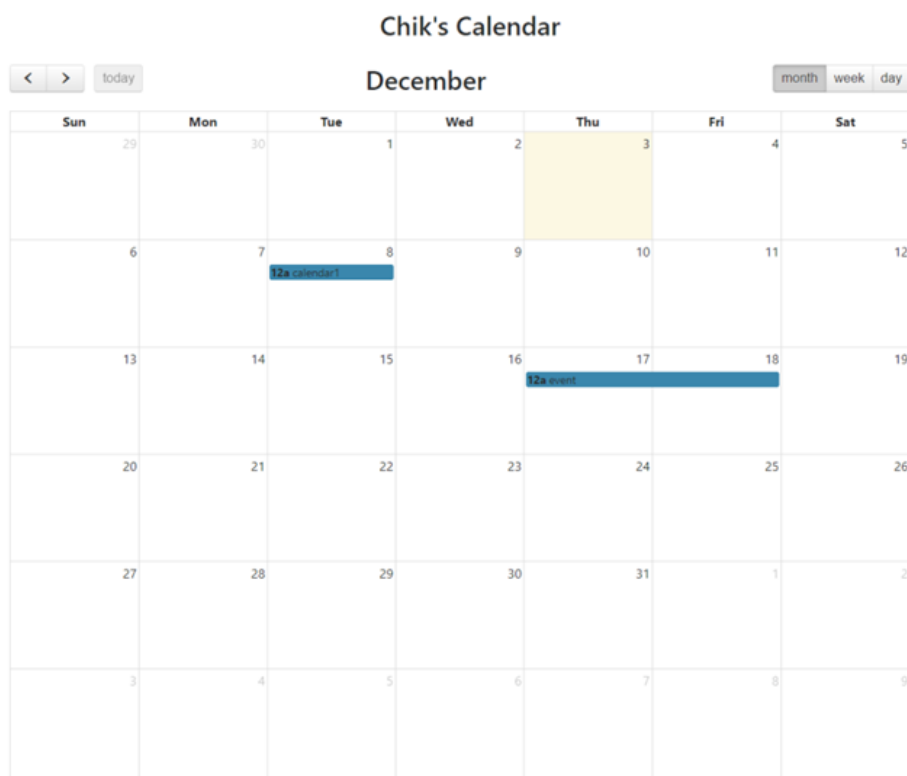


Рисунок 3.12 – Календар щоденника подій та завдань користувача

Сторінка контактів користувача програмної платформи для ведення щоденника та управління завданнями наведена на рис. 3.13.

Контакти			
User ID	Users Avatar	Username	Users Email
2		test_flo	test_email@local.com
9		Чувашин Чув	chuvash@user.com
10		Хингтон Билл	vansama@mail.com
11		admin	admin@email.com
15		Chel	calendar@email.com

Назад

Рисунок 3.13 – Сторінка контактів користувача програмної платформи для ведення щоденника та управління завданнями

Сторінка зустрічей користувача наведена на рис. 3.14.

Мої зустрічі			
Event ID	Event Title	Starting at	Ending at
8	calendar1	12-08 00:00:00	12-09 00:00:00
16	event	12-17 00:00:00	12-19 00:00:00

Назад

Рисунок 3.14 – Сторінка зустрічей користувача

3.5 Висновки за розділом 3

Розроблено програмну платформу для ведення щоденника та управління завданнями.

Запропоновано структуру програмної платформи для ведення щоденника та управління завданнями. Визначено, що основними класами програмної платформи для ведення щоденника та управління завданнями є Контакт, Аккаунт, Зустріч, Список зустрічей, Користувач, Адміністратор.

Функціонування програмної платформи для ведення щоденника та управління завданнями подамо за допомогою відповідних діаграм.

Архітектура вебзастосунку програмного забезпечення для ведення щоденника та управління завданнями побудована на основі шаблону Model-View-Controller. Описано функціонування програмної платформи для ведення щоденника та управління завданнями. Така архітектура забезпечує чітке розділення обов'язків між обробкою даних, логікою взаємодії та відображенням інформації. У цій структурі вся робота з записами щоденника та організацією завдань зосереджена у моделях, які відповідають за збереження стану, обробку інформації та зв'язок із сховищем даних.

Виконано проектування інтерфейсу взаємодії користувача з програмним забезпеченням для ведення щоденника та управління завданнями.

4 ЕКСПЛУАТАЦІЯ, ТЕСТУВАННЯ ТА ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ ПРОГРАМИ

4.1 Призначення й умови застосування програми

Програма призначена для ведення щоденника та управління завданнями. Програма написана на мові PHP, яка є однією з найпоширеніших мов для створення вебзастосунків та добре підходить для обробки запитів, роботи з базами даних і управління авторизацією користувачів. В якості середовища розробки для створення програмної платформи для ведення щоденника та управління завданнями обрано PhpStorm, що розроблене спеціально для роботи з PHP і суміжними вебтехнологіями, пропонує широкий спектр інструментів для написання, налагодження та тестування коду.

Програмне забезпечення для ведення щоденника та управління завданнями забезпечує виконання таких функцій:

- підтримка можливості ведення щоденника завдань та зустрічей користувача;
- підтримка можливості відображення завдань та зустрічей користувача у вигляді календаря подій;
- можливість отримання інформації про контакти користувача;
- можливість редагування інформації про зустрічі користувача;
- підтримка можливості доступу до адміністративної панелі програмної платформи для редагування даних системи.

Для роботи програмної платформи для ведення щоденника та управління завданнями потрібні такі технічні та програмні ресурси: комп'ютер або сервер із процесором із тактовою частотою не нижче 2,0 ГГц (рекомендується багатоядерний), обсяг оперативної пам'яті не менше 4 ГБ для мінімальної конфігурації та від 8 ГБ для стабільної роботи при одночасній роботі кількох користувачів. Вільний простір на диску має складати не менше

500 МБ для самої платформи та резервувати додаткове місце під зберігання даних і резервні копії.

Із програмного боку необхідна операційна система, сумісна з PHP-середовищем, можна використовувати Linux або Windows Server. Обов'язково потрібен вебсервер (Apache чи Nginx), встановлений PHP з відповідними модулями, а також система керування базами даних, наприклад, MySQL. Для локального розгортання чи тестування може знадобитися встановлення додаткового ПЗ для адміністрування баз даних (phpMyAdmin або інше). Також важливо передбачити браузер сучасної версії для користувачів, щоб інтерфейс програми коректно відображався й працював без помилок.

Функціональні характеристики розробленого програмного забезпечення для ведення щоденника та управління завданнями наведено у технічному завданні (додаток А). Фрагмент тексту програмного забезпечення для ведення щоденника та управління завданнями наведено у додатку Б.

4.2 Характеристики програми для ведення щоденника та управління завданнями

Програмна платформа для організації записів у щоденнику та контролю виконання завдань характеризується зручним інтерфейсом, що дозволяє швидко орієнтуватися у записах і переходити між розділами. Вона підтримує створення окремих подій, ведення списків зустрічей і зберігання контактів з можливістю швидкого доступу до кожного елемента. Передбачено механізми перевірки правильності введених даних, що знижує ризик помилок під час планування. Вся інформація структурується у базі даних і доступна для редагування чи перегляду у будь-який момент. Платформа передбачає багаторівневий доступ, що дозволяє розмежувати права користувачів та адміністраторів. Робота організована так, щоб усі зміни синхронізувалися без затримок, забезпечуючи актуальність інформації. Додаткові можливості при

необхідності розвитку програмної системи можуть включати нагадування та оповіщення для своєчасного виконання запланованих справ.

4.3 Інструкція по експлуатації програми

4.3.1 Звернення до програми

Для запуску програмної платформи для ведення щоденника та управління завданнями на серверній частині необхідно запуснути відповідний php-файл.

4.3.2 Вхідні й вихідні дані

Вхідними даними до програмної платформи для ведення щоденника та управління завданнями є інформація про користувача платформи, його зустрічі та контакти.

Вихідними даними програмної платформи для ведення щоденника та управління завданнями є календар зустрічей, вебсторінки порталу ведення щоденника.

4.4 Виконання програмної платформи для ведення щоденника та управління завданнями

Після запуску програмної платформи для ведення щоденника та управління завданнями користувачу відображається форма аутентифікації користувача (рис. 4.1), де потрібно ввести логін та пароль. Якщо користувач незареєстрований, потрібно обрати відповідне посилання для проходження процедури реєстрації.

Програмна платформа для ведення щоденника та управління завданнями

Вхід до платформи

Логін
Введіть свій Логін

Пароль
Введіть свій Пароль

Увійти

Немає акаунту? Тоді необхідно [зареєструватися](#)

Рисунок 4.1 – Інтерфейс сторінки аутентифікації користувача програмної платформи для ведення щоденника та управління завданнями

Для реєстрації (рис. 4.2) користувачу програмної платформи для ведення щоденника та управління завданнями необхідно ввести своє ім'я, електронну пошту, зображення та пароль.

Програмна платформа для ведення щоденника та управління завданнями

Реєстрація

Ім'я
Введіть Ім'я

Логін
Введіть Логін

Електронна пошта
Введіть свою електронну пошту

Файл з зображенням
 Файл не обрано

Пароль
Введіть пароль

Перевірка паролю
Введіть пароль (ще раз)

Зареєструватися

Вже є акаунт? [Заходь сюди](#)

Рисунок 4.2 – Сторінка реєстрації користувача програми для ведення щоденника та управління завданнями

Після входу до системи користувач бачить головну сторінку (рис. 4.3), де він може перейти до сторінок з календарем щоденника подій та завдань, контактів та зустрічей.



Рисунок 4.3 – Сторінка профілю користувача програми для ведення щоденника та управління завданнями

На сторінці календаря щоденника (рис. 4.4) відображаються події за завдання користувача у хронологічному порядку.

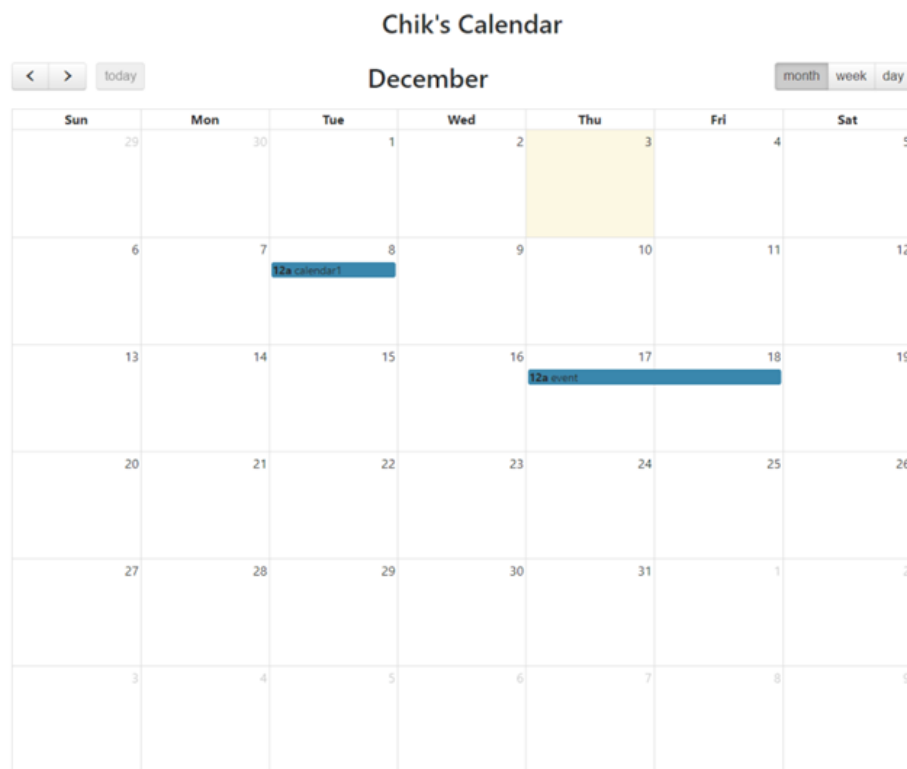


Рисунок 4.4 – Календар щоденника подій та завдань користувача

При виборі пункту меню Контакти відображається інформація про сторінка контактів користувача (рис. 4.5).

Контакти			
User ID	Users Avatar	Username	Users Email
2		test_fio	test_email@local.com
9		Чувашин Чув	chuvash@user.com
10		Хингтон Билл	vansama@mail.com
11		admin	admin@email.com
15		Chel	calendar@email.com
Назад			

Рисунок 4.5 – Сторінка контактів користувача програмної платформи для ведення щоденника та управління завданнями

При виборі пункту меню Зустрічі відображається сторінка запланованих зустрічей користувача (рис. 4.6), зокрема їх назва та час проведення.

Мої зустрічі			
Event ID	Event Title	Starting at	Ending at
8	calendar1	12-08 00:00:00	12-09 00:00:00
16	event	12-17 00:00:00	12-19 00:00:00

[Назад](#)

Рисунок 4.6 – Сторінка зустрічей користувача

При вході до системи як адміністратор (рис. 4.7) користувач отримає доступ до адміністративної панелі, де є можливість редагування даних системи.

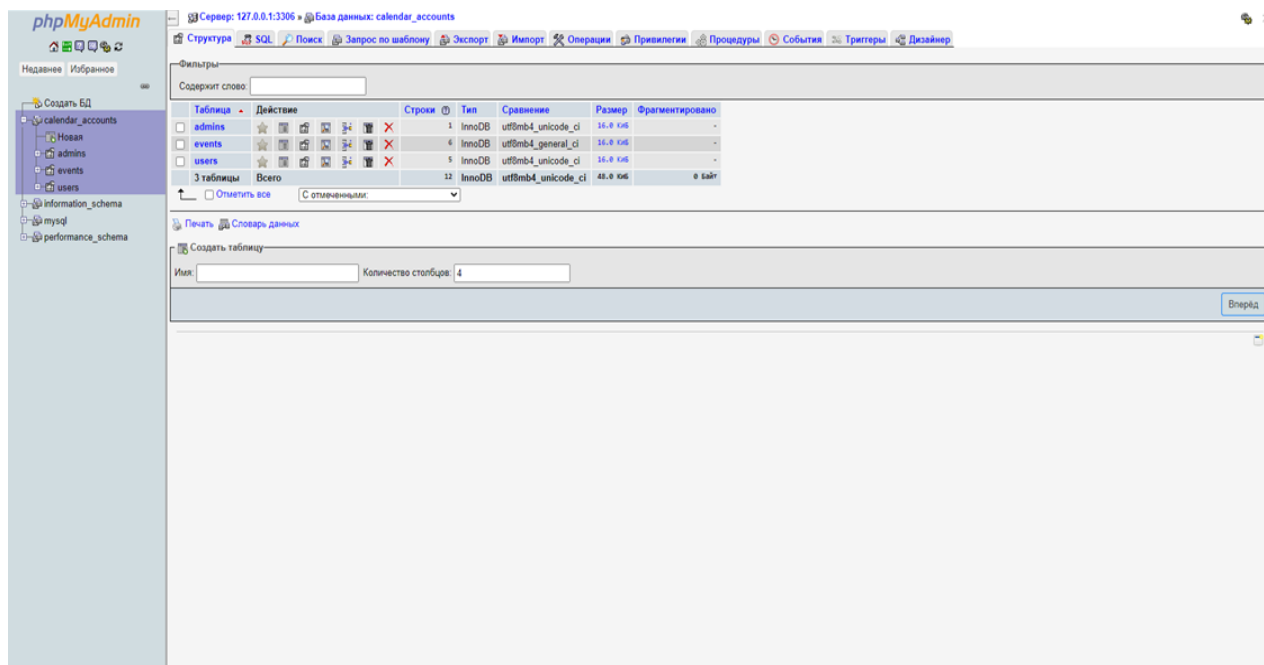


Рисунок 4.7 – Адміністративна панель програмної платформи для ведення щоденника та управління завданнями

4.5 Тестування програмної платформи для ведення щоденника та управління завданнями

Виконано тестування програмної платформи для ведення щоденника та управління завданнями.

Виявлено, що програма для ведення щоденника та управління завданнями функціонує правильно та злагоджено. Вона реалізує всі функціональні вимоги та успішно виконує свою основну задачу підтримки процесу ведення щоденника та управління завданнями.

Розроблене програмне забезпечення дозволяє забезпечувати автоматизацію процесів, пов'язаних з підтримкою управління завданнями користувача.

4.6 Висновки за розділом 4

Виконано тестування розробленого програмного забезпечення для ведення щоденника та управління завданнями. Результати тестування програмної платформи показали, що розроблена програма дозволяє забезпечити автоматизацію процесів, пов'язаних з підтримкою управління завданнями користувача.

ВИСНОВКИ

В ході виконання дипломної кваліфікаційної роботи магістра було проаналізовано та досліджено процес розробки програмного забезпечення для ведення щоденника та управління завданнями.

Визначено, що програмне забезпечення для ведення щоденника та управління завданнями слугує цифровим середовищем, яке дозволяє організовувати особисті справи, контролювати поточні та майбутні події, а також зберігати нотатки й важливі записи у впорядкованому вигляді. Такий інструмент об'єднує різні функції, що допомагають стежити за виконанням планів і водночас фіксувати деталі, які можуть знадобитися в роботі чи повсякденному житті. Завдяки подібним програмним рішенням зникає необхідність вести окремі паперові щоденники чи створювати нагадування в різних додатках, адже вся інформація збирається в одному місці й залишається доступною у зручному форматі.

За результатами проведеного аналізу зроблено висновок, що у наш час існує досить багато програмних засобів для ведення щоденників та управління завданнями. Проте деякі програми для ведення персональних щоденників та управління завданнями характеризуються складністю інтерфейсу або мати надлишкові функції, які ускладнюють швидке освоєння і викликають перевантаження користувача, особливо якщо програму використовують люди без технічного досвіду. По-друге, залежність від Інтернет з'єднання для синхронізації та збереження даних може створювати проблеми у разі нестабільного доступу до мережі. Також існує ризик втрати конфіденційної інформації через недостатній рівень захисту або витік даних. Важливим недоліком може бути обмежена сумісність з іншими застосунками або платформами, що ускладнює інтеграцію у вже існуючі робочі процеси. Крім того, деякі програми можуть не враховувати індивідуальні потреби користувачів, пропонуючи надто універсальні рішення без гнучкості налаштувань. У певних випадках вартість підписки або ліцензії на програмне

забезпечення може бути перешкодою для широкого використання. Таким чином, незважаючи на численні переваги, ці недоліки можуть впливати на загальну ефективність і зручність використання такого програмного забезпечення. Тому актуальною є розробка програмного забезпечення для ведення щоденника та управління завданнями.

Сформульовано функціональні вимоги до програмної платформи для ведення щоденника та управління завданнями.

Для реалізації програмного забезпечення для ведення щоденника та управління завданнями обрано мову програмування PHP, яка є однією з найпоширеніших мов для створення вебзастосунків та добре підходить для обробки запитів, роботи з базами даних і управління авторизацією користувачів. Крім того, завдяки великій кількості готових фреймворків можна значно пришвидшити розробку, забезпечуючи при цьому високу структурованість коду, безпеку та масштабованість.

Для створення програмного забезпечення для ведення щоденника та управління завданнями обрано середовище розробки PhpStorm, що розроблене спеціально для роботи з PHP і суміжними вебтехнологіями, пропонує широкий спектр інструментів для написання, налагодження та тестування коду. Завдяки глибокій інтеграції з популярними фреймворками та системами управління базами даних вдається значно знизити час налаштування середовища та сконцентруватися безпосередньо на розробці функціоналу. PhpStorm забезпечує зручну навігацію по проєкту, автоматичне доповнення коду, статичний аналіз та миттєву підсвітку помилок, що допомагає уникати багатьох помилок на ранніх етапах.

Новизна роботи полягає у тому, що створено модель взаємодії користувача з програмним забезпеченням для ведення щоденника та управління завданнями, яка подана у вигляді діаграми прецедентів, та дозволяє розроблювати програмне забезпечення для ведення щоденника та управління завданнями.

Практична цінність полягає у тому, що розроблено програмну платформу для ведення щоденника та управління завданнями.

Запропоновано структуру програмної платформи для ведення щоденника та управління завданнями. Визначено, що основними класами програмної платформи для ведення щоденника та управління завданнями є Контакт, Аккаунт, Зустріч, Список зустрічей, Користувач, Адміністратор.

Функціонування програмної платформи для ведення щоденника та управління завданнями подамо за допомогою відповідних діаграм.

Архітектура вебзастосунку програмного забезпечення для ведення щоденника та управління завданнями побудована на основі шаблону Model-View-Controller. Описано функціонування програмної платформи для ведення щоденника та управління завданнями. Така архітектура забезпечує чітке розділення обов'язків між обробкою даних, логікою взаємодії та відображенням інформації. У цій структурі вся робота з записами щоденника та організацією завдань зосереджена у моделях, які відповідають за збереження стану, обробку інформації та зв'язок із сховищем даних.

Виконано проєктування інтерфейсу взаємодії користувача з програмним забезпеченням для ведення щоденника та управління завданнями.

Виконано тестування розробленого програмного забезпечення для ведення щоденника та управління завданнями. Результати тестування програмної платформи показали, що розроблена програма дозволяє забезпечити автоматизацію процесів, пов'язаних з підтримкою управління завданнями користувача.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. What is Diary Management? Tools and Techniques [Electronic resource]. – Access mode: <https://www.scheduleit.com/blog/11279/what-is-diary-management-tools-and-techniques>.
2. Diary Management Tips For Busy PAs [Electronic resource]. – Access mode: <https://www.orielpartners.co.uk/blog/diary-management-tips-for-pas>.
3. How to Diary Management – A Step-by-Step Guide to Success [Electronic resource]. – Access mode: <https://hogonext.com/how-to-diary-management-a-step-by-step-guide-to-success/>.
4. Our Top 9 Tips to Effectively Manage Your Diary in a Busy Office [Electronic resource]. – Access mode: <https://onwardsanswering.com/our-top-9-tips-to-effectively-manage-your-diary-in-a-busy-office/>.
5. The Complete Guide to Diary Management [Electronic resource]. – Access mode: <https://smart-pa.com/uk/blog/the-complete-guide-to-diary-management>.
6. The Guide to Diary Management [Electronic resource]. – Access mode: <https://www.secsinthecity.co.uk/article/the-guide-to-diary-management>.
7. Best Daily Planner Apps [Electronic resource]. – Access mode: <https://resourcepreneur.com/daily-planner-apps/>.
8. Task Management Apps for 2025 [Electronic resource]. – Access mode: <https://www.pcmag.com/picks/the-best-task-management-apps>.
9. The 13 Best Digital Journal Apps [Electronic resource]. – Access mode: <https://friday.app/p/best-digital-journal-apps>.
10. The best planning software for an efficient diary [Electronic resource]. – Access mode: <https://teambookapp.com/blog/the-best-planning-software-for-an-efficient-diary-in-2024/>.
11. Clockify [Electronic resource]. – Access mode: <https://clockify.me/>.
12. Habitica [Electronic resource]. – Access mode: <https://habitica.com/static/home>.

13. TwoBird [Electronic resource]. – Access mode:
<https://www.twobird.com/>.

14. PHP Syntax [Electronic resource]. – Access mode:
https://www.w3schools.com/php/php_syntax.asp.

15. What is PHP? The PHP Programming Language Meaning Explained
[Electronic resource]. – Access mode: <https://www.freecodecamp.org/news/what-is-php-the-php-programming-language-meaning-explained/>.

16. PhpStorm for Windows [Electronic resource]. – Access mode:
<https://phpstorm.en.softonic.com/>.

17. PhpStorm Features [Electronic resource]. – Access mode:
<https://www.jetbrains.com/phpstorm/features/>.

ДОДАТОК А
Технічне завдання

Вступ

Програмне забезпечення може використовуватися для ведення щоденника та управління завданнями.

A.1 Підстава для розробки

Підставою для розробки є завдання на дипломну кваліфікаційну роботу на тему «Дослідження та програмна реалізація платформи для ведення щоденника та управління завданнями», затверджене наказом Національного університету «Запорізька політехніка» № 447 від 30 вересня 2025 р.

A.2 Призначення розробки

Програмний продукт призначений для ведення щоденника та управління завданнями.

A.3 Основні вимоги до програми, що розробляється

A.3.1 Вимоги до функціональних характеристик

Програмне забезпечення для ведення щоденника та управління завданнями забезпечує виконання таких функцій:

- підтримка можливості ведення щоденника завдань та зустрічей користувача;
- підтримка можливості відображення завдань та зустрічей користувача у вигляді календаря подій;
- можливість отримання інформації про контакти користувача;
- можливість редагування інформації про зустрічі користувача;
- підтримка можливості доступу до адміністративної панелі програмної платформи для редагування даних системи.

А.3.2 Вимоги до інтерфейсу програми

Інтерфейс програмної платформи для ведення щоденника та управління завданнями повинен бути зручним для користувачів. Повинна бути забезпечена можливість роботи в візуальному режимі.

А.3.3 Вимоги до надійності

Програмна платформа для ведення щоденника та управління завданнями повинно забезпечити надійне функціонування.

А.3.4 Умови експлуатації

Для експлуатації програмної платформи для ведення щоденника та управління завданнями необхідна наявність персонального комп'ютера.

А.3.5 Вимоги до складу та параметрів технічний засобів

Для роботи програмної платформи для ведення щоденника та управління завданнями потрібні такі технічні та програмні ресурси: комп'ютер або сервер із процесором із тактовою частотою не нижче 2,0 ГГц (рекомендується багатоядерний), обсяг оперативної пам'яті не менше 4 ГБ для мінімальної конфігурації та від 8 ГБ для стабільної роботи при одночасній роботі кількох користувачів. Вільний простір на диску має складати не менше 500 МБ для самої платформи та резервувати додаткове місце під зберігання даних і резервні копії.

Із програмного боку необхідна операційна система, сумісна з PHP-середовищем, можна використовувати Linux або Windows Server. Обов'язково

потрібен вебсервер (Apache чи Nginx), встановлений PHP з відповідними модулями, а також система керування базами даних, наприклад, MySQL. Для локального розгортання чи тестування може знадобитися встановлення додаткового ПЗ для адміністрування баз даних (phpMyAdmin або інше). Також важливо передбачити браузер сучасної версії для користувачів, щоб інтерфейс програми коректно відображався й працював без помилок.

А.3.6 Вимоги до маркування і пакування

Програма для ведення щоденника та управління завданнями може бути записана на будь-якому носії інформації.

На пакуванні повинна бути назва програми – «Програмна платформа для ведення щоденника та управління завданнями».

А.4 Вхідні дані до роботи

Вхідними даними до програмної платформи для ведення щоденника та управління завданнями є інформація про користувача платформи, його зустрічі та контакти.

Вихідними даними програмної платформи для ведення щоденника та управління завданнями є календар зустрічей, вебсторінки порталу ведення щоденника.

ДОДАТОК Б
Фрагмент тексту програми

```

class DB {
    private $hst = 'localhost';
    private $usr = 'root';
    private $pss = '';
    private $dbn = 'djrnlts';
    private $lnk;

    public function __construct() {
        $this->lnk = new mysqli($this->hst, $this->usr, $this->
        >pss, $this->dbn);
        if ($this->lnk->connect_error) {
            die('DB CNCT FLD');
        }
    }

    public function qry($q) {
        return $this->lnk->query($q);
    }

    public function esc($str) {
        return $this->lnk->real_escape_string($str);
    }

    public function clse() {
        $this->lnk->close();
    }
}

class CCNT {
    private $db;

    public function __construct($db) {
        $this->db = $db;
    }

    public function crt($nm, $psw) {
        $nm = $this->db->esc($nm);
        $psw = password_hash($psw, PASSWORD_DEFAULT);
        $q = "INSERT INTO ccnt (nm, psw) VALUES ('$nm', '$psw')";
        return $this->db->query($q);
    }

    public function lg($nm, $psw) {
        $nm = $this->db->esc($nm);
        $q = "SELECT * FROM ccnt WHERE nm='$nm'";
    }
}

```

```

        $r = $this->db->qry($q);
        if ($r->num_rows == 1) {
            $rw = $r->fetch_assoc();
            if (password_verify($psw, $rw['psw'])) {
                return $rw['id'];
            }
        }
        return false;
    }
}

class NT {
    private $db;

    public function __construct($db) {
        $this->db = $db;
    }

    public function crt($ccnt_id, $ttl, $cntnt) {
        $ttl = $this->db->esc($ttl);
        $cntnt = $this->db->esc($cntnt);
        $q = "INSERT INTO nt (ccnt_id, ttl, cntnt) VALUES
($ccnt_id, '$ttl', '$cntnt')";
        return $this->db->qry($q);
    }

    public function get($ccnt_id) {
        $q = "SELECT * FROM nt WHERE ccnt_id=$ccnt_id";
        return $this->db->qry($q);
    }
}

class TSK {
    private $db;

    public function __construct($db) {
        $this->db = $db;
    }

    public function crt($ccnt_id, $dsc) {
        $dsc = $this->db->esc($dsc);
        $q = "INSERT INTO tsk (ccnt_id, dsc) VALUES ($ccnt_id,
'$dsc')";
        return $this->db->qry($q);
    }
}

```

```

public function get($ccnt_id) {
    $q = "SELECT * FROM tsk WHERE ccnt_id=$ccnt_id";
    return $this->db->qry($q);
}

public function dn($tsk_id) {
    $q = "UPDATE tsk SET stts='cmpltd' WHERE id=$tsk_id";
    return $this->db->qry($q);
}
}

class CNTRLR {
    private $db;
    private $ccnt;
    private $nt;
    private $tsk;

    public function __construct() {
        $this->db = new DB();
        $this->ccnt = new CCNT($this->db);
        $this->nt = new NT($this->db);
        $this->tsk = new TSK($this->db);
    }

    public function rstr($nm, $psw) {
        return $this->ccnt->crt($nm, $psw);
    }

    public function lgn($nm, $psw) {
        return $this->ccnt->lg($nm, $psw);
    }

    public function add_nt($ccnt_id, $ttl, $cntnt) {
        return $this->nt->crt($ccnt_id, $ttl, $cntnt);
    }

    public function vw_nt($ccnt_id) {
        return $this->nt->get($ccnt_id);
    }

    public function add_tsk($ccnt_id, $dsc) {
        return $this->tsk->crt($ccnt_id, $dsc);
    }

    public function vw_tsk($ccnt_id) {
        return $this->tsk->get($ccnt_id);
    }
}

```

```

    }

    public function cmplt_tsk($tsk_id) {
        return $this->tsk->dn($tsk_id);
    }

    public function __destruct() {
        $this->db->close();
    }
}

$cntlr = new CNTRLR();

if (isset($_POST['rstr'])) {
    $cntlr->rstr($_POST['nm'], $_POST['psw']);
    echo 'RSTR SCTSSFLLY';
}

if (isset($_POST['lgn'])) {
    $id = $cntlr->lgn($_POST['nm'], $_POST['psw']);
    if ($id) {
        session_start();
        $_SESSION['ccnt_id'] = $id;
        echo 'LGN SCTSSFLLY';
    } else {
        echo 'LGN FLD';
    }
}

if (isset($_POST['add_nt'])) {
    session_start();
    $cntlr->add_nt($_SESSION['ccnt_id'], $_POST['ttl'],
$_POST['cntnt']);
    echo 'NT CRTD';
}

if (isset($_POST['add_tsk'])) {
    session_start();
    $cntlr->add_tsk($_SESSION['ccnt_id'], $_POST['dsc']);
    echo 'TSK CRTD'
}

$hst = 'localhost';
$usr = 'root';
$pss = '';
$dbn = 'djrnltmng';

```

```

$conn = new mysqli($hst, $usr, $pss, $dbn);

if ($conn->connect_error) {
    die("FLD CNNT CT: " . $conn->connect_error);
}

function crt_tbls($c) {
    $sql1 = "CREATE TABLE IF NOT EXISTS usr (
        id INT AUTO_INCREMENT PRIMARY KEY,
        nm VARCHAR(255) NOT NULL,
        pwd VARCHAR(255) NOT NULL
    )";
    $sql2 = "CREATE TABLE IF NOT EXISTS nt (
        id INT AUTO_INCREMENT PRIMARY KEY,
        ttl VARCHAR(255) NOT NULL,
        cntnt TEXT,
        dt DATETIME,
        usr_id INT
    )";
    $sql3 = "CREATE TABLE IF NOT EXISTS tsk (
        id INT AUTO_INCREMENT PRIMARY KEY,
        ttl VARCHAR(255) NOT NULL,
        dscripntn TEXT,
        stts VARCHAR(50),
        ddln DATETIME,
        usr_id INT
    )";

    $c->query($sql1);
    $c->query($sql2);
    $c->query($sql3);
}

crt_tbls($conn);

function rgstr($nm, $pwd, $c) {
    $hpwd = password_hash($pwd, PASSWORD_DEFAULT);
    $stmt = $c->prepare("INSERT INTO usr (nm, pwd) VALUES (?,
    ?)");
    $stmt->bind_param("ss", $nm, $hpwd);
    return $stmt->execute();
}

function lg($nm, $pwd, $c) {
    $stmt = $c->prepare("SELECT id, pwd FROM usr WHERE nm=?");

```

```

$stmt->bind_param("s", $nm);
$stmt->execute();
$stmt->store_result();
if ($stmt->num_rows > 0) {
    $stmt->bind_result($id, $hpwd);
    $stmt->fetch();
    if (password_verify($pwd, $hpwd)) {
        return $id;
    }
}
return false;
}

function add_nt($ttl, $cntnt, $dt, $usr_id, $c) {
    $stmt = $c->prepare("INSERT INTO nt (ttl, cntnt, dt, usr_id)
VALUES (?, ?, ?, ?)");
    $stmt->bind_param("sssi", $ttl, $cntnt, $dt, $usr_id);
    return $stmt->execute();
}

function add_tsk($ttl, $dscrptn, $stts, $ddltn, $usr_id, $c) {
    $stmt = $c->prepare("INSERT INTO tsk (ttl, dscrptn, stts,
ddltn, usr_id) VALUES (?, ?, ?, ?, ?)");
    $stmt->bind_param("ssssi", $ttl, $dscrptn, $stts, $ddltn,
$usr_id);
    return $stmt->execute();
}

function gt_nts($usr_id, $c) {
    $stmt = $c->prepare("SELECT * FROM nt WHERE usr_id=? ORDER BY
dt DESC");
    $stmt->bind_param("i", $usr_id);
    $stmt->execute();
    return $stmt->get_result();
}

function gt_tsk($usr_id, $c) {
    $stmt = $c->prepare("SELECT * FROM tsk WHERE usr_id=? ORDER
BY ddltn ASC");
    $stmt->bind_param("i", $usr_id);
    $stmt->execute();
    return $stmt->get_result();
}

function rmv_nt($id, $usr_id, $c) {

```

```

        $stmt = $c->prepare("DELETE FROM nt WHERE id=? AND
usr_id=?");
        $stmt->bind_param("ii", $id, $usr_id);
        return $stmt->execute();
    }

function rmv_tsk($id, $usr_id, $c) {
    $stmt = $c->prepare("DELETE FROM tsk WHERE id=? AND
usr_id=?");
    $stmt->bind_param("ii", $id, $usr_id);
    return $stmt->execute();
}

function upd_tsk($id, $ttl, $dscrptn, $stts, $ddln, $usr_id, $c)
{
    $stmt = $c->prepare("UPDATE tsk SET ttl=?, dscrptn=?, stts=?,
ddln=? WHERE id=? AND usr_id=?");
    $stmt->bind_param("ssssii", $ttl, $dscrptn, $stts, $ddln,
$id, $usr_id);
    return $stmt->execute();
}

session_start();
$curr_usr = null;

if (isset($_POST['actn'])) {
    switch ($_POST['actn']) {
        case 'rgstr':
            rgstr($_POST['nm'], $_POST['pwd'], $conn);
            break;
        case 'lg':
            $id = lg($_POST['nm'], $_POST['pwd'], $conn);
            if ($id) {
                $_SESSION['usr_id'] = $id;
            }
            break;
        case 'add_nt':
            if (isset($_SESSION['usr_id'])) {
                add_nt($_POST['ttl'], $_POST['cntnt'],
$_POST['dt'], $_SESSION['usr_id'], $conn);
            }
            break;
        case 'add_tsk':
            if (isset($_SESSION['usr_id'])) {
                add_tsk($_POST['ttl'], $_POST['dscrptn'],
$_POST['stts'], $_POST['ddln'], $_SESSION['usr_id'], $conn);
            }
    }
}

```

```

        }
        break;
    case 'rmv_nt':
        if (isset($_SESSION['usr_id'])) {
            rmv_nt($_POST['id'], $_SESSION['usr_id'], $conn);
        }
        break;
    case 'rmv_tsk':
        if (isset($_SESSION['usr_id'])) {
            rmv_tsk($_POST['id'], $_SESSION['usr_id'],
$conn);
        }
        break;
    case 'upd_tsk':
        if (isset($_SESSION['usr_id'])) {
            upd_tsk($_POST['id'], $_POST['ttl'],
$_POST['dscrptn'], $_POST['stts'], $_POST['ddln'],
$_SESSION['usr_id'], $conn);
        }
        break;
    case 'lgut':
        session_destroy();
        break;
    }
}

if (isset($_SESSION['usr_id'])) {
    $curr_usr = $_SESSION['usr_id'];
}

$(document).ready(function() {
    var klndr = $('#klndr').fullKlndr({
        editable:true,
        header:{
            left:'prev,next today',
            center:'ttl',
            right:'mnth,agndWeek,agndDay'
        },
        evnts: 'load.php',
        slktable:true,
        slktHelper:true,
        slkt: function(strt, knc, allDay)
        {
            var ttl = prompt("Enter Evnt Ttl");
            if(ttl)
            {

```

```

        var strt = $.fullKlndr.formatDate(strt, "Y-MM-DD
HH:mm:ss");
        var knc = $.fullKlndr.formatDate(knc, "Y-MM-DD HH:mm:ss");
        $.ajax({
            url:"insert.php",
            type:"POST",
            data:{ttl:ttl, strt:strt, knc:knc},
            success:function()
            {
                klndr.fullKlndr('refetchEvnts');
                alert("Added Successfully");
            }
        })
    },
    editable:true,
    evntResize:function(evnt)
    {
        var strt = $.fullKlndr.formatDate(evnt.strt, "Y-MM-DD
HH:mm:ss");
        var knc = $.fullKlndr.formatDate(evnt.knc, "Y-MM-DD
HH:mm:ss");
        var ttl = evnt.ttl;
        var id = evnt.id;
        $.ajax({
            url:"update.php",
            type:"POST",
            data:{ttl:ttl, strt:strt, knc:knc, id:id},
            success:function(){
                klndr.fullKlndr('refetchEvnts');
                alert('Evnt Update');
            }
        })
    },

    evntDrop:function(evnt)
    {
        var strt = $.fullKlndr.formatDate(evnt.strt, "Y-MM-DD
HH:mm:ss");
        var knc = $.fullKlndr.formatDate(evnt.knc, "Y-MM-DD
HH:mm:ss");
        var ttl = evnt.ttl;
        var id = evnt.id;
        $.ajax({
            url:"update.php",
            type:"POST",

```

```

    data:{ttl:ttl, strt:strtt, knc:knc, id:id},
    success:function()
    {
        klndr.fullKlndr('refetchEvnts');
        alert("Evnt Updated");
    }
    });
},

evntClick:function(evnt)
{
    if(confirm("Hochesh vydalyty?"))
    {
        var id = evnt.id;
        $.ajax({
            url:"delete.php",
            type:"POST",
            data:{id:id},
            success:function()
            {
                klndr.fullKlndr('refetchEvnts');
                alert("Evnt Removed");
            }
        })
    }
},

});
});
}

```

ДОДАТОК В
Слайди презентації

Міністерство освіти і науки України
Національний університет «Запорізька політехніка»
Кафедра програмних засобів

Дипломна кваліфікаційна робота

**Дослідження та програмна реалізація
платформи для ведення щоденника та
управління завданнями**

Виконав
ст. гр. КНТ-114м

Сергій ЛЮБАС

Керівник
к.т.н., доцент

Тетяна КОЛПАКОВА

Рисунок В.1 – Слайд 1

Об'єкт, предмет та мета роботи

Об'єктом дослідження є процес розробки програмного забезпечення для ведення щоденника та управління завданнями.

Предмет дослідження – програмні засоби для ведення щоденника та управління завданнями.

Мета роботи – дослідження та розробка програмної платформи для ведення щоденника та управління завданнями.

2

Рисунок В.2 – Слайд 2

Задачі роботи:

Для досягнення поставленої мети у кваліфікаційній роботі магістра необхідно розв'язати такі задачі:

- виконати аналіз предметної області та програмних засобів для ведення персональних щоденників та управління завданнями;
- розробити модель взаємодії користувача з програмним забезпеченням для ведення щоденника та управління завданнями;
- здійснити проектування програмного забезпечення для ведення щоденника та управління завданнями;
- створити програмної платформи для ведення щоденника та управління завданнями;
- виконати тестування розробленої програмної платформи для ведення щоденника та управління завданнями.

3

Рисунок В.3 – Слайд 3

Програмні засоби для ведення щоденника та управління завданнями

Критерій порівняння	Clockify	Habitica	TwoBird
Інтеграція з календарями	+	+-	+
Робота з поштою	+-	-	+
Мотиваційні механізми	+-	+	-
Простота інтерфейсу	+	+-	+-
Зручність для списків справ	+-	+-	+
Можливість нагадувань	+-	+	+

4

Рисунок В.4 – Слайд 4

Вибір мови програмування

Критерій порівняння мов програмування	Мова програмування		
	PHP	Python	JavaScript
Швидкість розробки	+	+-	+-
Легкість інтеграції з СУБД	+	+	+-
Простота розгортання на більшості хостингів	+	+-	+-
Зручність для створення ПЗ для ведення щоденника та управління завданнями	+	-	-
Велика спільнота і кількість рішень	+	+	+-

5

Рисунок В.5 – Слайд 5

Порівняльний аналіз середовищ розробки

Критерій порівняння середовищ розробки	Середовища розробки		
	PhpStorm	Visual Studio Code	Sublime Text
Інтеграція з фреймворками	+	+-	-
Автоматизація та підказки при написанні коду	+	+	+-
Простота інтерфейсу	+	-	+
Наявність вбудованих інструментів налагодження	+	+	-
Зручність роботи з базами даних	+	+-	-

6

Рисунок В.6 – Слайд 6

Модель взаємодії користувача з ПЗ для ведення щоденника та управління завданнями

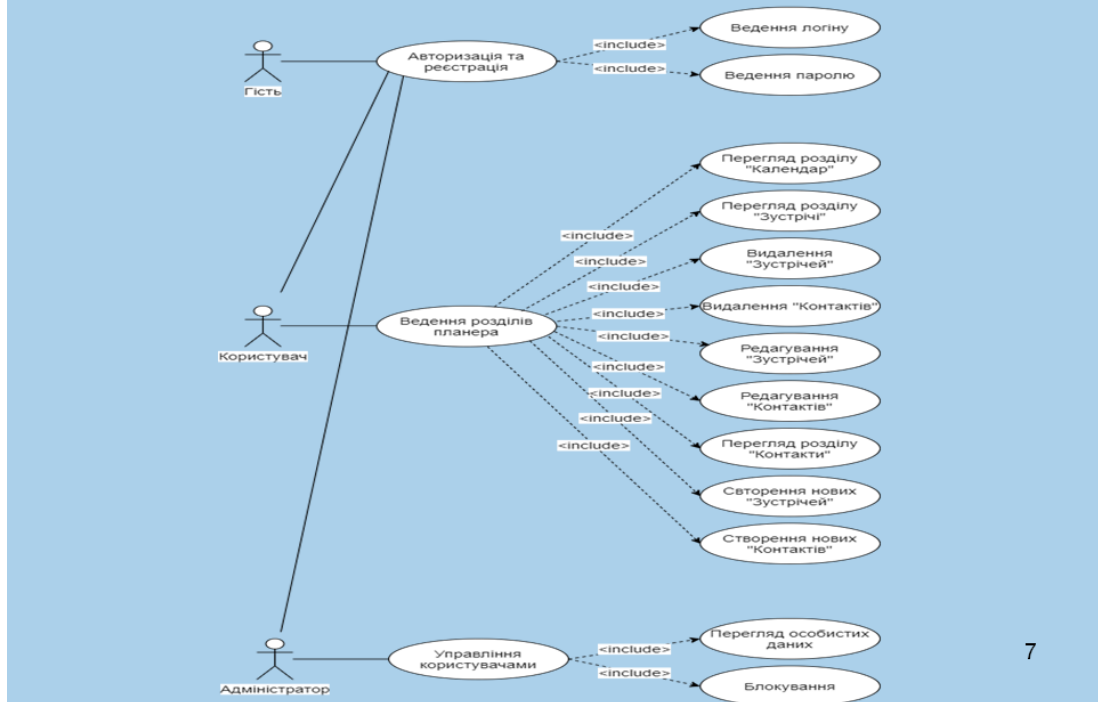


Рисунок В.7 – Слайд 7

Діаграма класів програмного забезпечення

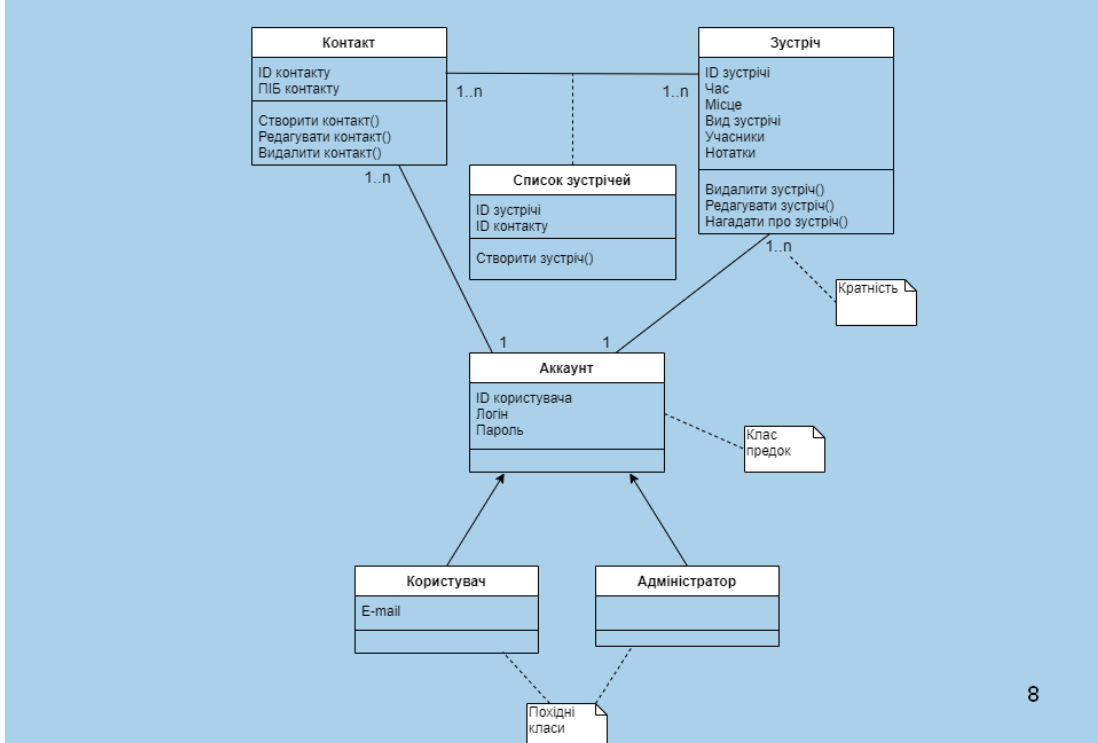


Рисунок В.8 – Слайд 8

Діаграма компонентів ПЗ для ведення щоденника та управління завданнями

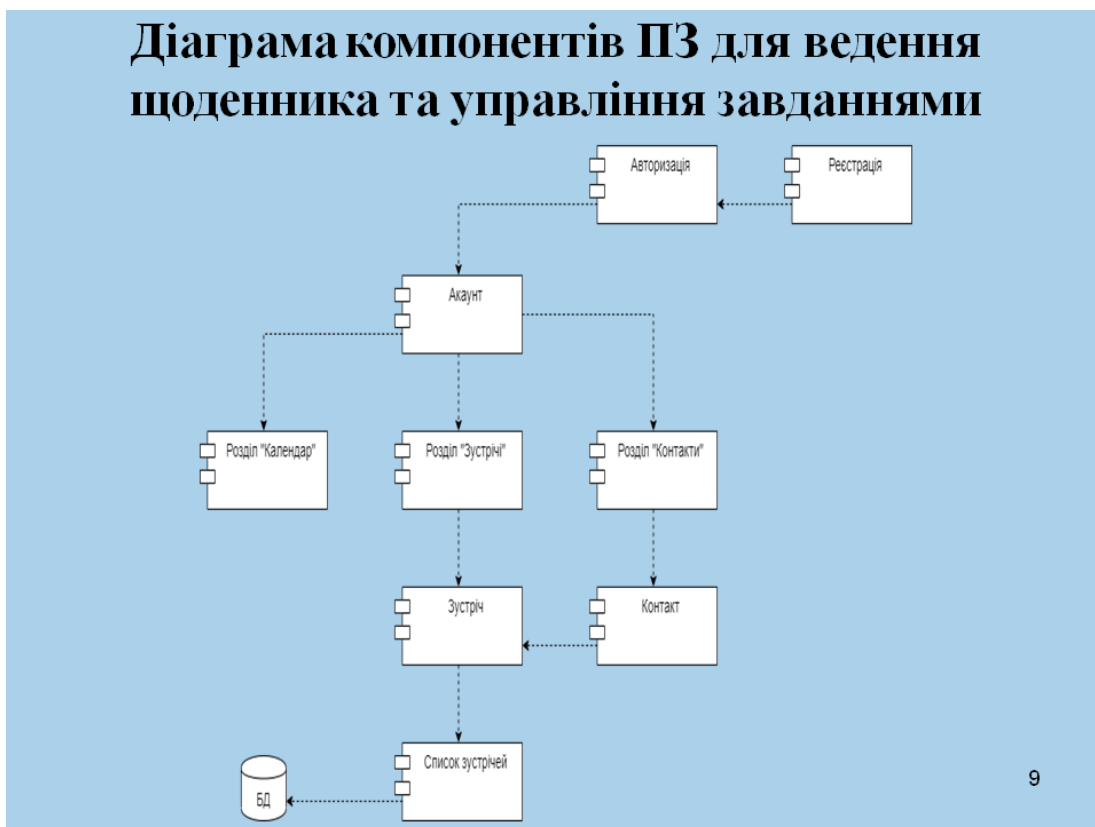


Рисунок В.9 – Слайд 9

Функціонування програмної платформи для ведення щоденника та управління завданнями

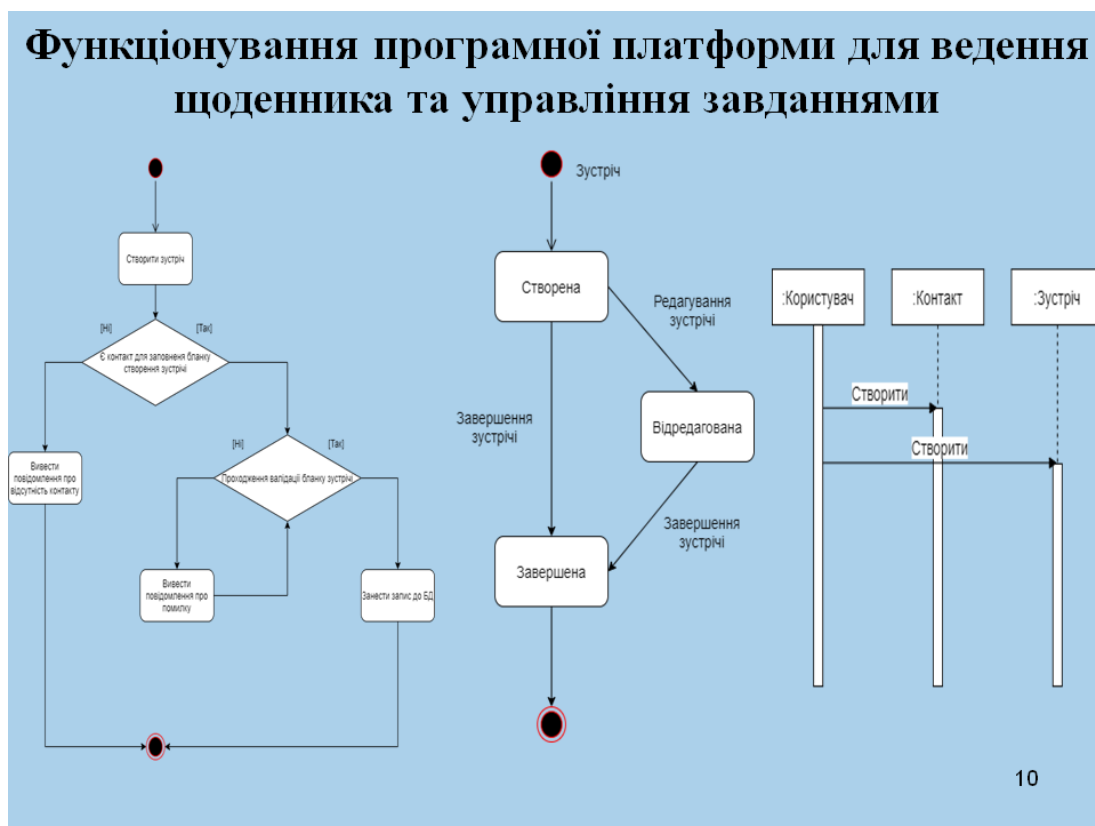
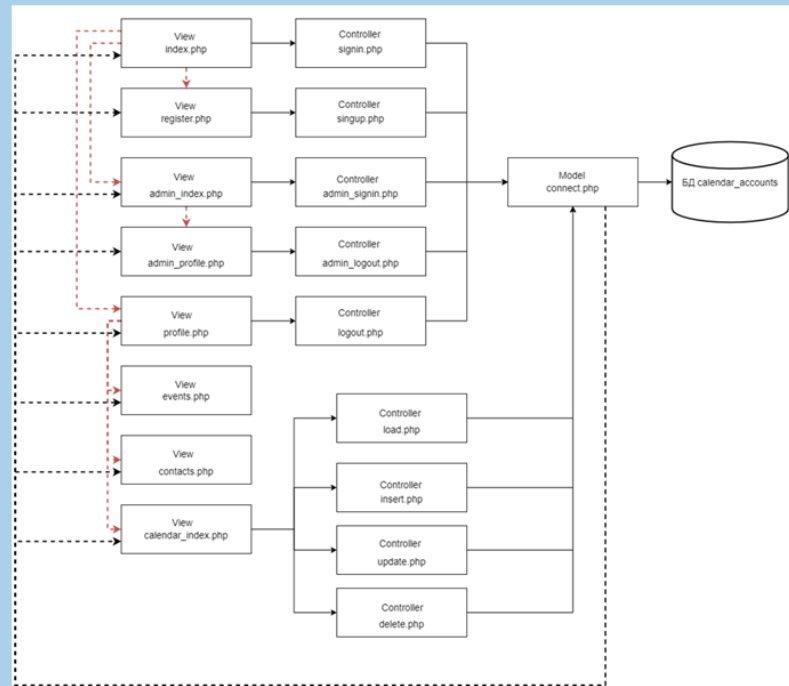


Рисунок В.10 – Слайд 10

Схема архітектури ПЗ для ведення щоденника та управління завданнями



11

Рисунок В.11 – Слайд 11

Робота з програмою. Вхід до програми. Сторінка профілю користувача

Програмна платформа для ведення щоденника та управління завданнями

Вхід до платформи

Логін
Введіть свій Логін

Пароль
Введіть свій Пароль

Увійти

Немає акаунту? Тоді необхідно зареєструватися

Програмна платформа для ведення щоденника та управління завданнями

Регістрація

Ім'я
Введіть Ім'я

Логін
Введіть Логін

Електронна пошта
Введіть свою електронну пошту

Файл з зображенням
Файл Файл не обрано

Пароль
Введіть пароль

Перевірка паролю
Введіть пароль (ще раз)

Зареєструватися

Вже є акаунт? Заходь сюди

My name: Chik
My email: Chik59354@gmail.com

Мій календар

Контакти

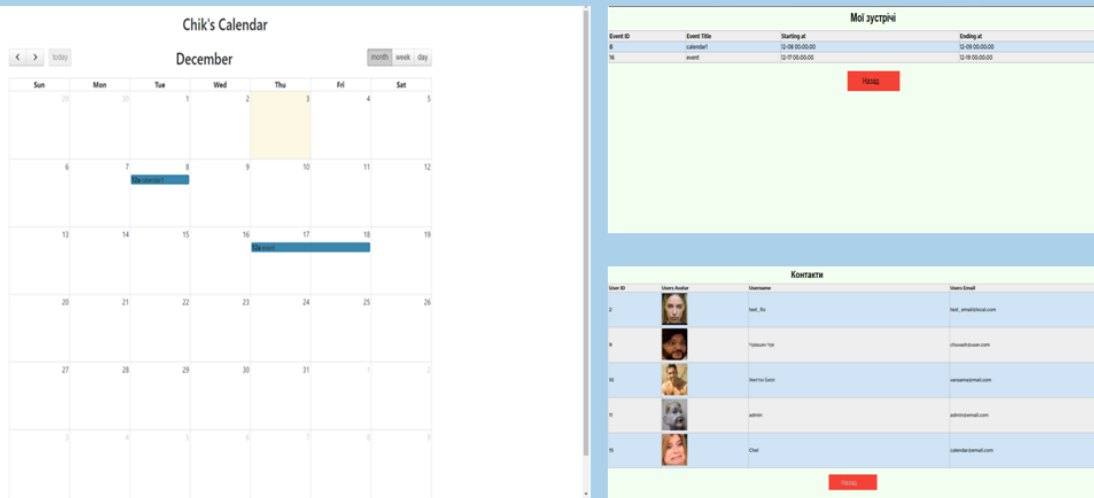
Зустрічі

Вихід

12

Рисунок В.12 – Слайд 12

Робота з програмою. Календар щоденника подій. Контакти та зустрічі



13

Рисунок В.13 – Слайд 13

Висновки

В ході виконання даної дипломної кваліфікаційної роботи було проаналізовано та досліджено процес розробки програмного забезпечення для ведення щоденника та управління завданнями.

Новизна роботи полягає у тому, що створено модель взаємодії користувача з програмним забезпеченням для ведення щоденника та управління завданнями, яка подана у вигляді діаграми прецедентів, та дозволяє розроблювати програмне забезпечення для ведення щоденника та управління завданнями.

Практична цінність полягає у тому, що розроблено програмну платформу для ведення щоденника та управління завданнями.

Після проведення тестування встановлено, що програмне забезпечення для ведення щоденника та управління завданнями відповідає вимогам технічного завдання та може використовуватися за призначенням.

14

Рисунок В.14 – Слайд 14