

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 проекту (роботи)

магістр

(ступінь вищої освіти)

на тему ДОСЛІДЖЕННЯ ТА ПРОГРАМНА РЕАЛІЗАЦІЯ
ЗАСТОСУНКУ ДЛЯ ПЛАНУВАННЯ ТА
ОРГАНІЗАЦІЇ ЩОДЕННИХ СПРАВ
RESEARCH AND SOFTWARE IMPLEMENTATION OF AN APPLICATION
FOR DAILY TASK SCHEDULING AND 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

ГАЛИШ О.С.

(ПРИЗВИЩЕ та ініціали)

Керівник ОЛІЙНИК А.О.

(ПРИЗВИЩЕ та ініціали)

Рецензент СКРУПСЬКИЙ С.Ю.

(ПРИЗВИЩЕ 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)

ГАЛИША Олега Сергійовича

(ПРИЗВИЩЕ, ім'я, по батькові)

1. Тема проєкту (роботи) Дослідження та програмна реалізація застосунку для
планування та організації щоденних справ. Research and Software
Implementation of an Application for Daily Task Scheduling and Management

керівник проєкту (роботи) д.т.н., професор, ОЛІЙНИК Андрій Олександрович ,
(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	ПРІЗВИЩЕ, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-4 Основна частина	ОЛІЙНИК А.О., професор		
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) Олег ГАЛИШ
(Ім'я ПРІЗВИЩЕ)

Керівник проєкту (роботи)

(підпис) Андрій ОЛІЙНИК
(Ім'я ПРІЗВИЩЕ)

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи магістра:
99 с., 3 табл., 39 рис., 3 дод., 20 джерел.

С#, VISUAL STUDIO, ЗАСТОСУНОК, МОВА ПРОГРАМУВАННЯ,
МОДЕЛЬ, ПЛАНУВАННЯ ДІЯЛЬНОСТІ, ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ.

Об'єкт дослідження – процес розробки програмного забезпечення для планування та організації щоденних справ.

Предмет дослідження – програмні засоби для планування та організації щоденних справ.

Мета роботи – дослідження та розробка програмного застосунку для планування та організації щоденних справ.

Матеріали, методи та технічні засоби: мова програмування С#, середовище розробки Visual Studio.

Результати. Розроблено програмне забезпечення для планування та організації щоденних справ за допомогою мови програмування С# та середовища розробки Visual Studio.

Практична цінність роботи полягає у розробці програмного забезпечення для планування та організації щоденних справ.

Висновки. Виконано проєктування програмного забезпечення для планування та організації щоденних справ. Розроблено програмне забезпечення для планування та організації щоденних справ. Здійснено тестування розробленого програмного забезпечення для планування та організації щоденних справ.

Галузь використання – індивідуальне планування діяльності.

ABSTRACT

Explanatory note to the diploma qualifying work of the master: 99 pages, 3 tables, 39 figures, 3 appendixes, 20 sources.

C#, VISUAL STUDIO, APPLICATION, PROGRAMMING LANGUAGE, MODEL, PLANNING OF ACTIVITIES, SOFTWARE.

The object of research is the process of developing software for planning and organizing daily activities.

The subject of the research is software for planning and organizing daily activities.

The purpose of this work is to develop the software for planning and organizing daily activities.

Materials, methods and technical tools: C# programming language, Visual Studio development environment.

Results. The software tools for planning and organizing daily activities using the C# programming language and the Visual Studio development environment have been developed.

The practical value of the work lies in the development of software for planning and organizing daily activities.

Conclusions. The software for planning and organizing daily activities has been designed. The software for planning and organizing daily activities has been developed. The developed software for planning and organizing daily activities has been tested.

Scope of use – individual activity planning.

ЗМІСТ

	С.
Перелік скорочень та умовних познач	8
Вступ.....	9
1 Аналіз предметної області.....	11
1.1 Програмне забезпечення для планування та організації щоденних справ.....	11
1.2 Аналіз програмного забезпечення для планування та організації щоденних справ.....	20
1.3 Висновки за розділом 1	28
2 Матеріали і методи	30
2.1 Вибір мови програмування	30
2.2 Вибір середовища розробки для створення програмного забезпечення для планування та організації щоденних справ.....	32
2.3 Розробка моделі взаємодії користувача з програмним забезпеченням для планування та організації щоденних справ.....	36
2.4 Висновки за розділом 2	38
3 Опис програми	39
3.1 Структура програмного забезпечення для планування та організації щоденних справ.....	39
3.2 Функціонування програмного застосунку для планування та організації щоденних справ.....	43
3.3 Розробка бази даних програмного забезпечення для планування та організації щоденних справ	46
3.4 Проєктування інтерфейсу програмного забезпечення для планування та організації щоденних справ	48
3.5 Висновки за розділом 3	51
4 Експлуатація, тестування та експериментальне дослідження програми.....	53
4.1 Призначення й умови застосування програми	53

4.2 Характеристики програми для планування та організації щоденних справ.....	55
4.3 Інструкція по експлуатації програми.....	56
4.3.1 Звернення до програми.....	56
4.3.2 Вхідні й вихідні дані	57
4.3.3 Повідомлення.....	57
4.4 Виконання програмного забезпечення для планування та організації щоденних справ.....	57
4.5 Тестування програмного забезпечення для планування та організації щоденних справ.....	61
4.6 Висновки за розділом 4	63
Висновки	64
Перелік джерел посилання	67
Додаток А Технічне завдання	69
Додаток Б Фрагмент тексту програми	73
Додаток В Слайди презентації	91

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

MVC – Model-View-Controller;

БД – база даних;

ПЗ – програмне забезпечення;

ЩС – щоденні справи.

ВСТУП

У сучасних умовах швидкого ритму життя та постійного інформаційного навантаження дедалі більшої ваги набуває здатність ефективно розподіляти час і ресурси. Велика кількість завдань, термінів виконання, зустрічей і побутових обов'язків потребує системного підходу до організації повсякденних справ. Традиційні методи планування, як-от паперові щоденники чи звичайні списки справ, часто поступаються сучасним програмним інструментам, що забезпечують мобільність, автоматизацію нагадувань і синхронізацію між різними пристроями [1], [2].

Програмне забезпечення для планування дозволяє не лише створювати та зберігати плани, а й контролювати їх виконання, аналізувати витрати часу, інтегруватися з календарями, електронною поштою та іншими сервісами. Такий функціонал сприяє підвищенню особистої продуктивності, зниженню рівня стресу та оптимізації робочих і особистих процесів [3], [4].

Поширення дистанційної роботи, гнучких графіків та мультизадачності додатково підсилює попит на ефективні програмні рішення для планування. Таким чином, створення та вдосконалення програмного забезпечення для організації повсякденних справ залишається актуальним завданням, яке відповідає сучасним викликам і потребам як окремих користувачів, так і організацій [5], [6].

Проте деякі програми для планування та організації щоденних справ мають певні недоліки. Однією з проблем є те, що часто такі програмні інструменти вимагають додаткового часу на налаштування та регулярне оновлення даних, що може стати рутинним і навіть обтяжливим для користувача. Деякі програми надто складні у використанні або перенасичені функціями, які не завжди потрібні, що іноді призводить до перевантаження інтерфейсу і втрати мотивації їх використовувати. Окремою проблемою є ризики, пов'язані з конфіденційністю особистої інформації, оскільки дані про зустрічі, контакти чи завдання часто зберігаються у хмарних сервісах і можуть

бути вразливими до несанкціонованого доступу. Крім того, надмірне використання цифрових програмних планувальників може призвести до втрати навичок самоорганізації без допоміжних інструментів і сформуванати залежність від автоматизованих нагадувань [1]-[6].

Тому актуальною є розробка програмного забезпечення для планування та організації щоденних справ. У дипломній кваліфікаційній роботі магістра розв'язується актуальне завдання розробки та дослідження програмного забезпечення для планування та організації щоденних справ.

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

- виконати аналіз предметної області та програмних засобів для планування та організації щоденних справ;
- розробити модель взаємодії користувача з програмним забезпеченням для планування та організації щоденних справ;
- здійснити проектування програмного забезпечення для планування та організації щоденних справ;
- створити програмне забезпечення для планування та організації щоденних справ;
- виконати тестування розробленого програмного забезпечення для планування та організації щоденних справ.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Програмне забезпечення для планування та організації щоденних справ

Програмне забезпечення для планування та організації щоденних справ використовується для керування особистим та робочим розпорядком, допомагаючи користувачеві структурувати завдання, контролювати хід їх виконання та своєчасно реагувати на важливі події. Такі програмні застосунки зазвичай поєднують у собі можливості керування списками справ, інтеграцію календаря, а також засоби нагадувань і сповіщень, завдяки яким можна уникнути пропущених граничних строків виконання задач чи незавершених пунктів плану. Окрім базового планування, багато рішень передбачають підтримку хмарного зберігання даних, що дозволяє отримувати доступ до власного розкладу з різних пристроїв. У деяких випадках додається можливість вести нотатки, працювати з файлами чи контактами, що робить такі інструменти універсальними у повсякденному використанні. Важливою перевагою сучасних рішень є інтуїтивний інтерфейс, що дає змогу швидко освоїтись навіть тим, хто раніше не користувався подібними сервісами. Завдяки цьому планувальники стають незамінними помічниками для тих, хто прагне тримати під контролем власні справи й розвивати ефективні звички самоорганізації [1].

Організація повсякденних справ та контроль за їх виконанням потребують ретельного підходу до планування і чіткого поділу обов'язків, особливо коли йдеться не лише про власні завдання, а й про роботу в команді. Щоденне впорядкування справ базується на тому, щоб вчасно визначати, що саме потребує уваги, відстежувати стан виконання і правильно розподіляти зусилля протягом робочого дня. Програмні засоби для планування покликані не обмежуватись простим списком завдань, а давати змогу керувати всім обсягом роботи більш системно: від моменту постановки окремої задачі до моменту, коли вона перевірена або закрита. Такий інструмент допомагає не

втрачати контроль над термінами, бачити взаємозв'язки між діями, розуміти, які етапи вимагають більше ресурсів чи часу. Для ефективного впровадження плану важливо мати змогу оновлювати дані, відстежувати зміни та фіксувати результати, що особливо актуально при роботі з великою кількістю дрібних кроків, які складаються у загальний план. Використання цифрових рішень дозволяє уникнути хаосу, мінімізує ризик втрати важливої інформації, забезпечує зручний доступ до актуального стану справ і відкриває простір для командної взаємодії. Кожен учасник процесу отримує можливість розуміти свої обов'язки, а також бачити загальну картину, що суттєво підвищує дисципліну виконання. Правильна робота з такими засобами перетворює нескінченні списки у чітко структуровану систему з можливістю пріоритезації, розподілу завантаження та гнучкого реагування на зміни. У результаті вдається досягти більшої продуктивності без зайвого стресу й уникнути ситуацій, коли важливі справи залишаються поза увагою [1]-[3].

Розпланований розклад допомагає впорядкувати щоденні справи та уникнути відчуття, що все виходить з-під контролю. Завдяки чіткому графіку легше дотримуватися структури, менше хвилюватися і залишатися продуктивним навіть у насичені дні. Якщо такий план створюється у цифровому форматі, стає доступною можливість швидко вносити корективи та стежити за змінами без плутанини й затримок. Видима схема дня дозволяє бачити всі справи одразу, що спрощує підготовку до виконання важливого. Такий підхід полегшує планування навіть тоді, коли немає натхнення або енергії для максимальних результатів. Привчання до постійного щоденного розкладу дозволяє залишатися організованим у будь-яких обставинах. Використання готового макету графіка полегшує справу: досить створити універсальну основу, а потім налаштовувати її під конкретну дату, додаючи нові завдання і викреслюючи виконане [1]-[3].

Один і той самий шаблон можна тримати у програмному інструменті, що зберігає всі зміни й оновлюється в реальному часі. У такому плануванні доречно тримати як регулярні пункти, що повторюються день у день, так і

разові справи з конкретними строками. Якщо не все відомо заздалегідь, частину завдань можна доповнити пізніше [4], [5].

Щоб уникнути хаосу, справам варто надавати різну важливість, визначаючи, що має найбільший вплив саме зараз. Це допомагає не розпорошувати зусилля на дрібниці й зосереджуватися на тих справах, які неможливо відкласти без шкоди для результату. Розуміння того, які строки встановлені для кожного пункту, дозволяє не упускати термінів виконання і тримати під контролем навіть великі обсяги роботи. Усі ці деталі, об'єднані в єдину структуру, знімають частину ментального навантаження й звільняють простір для того, щоб зосередитися на виконанні справ, а не на постійному згадуванні, що і коли потрібно зробити [4], [5].

Після фіксації термінів виконання у цифровому переліку завдань варто окремо звернути увагу на ті дії, які мають властивість повторюватися. Такі справи, як командні наради кожного тижня, регулярні розмови з клієнтами чи заплановані клуби за інтересами, зручніше одразу закласти у розклад. Інструменти для роботи над проєктами дозволяють автоматично розміщувати ці постійні події у календарі, тож зайвий раз переносити їх вручну вже не доведеться. Окрім того, варто озирнутися на ці регулярні пункти й перевірити, чи немає серед них таких, що втратили сенс або могли б бути передані іншому [4], [5].

Коли шаблон майбутнього дня наповнюється завданнями, постає завдання розташувати їх так, щоб структура виглядала логічною й реалістичною. Одні справи потребують виконання на початку дня через свою значущість чи строки, інші можна пересунути на кінець або навіть залишити на потім. Організованість значно зростає, якщо розставити пункти за часом виконання чи пріоритетністю, залишаючи місце для повторюваних подій у правильному місці. Під час заповнення календаря доцільно стежити за тим, аби не перевантажити день завданнями настільки, що не залишиться ані хвилини на перепочинок чи раптові коригування. Для цього зручно

періодично оцінювати щільність свого графіка у візуальному режимі й прибирати зайве [4], [5].

Коли основна структура розписана, залишається пам'ятати, що будь-який план — лише напрямок, а не закріплена схема. Справжня перевага сучасних цифрових рішень у тому, що вони дозволяють налаштовувати розклад під обставини, які постійно змінюються. Перенесення розмов, зсуви у пріоритетах, неочікувані доручення від клієнтів — усе це не стане проблемою, якщо можна у швидко відкоригувати час або замінити одну подію іншою. Такий підхід не тільки полегшує щоденне планування, а й дозволяє залишатися зібраним попри будь-які зміни, які приносить день [4], [5].

Щоб тримати під контролем власний день і уникати хаосу, варто мати чітко продуманий план дій, який задає ритм роботі й відпочинку. Грамотно складений щоденний розклад здатний не тільки навести лад у справах, а й звільнити голову від дрібниць, що розсіюють увагу. Для цього зручно користуватися заздалегідь підготовленим макетом, який служить своєрідною основою для організації завдань і зустрічей [5], [6].

Спершу доцільно визначитися, яка саме структура підходить під індивідуальні потреби — комусь достатньо простого списку справ у телефоні, іншим зручніше мати розбитий за годинами план на тиждень або роздруковану таблицю з місцем для нотаток і поміток. Студенту може стати у пригоді варіант із секціями для занять і навчання, тоді як фахівець обере погодинний план для координації ділових дзвінків, робочих зустрічей і рутинних обов'язків. Коли каркас розкладу готовий, він потребує тонкого налаштування. Деякі секції доведеться видалити або, навпаки, додати нові — усе залежить від того, як побудований типовий день. Для когось доречним буде мінімалістичний вигляд без зайвих деталей, а комусь потрібні інтеграції з іншими сервісами й нагадування про важливі дати [5], [6].

Далі слід визначити справи, що мають першочергове значення, і закріпити за ними відповідні часові проміжки. Найважливіші завдання бажано розміщувати у ті години, коли працездатність на максимумі — так досягається

кращий результат і витрачається менше зусиль. Якщо робота передбачає інтенсивні блоки, варто пам'ятати і про паузи: короткі відволікання від екрана чи свіже повітря повертають сили й допомагають тримати темп [5], [6].

Час від часу розклад має переглядатися й налаштовуватися під нові обставини. Немає потреби дотримуватися його сліпо — життя часто вносить корективи, і гнучкість у плануванні лише додає впевненості. Краще наприкінці дня чи тижня оцінити, як минув час: що вийшло виконати вчасно, а де варто збільшити чи зменшити відведений проміжок [5], [6]. Наприкінці дня корисно зупинитися й оцінити, що вдалося, а що можна змінити наступного разу. Якщо якісь завдання забрали більше сил, ніж очікувалося, це привід внести правки у тривалість подібних справ у майбутньому. Такий підхід перетворює щоденний шаблон із простої таблиці на гнучкий інструмент, що налаштовується під ритм людини й допомагає не втрачати пульс подій. Головна мета — щоб графік працював на людину, а не навпаки [5], [6].

Існують різні методи, які дозволяють вибудувати зручну структуру щоденних справ. Одні надають перевагу точному розподілу часу протягом дня, інші зосереджуються насамперед на важливості завдань або дотриманні дедлайнів. Погодинний підхід залишається одним із найпоширеніших, адже дає змогу чітко знати, коли і чим займатися, включаючи й особисті питання, які виникають поза робочим контекстом [5], [6]. Інший спосіб полягає в тому, щоб концентрувати увагу не на годинах, а на черговості справ за ступенем значущості. Такий формат виглядає як перелік ключових пунктів без жорсткої прив'язки до часу. Проте, щодня цей перелік потребує оновлення: зміни в зустрічах чи нові завдання постійно впливають на черговість і потребують переоцінки. Ще один підхід — орієнтація на строки. Коли всі завдання згруповані за датами виконання, виникає чітке розуміння, які елементи не можна відкладати, а що може зачекати. Однак така система, як і попередня, вимагає постійного перегляду та доповнення [5], [6].

Серед великої кількості шаблонів іноді непросто зупинитися на одному варіанті. Щоб підібрати найзручніший, корисно зіставити власні щоденні

звички з можливостями того чи іншого планувальника. Якщо робочий день насичений деталями, а завдань багато, вибір правильного формату допоможе не витратити час даремно й не тримати все в голові [5], [6].

Програмні рішення, своєю чергою, відкривають значно ширші можливості — від швидкого редагування до автоматичної синхронізації на всіх пристроях. Для багатьох це зручно, адже можна працювати з розкладом, де б людина не була. До того ж, такі інструменти легко об'єднуються з календарями, нагадуваннями та поштою, що особливо корисно тим, хто щодня управляє кількома напрямками роботи чи проєктами [6], [7].

Після вибору формату настає момент визначитися з програмами чи сервісами, які найкраще доповнять щоденне планування. Серед варіантів — від простих списків справ до комплексних рішень із цілою системою контролю завдань, які можна обговорювати разом із колегами. У такому середовищі завжди є змога розставити пріоритети, поділитися планами з командою чи швидко внести зміни [6], [7]. Зрештою, головне не форма, а здатність обраного програмного інструмента підтримувати чіткий ритм дня та допомагати не втрачати фокус на головному [6], [7].

Робота над щоденними справами часто передбачає застосування різноманітних програмних інструментів, що дають змогу не просто фіксувати заплановані дії, а й вибудовувати їх у зручній послідовності та контролювати виконання. Для багатьох достатньо простого переліку справ, проте така форма організації швидко втрачає ефективність, коли з'являється потреба переносити задачі, змінювати їх важливість чи розподіляти обов'язки між кількома людьми. Тому сьогодні дедалі більше користувачів звертається до цифрових програмних сервісів, які дають змогу користувачам не лише мати список справ під рукою, а й синхронізувати його з іншими даними та налаштувати спільний доступ [6], [7].

Онлайн рішення дозволяють легко додавати або редагувати пункти плану з будь-якого пристрою, що робить роботу більш гнучкою. Для візуалізації ходу виконання дедалі частіше використовуються спеціальні

дошки, де задачі переміщуються між станами від початку до завершення. Такий підхід полегшує виявлення етапів, де накопичуються затримки, і допомагає не втрачати контроль над термінами. Сучасні програми для планування часто поєднують різні підходи: класичні списки, інтуїтивні дошки та додаткові інструменти для коментування, прикріплення файлів або обговорення питань у команді. Завдяки цьому керування навіть нескладними щоденними справами перетворюється на впорядкований процес, де можна визначати пріоритети, бачити залежності між діями та швидко коригувати заплановане. Застосування таких рішень робить виконання завдань більш прогнозованим і дозволяє уникати хаотичних ситуацій, коли важливі деталі випадають з поля зору. Усе це спрямоване на те, щоб підвищити продуктивність і підтримувати баланс між справами, не витрачаючи зайвих зусиль на рутинний контроль [4], [5].

Застосування спеціального програмного середовища для контролю над виконанням робочих етапів відкриває чіткий і зручний спосіб упорядкувати навіть найдрібніші справи. Робота над проектами потребує не лише переліку дій, а й розуміння, хто, коли і за що відповідає. Середовище управління процесами дає можливість імпортувати готові масиви завдань або формувати їх прямо у проєкті, доповнюючи описами, поясненнями чи додатковими файлами. Важливо, що кожен етап можна налаштувати під конкретну команду: комусь потрібна лінійна структура, інші ж ефективніше працюють з гнучкими дошками, де видно увесь рух завдання від задуму до завершення. Таке середовище передбачає просте розподілення ролей: достатньо додати відповідального, щоб уникнути хаосу й непорозумінь. Візуальні подання спрощують моніторинг, дозволяючи швидко побачити, де накопичуються затримки або що потребує особливої уваги. Для зручності передбачаються різні фільтри та позначки, завдяки яким можна сортувати справи за станом чи важливістю. Робочий процес збагачується інструментами спільної взаємодії: обговорення всередині завдань, повідомлення та нагадування допомагають не втрачати важливі деталі й тримати команду в курсі оновлень. Якщо потрібен

огляд поточного стану справ, інтегровані панелі моніторингу збирають ключові показники й надають їх у зручній формі, що дозволяє швидко оцінити темпи просування. Завдяки такому підходу щоденне керування стає послідовним і більш передбачуваним, а кожен учасник бачить, як його внесок впливає на спільний результат [4], [5].

Організація спільної роботи над будь-яким проектом завжди потребує чіткого розподілу обов'язків і регулярного контролю за виконанням завдань кожним учасником. Особливо важливо забезпечити таку прозорість, коли справи передаються іншим членам команди. Навіть якщо певне доручення було озвучено під час робочої зустрічі, варто закріпити його у цифровій системі та призначити конкретному виконавцю — це дає гарантію, що справа не загубиться серед інших справ і буде виконана вчасно [5], [6].

Завдяки сучасним програмним рішенням можна не лише фіксувати завдання, а й миттєво відслідковувати, як просувається робота, хто над чим працює та чи не перевантажений хтось із колег. Якщо у когось спостерігається надмірне навантаження, достатньо кількох натискань миші, аби перерозподілити обсяг роботи, не порушуючи загального плану. Такий підхід значно економить час і допомагає уникати дублювання зусиль [5], [6].

Щоденне звернення до власного простору завдань дозволяє чітко бачити, що заплановано на день, а які пункти ще очікують своєї черги. За потреби можна швидко оновити пріоритети, змінити послідовність дій чи додати додаткові матеріали — коментарі, файли або нотатки. Коли якась справа затримується, система нагадає про це, виділяючи прострочені пункти та спрощуючи прийняття рішень: чи перенести термін, чи пришвидшити виконання [4], [5]. Наявність вбудованих індикаторів прогресу у подібних програмах показує, на якому етапі перебуває кожна справа. Учасники команди самостійно оновлюють статуси, що дозволяє актуалізувати дані в системі й одразу бачити реальний стан справ. Щоб уникати плутанини, найкраще після завершення кожного дня приділяти кілька хвилин на звірку виконаного і фіксацію того, що ще залишається відкритим [6], [7].

Також у програмному забезпеченні для планування та організації щоденних справ надзвичайно важливо забезпечити спільний доступ до основних списків завдань для всієї команди — так кожен завжди розуміє, хто й за що відповідає. Водночас особисті дрібниці чи приватні нотатки, які не стосуються загального проєкту, варто зберігати окремо — це дозволяє тримати робочий простір впорядкованим і зрозумілим для всіх [6], [7].

Ефективне використання часу нерідко залежить від того, наскільки добре організовані поточні справи та наскільки злагоджено працює команда. Щоб уникнути цього, доцільно використовувати сучасні програмні засоби, які допомагають зводити до купи всі плани та задачі й водночас не витратити зайві години на постійне ручне упорядкування нотаток. Програмне забезпечення для організації проєктів дозволяє поєднувати календарі, щоденники й функції пріоритезації так, щоб було зрозуміло, що варто робити в першу чергу, а що можна перенести. Всі завдання видно на одному екрані, їх можна позначати як нагальні або менш важливі, щоб фокус залишався там, де це потрібно найбільше. Коли кожен у колективі бачить, що саме потрібно зробити й до якого часу, зменшується ймовірність, що щось загубиться. Учасники команди беруть на себе відповідальність за частину справ, а керівник має змогу відстежувати, наскільки рівномірно розподілене навантаження. Частину роботи завжди можна передати комусь іншому, якщо виникає перевантаження [6], [7].

Визначено, що програмне забезпечення для планування та організації щоденних справ використовується для керування особистим та робочим розпорядком, допомагаючи користувачеві структурувати завдання, контролювати хід їх виконання та своєчасно реагувати на важливі події. Такі програмні застосунки зазвичай поєднують у собі можливості керування списками справ, інтеграцію календаря, а також засоби нагадувань і сповіщень, завдяки яким можна уникнути пропущених граничних строків виконання задач чи незавершених пунктів плану.

1.2 Аналіз програмного забезпечення для планування та організації щоденних справ

Проаналізуємо програмне забезпечення для планування та організації щоденних справ [8]-[16].

Програмне забезпечення Twobird (рис. 1.1) пропонує незвичайний підхід до організації щоденних справ, перетворюючи звичайну електронну скриньку на повноцінний простір для управління завданнями та листуванням одночасно. Його робота ґрунтується на тісному зв'язку з сервісами електронної пошти, серед яких підтримуються популярні платформи від Google і Microsoft, що дозволяє користувачеві залишатись у звичному середовищі, не відриваючись від щоденних листів [11], [14].

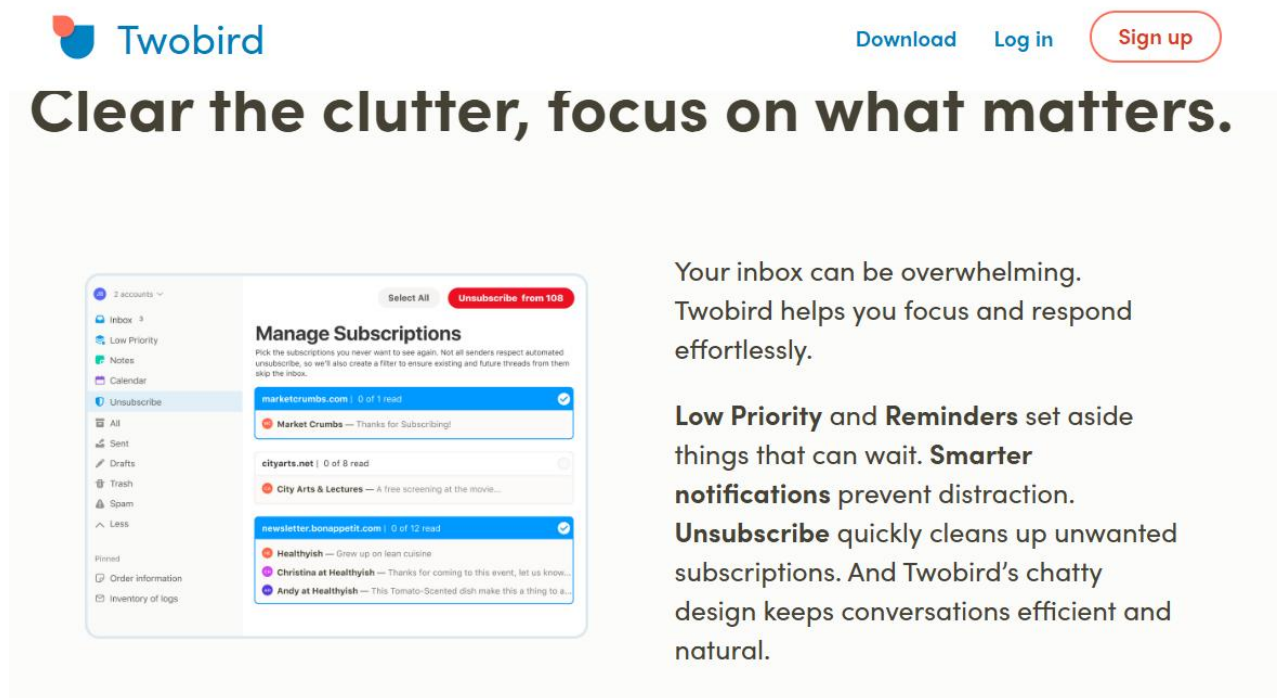


Рисунок 1.1 – Програмне забезпечення Twobird [14]

Однією з характерних рис цього рішення є вміння відсортовувати потік повідомлень, автоматично визначаючи, які з них варті першочергової уваги, а які можна залишити на потім. Система самостійно пропонує зручні нагадування, завдяки яким важливі повідомлення не губляться серед інших і

зберігають свою актуальність у потрібний момент. Додатково користувач звільняється від надокучливих сповіщень, оскільки Twobird уміє інтелектуально фільтрувати повідомлення, зменшуючи кількість зайвих відволікань і допомагаючи зосередитися на ключових діях [11], [14].

Крім опрацювання звичайної кореспонденції, платформа дозволяє без зайвих кроків відмовлятися від підписок, які більше не потрібні, що суттєво очищує вхідні повідомлення та спрощує роботу з ними. Також серед функціональних можливостей — можливість залишати персональні нотатки безпосередньо у межах листів, що робить кожен ланцюжок обговорень більш організованим і структурованим [11], [14].

Доступ до Twobird відкрито на різних пристроях, як на настільних операційних системах, так і на мобільних, що дозволяє підтримувати безперервну роботу незалежно від місця перебування користувача. Такий підхід дає змогу перетворити повсякденну поштову рутину на гнучкий і зручний інструмент планування та контролю завдань, де вся необхідна інформація залишається під рукою у будь-який момент [11], [14].

Основними характеристиками програмного забезпечення Twobird є такі [11], [14]:

- інтеграція з поштою: програмного забезпечення Twobird забезпечує повноцінну роботу з сервісами Google та Microsoft без потреби додаткових налаштувань [11], [14];

- розумна фільтрація листів: програмного забезпечення Twobird автоматично визначає важливість вхідних повідомлень та групує їх за пріоритетністю [11], [14];

- нагадування про важливе: програмного забезпечення Twobird дозволяє встановлювати відкладені сповіщення для листів, щоб не втратити важливі завдання [11], [14];

- захист від спаму: програмного забезпечення Twobird спрощує процедуру відмови від непотрібних розсилок без зайвих дій [11], [14];

- нотатки в листах: програмного забезпечення Twobird дозволяє створювати текстові коментарі безпосередньо у тілі повідомлення для кращої організації інформації [11], [14];

- багатоплатформеність: програмного забезпечення Twobird підтримує роботу на Windows, Mac, iOS і Android, що робить планування доступним з будь-якого пристрою [11], [14];

- сповіщення без шуму: програмного забезпечення Twobird надсилає лише справді важливі повідомлення, знижуючи кількість відволікань під час роботи [11], [14].

Серед недоліків програмного забезпечення Twobird можна відзначити обмежені можливості для складного проєктного планування, що робить його менш придатним для роботи з великими командами або багаторівневими завданнями. Іноді користувачі стикаються з нестачею гнучких інструментів для налаштування робочих процесів під власні потреби, а також відсутністю розширеної інтеграції з деякими зовнішніми сервісами, що могли б доповнити функціонал. Окрім того, при великій кількості листів і нотаток система інколи потребує більше часу для обробки даних, що може впливати на швидкість відгуку застосунку [11], [14].

Програмне забезпечення Devonthink (рис. 1.2) представляє собою універсальний простір, де можна не лише складати списки справ, а й зручно зберігати всі супровідні матеріали, які часто супроводжують повсякденне планування. Його основна ідея полягає у створенні впорядкованого сховища, де поруч зі звичайними нагадуваннями можна тримати нотатки, відскановані документи, вирізки з вебсторінок чи важливі файли, об'єднуючи це в одну гнучку інформаційну систему. Такий підхід особливо корисний тим, хто прагне зберігати порядок у великому обсязі даних і мати можливість швидко знаходити потрібне без плутанини серед різних папок чи сервісів [10], [15].

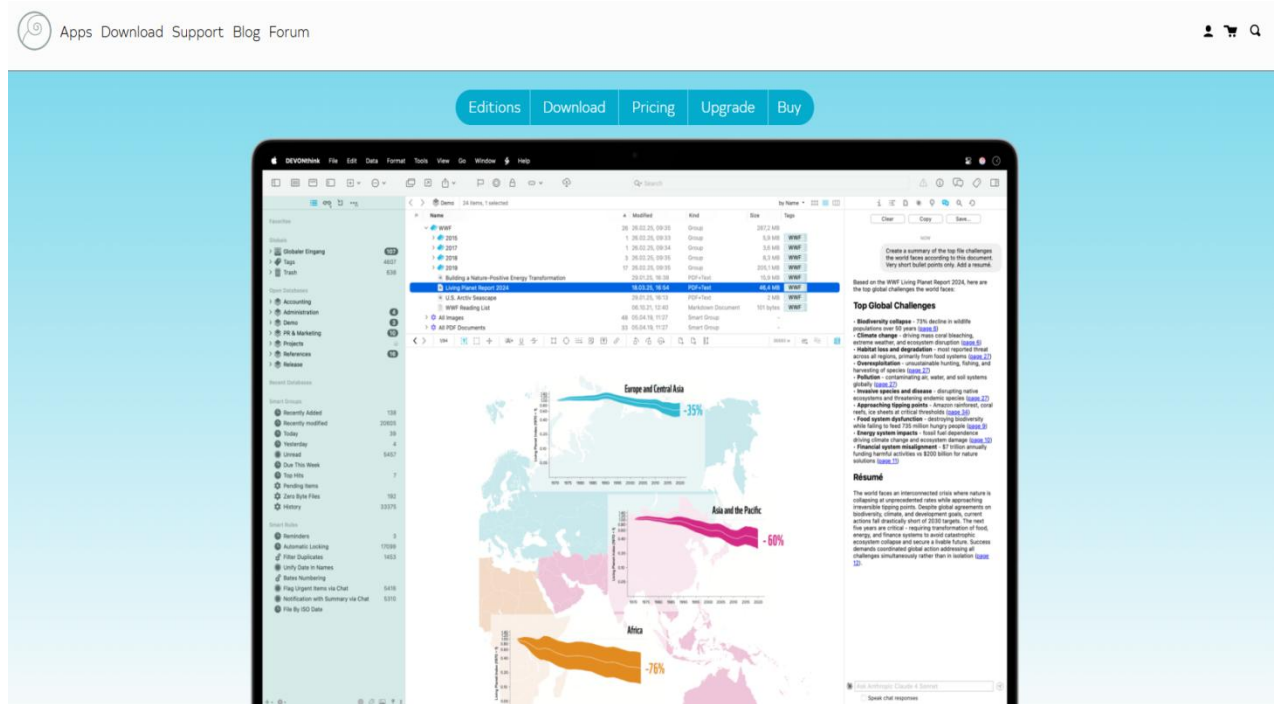


Рисунок 1.2 – Програмне забезпечення Devonthink [15]

Програмне забезпечення Devonthink зосереджується більше на організації даних і текстових матеріалів, ніж на класичному управлінні задачами у стилі розширених менеджерів проєктів. Через це користувач може відчувати нестачу складних сценаріїв відстеження термінів, розподілу обов'язків чи детальної декомпозиції завдань, які зазвичай потребують спеціалізованих рішень для командної роботи. Тим не менш Devonthink залишається сильним помічником для тих, хто хоче поєднати планування, нотатки та архівування у єдиному місці й мати глибокий контроль над власною інформацією, спираючись на потужну систему пошуку й класифікації контенту. Така комбінація робить цей продукт практичним вибором для особистого використання або індивідуальної організації знань і справ, де важлива не стільки структура проєкту, скільки зручність доступу до великого інформаційного масиву [10], [15].

Програмне забезпечення Devonthink має такі особливості [10], [15]:

– централізоване сховище даних: програмне забезпечення Devonthink дозволяє зберігати та впорядковувати документи, нотатки й списки справ в одному місці [10], [15];

- розширений пошук: програмне забезпечення Devonthink забезпечує швидкий доступ до потрібних файлів і фрагментів інформації завдяки потужним алгоритмам індексації [10], [15];
- інтеграція з різними форматами: програмне забезпечення Devonthink підтримує збереження текстових файлів, зображень, PDF та вебзаписів для повного контролю над усіма матеріалами [10], [15];
- автоматизація впорядкування: програмне забезпечення Devonthink застосовує інтелектуальні правила для класифікації й сортування документів без ручного втручання [10], [15];
- синхронізація між пристроями: програмне забезпечення Devonthink дозволяє працювати з одними й тими самими даними на кількох пристроях, зберігаючи актуальність інформації [10], [15];
- фокус на індивідуальній роботі: програмне забезпечення Devonthink більше орієнтовано на особисте використання, ніж на командне управління проєктами [10], [15];
- обмежений менеджер завдань: програмне забезпечення Devonthink надає базові інструменти для списків справ, але не замінює спеціалізовані рішення для контролю складних проєктів [10], [15].

Серед недоліків програмного забезпечення Devonthink можна відзначити обмежені можливості для детального планування складних проєктів, що може стати перешкодою для тих, хто шукає інструмент із глибоким управлінням завданнями та спільною роботою в команді. Іноді користувачі стикаються з необхідністю витрачати час на налаштування структури даних, оскільки велика кількість функцій і гнучкість організації можуть виявитися занадто громіздкими для швидкого старту. Також певним недоліком програмного забезпечення є залежність від екосистеми використовуваних пристроїв Apple, що звужує коло потенційних користувачів, які працюють на різних операційних системах і хочуть мати повноцінний доступ до своїх архівів з будь-якого середовища [10], [15].

Програмне забезпечення Sunsama (рис. 1.3) слугує зручним інструментом для тих, хто прагне організувати власний день і навчитися розумно розподіляти час без складних налаштувань і зайвих деталей. Цей застосунок приваблює перш за все своєю простотою та продуманим інтерфейсом, що дає змогу швидко розібратися навіть тим, хто вперше користується цифровими планувальниками. Основна ідея полягає в тому, щоб користувач міг зосередитися на ключових завданнях, розбити їх на підзадачі, зазначити конкретні терміни виконання й оцінити необхідний час, не перевантажуючи свій розклад другорядними деталями [12], [16].

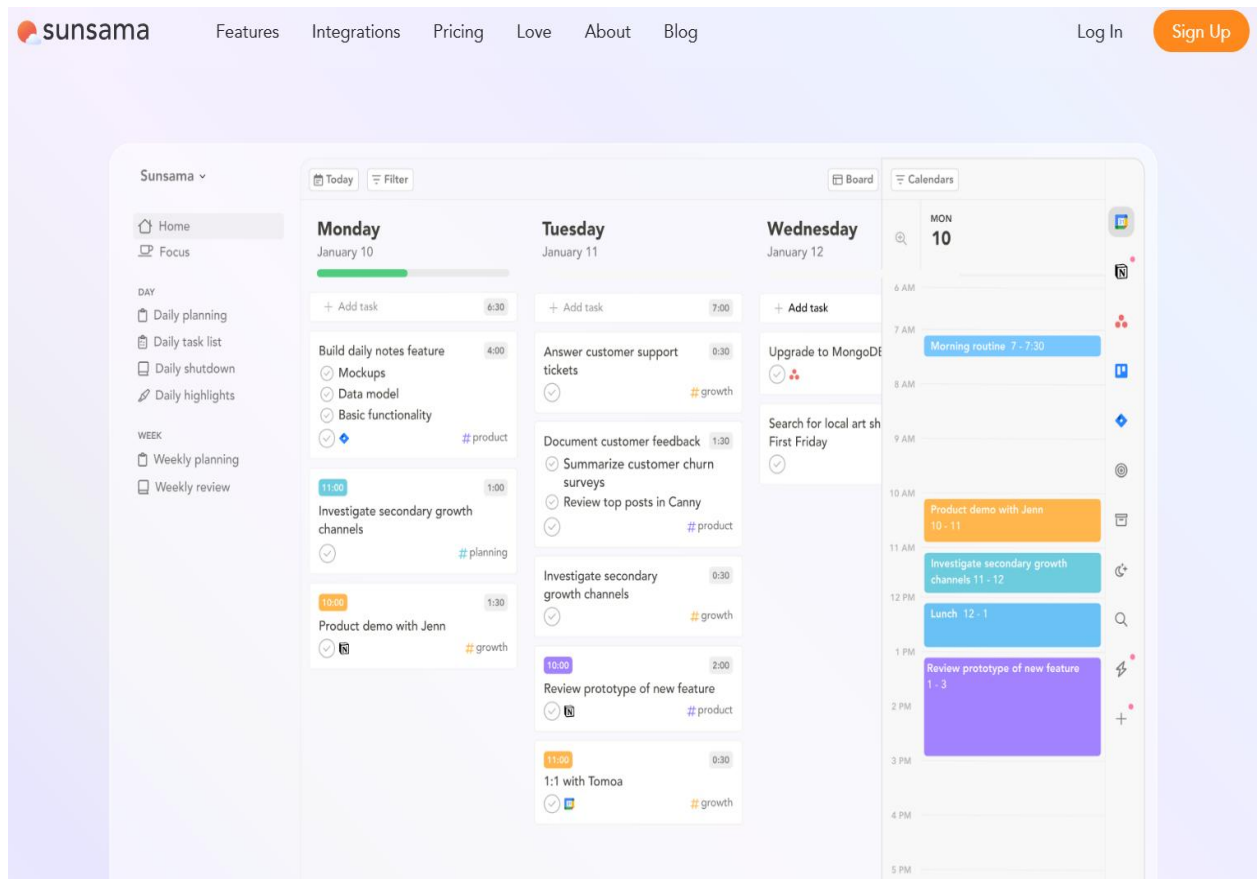


Рисунок 1.3 – Програмне забезпечення Sunsama [16]

Програмне забезпечення Завдяки тому, що Sunsama поєднує календар і щоденник у єдиному вікні, планування відбувається у зрозумілому форматі, де завдання можна миттєво співвіднести з наявним розкладом. Вбудовані підказки й можливість аналізувати власний прогрес допомагають переглядати

щоденні досягнення й обдумувати план на наступний день. Такий підхід сприяє формуванню корисних звичок планування й дозволяє навчитися реалістично оцінювати власні можливості [12], [16].

Разом із цим Sunsama не намагається охопити всі сценарії використання. Його мінімалістичний підхід робить роботу комфортною для особистого планування, однак при роботі з великими проєктами чи командною взаємодією може відчуватися нестача гнучких інструментів для комплексного управління робочими процесами. Крім того, відсутність повноцінної безкоштовної версії обмежує коло тих, хто хотів би оцінити всі можливості застосунку без фінансових витрат. Попри ці нюанси, Sunsama залишається одним із найзрозуміліших і найприємніших у користуванні варіантів для тих, хто прагне навчитися планувати справи усвідомлено й тримати особисту продуктивність під контролем [12], [16].

Програмне забезпечення Sunsama має такі особливості [12], [16]:

- простий інтерфейс: програмне забезпечення Sunsama забезпечує легкий старт для новачків без потреби у складному налаштуванні [12], [16];
- поєднання календаря й задач: програмне забезпечення Sunsama дозволяє одночасно бачити список справ і розклад користувача у зручному вигляді [12], [16];
- підтримка підзадач: програмне забезпечення Sunsama дає змогу деталізувати великі завдання на менші етапи для кращого контролю виконання [12], [16];
- інструменти саморефлексії: програмне забезпечення Sunsama мотивує користувача переглядати результати дня та планувати наступні дії більш обдуманно [12], [16];
- відсутність безкоштовного тарифу: використання можливе лише за умови оформлення платного доступу [12], [16];
- обмежений функціонал: програмне забезпечення Sunsama не пропонує розширених опцій для роботи зі складними багаторівневими проєктами [12], [16].

Серед недоліків програмного забезпечення Sunsama можна відзначити відсутність можливості працювати безкоштовно, що змушує користувачів одразу замислюватися про додаткові витрати навіть на початковому етапі знайомства з сервісом. Крім того, спрощений набір функцій інколи не дозволяє гнучко налаштовувати завдання чи керувати складними робочими процесами, що робить цей інструмент менш придатним для командних проєктів або багаторівневого планування. Деякі користувачі можуть також зіткнутися з тим, що вбудовані можливості інтеграції обмежені і не охоплюють усі популярні сервіси, які часто використовуються у щоденній роботі [12], [16].

Порівняльну характеристику програмного забезпечення для планування та організації щоденних справ наведено у таблиці 1.1.

Таблиця 1.1 – Порівняння програмного забезпечення для планування та організації щоденних справ

Критерій порівняння	Twobird [14]	Devonthink [15]	Sunsama [16]
Зручність планування завдань	+	+-	+
Інтеграція з поштою	+	-	+-
підтримка командної роботи	+-	+	+-
Наявність безкоштовної версії	-	+-	-
Простота інтерфейсу	+-	+-	-
Засоби збереження та редагування документів	+-	+	+-

За результатами проведеного аналізу можна зробити висновок, що у наш час існує досить багато програмних засобів для планування та організації щоденних справ. Проте деякі програмні засоби для планування та організації щоденних справ вимагають додаткового часу на налаштування та регулярне

оновлення даних, що може стати рутинним і навіть обтяжливим для користувача. Деякі програми надто складні у використанні або перенасичені функціями, які не завжди потрібні, що іноді призводить до перевантаження інтерфейсу і втрати мотивації їх використовувати. Окремою проблемою є ризики, пов'язані з конфіденційністю особистої інформації, оскільки дані про зустрічі, контакти чи завдання часто зберігаються у хмарних сервісах і можуть бути вразливими до несанкціонованого доступу. Крім того, надмірне використання цифрових програмних планувальників може призвести до втрати навичок самоорганізації без допоміжних інструментів і сформувати залежність від автоматизованих нагадувань. Тому актуальною є розробка програмного забезпечення для планування та організації щоденних справ.

При розробці програмного забезпечення для планування та організації щоденних справ необхідно забезпечити такі функціональні вимоги:

- підтримка можливості подання інформації для планування та організації щоденних справ користувача;
- підтримка можливості авторизації та реєстрації користувачів;
- можливість відображення календаря зустрічей;
- можливість відображення інформації про діяльність та справи користувача;
- підтримка можливості зберігання інформації про контакти користувача;
- підтримка можливості редагування налаштувань.

1.3 Висновки за розділом 1

Визначено, що програмне забезпечення для планування та організації щоденних справ використовується для керування особистим та робочим розпорядком, допомагаючи користувачеві структурувати завдання, контролювати хід їх виконання та своєчасно реагувати на важливі події. Такі програмні застосунки зазвичай поєднують у собі можливості керування

списками справ, інтеграцію календаря, а також засоби нагадувань і сповіщень, завдяки яким можна уникнути пропущених граничних строків виконання задач чи незавершених пунктів плану.

За результатами проведеного аналізу зроблено висновок, що у наш час існує досить багато програмних засобів для планування та організації щоденних справ. Проте деякі програмні засоби для планування та організації щоденних справ вимагають додаткового часу на налаштування та регулярне оновлення даних, що може стати рутинним і навіть обтяжливим для користувача. Деякі програми надто складні у використанні або перенасичені функціями, які не завжди потрібні, що іноді призводить до перевантаження інтерфейсу і втрати мотивації їх використовувати. Окремою проблемою є ризики, пов'язані з конфіденційністю особистої інформації, оскільки дані про зустрічі, контакти чи завдання часто зберігаються у хмарних сервісах і можуть бути вразливими до несанкціонованого доступу. Крім того, надмірне використання цифрових програмних планувальників може призвести до втрати навичок самоорганізації без допоміжних інструментів і сформувати залежність від автоматизованих нагадувань. Тому актуальною є розробка програмного забезпечення для планування та організації щоденних справ.

Сформульовано функціональні вимоги до програмного забезпечення для планування та організації щоденних справ.

2 МАТЕРІАЛИ І МЕТОДИ

2.1 Вибір мови програмування

При розробці програмного забезпечення для планування та організації щоденних справ було використано мову програмування C# [17], [18].

C# вважається однією з тих мов, що поєднують легкість освоєння з широким спектром інструментів для реалізації найрізноманітніших програмних проєктів — від компактних утиліт до масштабних корпоративних рішень. Завдяки сильній інтеграції з платформою .NET відкриваються широкі можливості для створення настільних програм і вебзастосунків, а підтримка багатьох фреймворків спрощує впровадження різноманітних функцій. Мова забезпечує стабільну взаємодію зі складними наборами даних і дає змогу працювати з багатьма процесами одночасно, що особливо важливо для ресурсоємних завдань. З роками C# удосконалюється, доповнюючись новими функціональними можливостями, що дозволяє йти в ногу з технічним розвитком галузі. Питання безпеки коду стоїть у пріоритеті, завдяки чому мінімізуються помилки під час експлуатації програмних продуктів. Асинхронне виконання задач у цій мові спрощує створення застосунків, здатних швидко реагувати на запити користувача. Саме ці особливості підтримують високий попит на C# серед професіоналів, які працюють у різних напрямках сучасної розробки [17], [18].

Мова програмування C# добре пристосована для написання різних програм завдяки широкій підтримці у середовищі .NET і можливості працювати з різними фреймворками. Серед переваг можна відзначити стабільну роботу зі складними структурами даних і багатопотоковість, що дозволяє ефективно реалізовувати обробку численних паралельних завдань. C# постійно розвивається, отримуючи нові можливості та розширення, що дає змогу адаптувати код до сучасних вимог ринку. Особливу увагу приділено безпеці виконання програмного коду, що мінімізує ризики помилок під час роботи застосунків [17], [18].

Підтримка сучасних концепцій, таких як асинхронність, дозволяє створювати більш чутливі та швидкодійні сервіси. Завдяки цьому мова залишається затребуваною серед професійних розробників і активно використовується у багатьох сферах програмної інженерії [17], [18].

Мова програмування C# часто використовується для проєктування програм, які мають організовувати повсякденні завдання, оскільки ця мова пропонує технічні засоби, що дозволяють легко реалізовувати навіть складні функціональні вимоги. Об'єктна структура у поєднанні з багатим арсеналом готових бібліотек значно полегшує підключення зовнішніх сервісів і побудову логіки, що потребує високої надійності. Завдяки тісному зв'язку з екосистемою .NET стає можливим запуск одного й того самого застосунку на різних операційних системах без суттєвих доробок [17], [18].

Крім того, C# відзначається активною спільнотою розробників і великою кількістю навчальних матеріалів, що спрощує підтримку й розвиток продукту у перспективі. Використання цієї мови також забезпечує високий рівень безпеки та стабільності, що особливо важливо для програм, де зберігаються особисті дані користувачів [17], [18].

Наявність зручних інструментів для візуальної частини і роботи з даними допомагає швидко створювати засоби планування, що мають функції нагадування, календаря чи синхронізації з іншими пристроями [17], [18].

Додатковим стимулом на користь цієї мови є те, що її активно підтримують численні фахівці, які постійно діляться досвідом і розробляють нові рішення, що робить супровід і вдосконалення застосунків більш передбачуваними. Окремо варто відзначити надійність і захищеність даних, що особливо цінно, коли йдеться про персональну інформацію користувачів [17], [18].

Обґрунтування вибору мови програмування для реалізації програмного забезпечення для планування та організації щоденних справ наведено у таблиці 2.1.

Таблиця 2.1 – Обґрунтування вибору мови програмування для реалізації ПЗ для планування та організації щоденних справ

Критерій порівняння мов програмування	Мова програмування		
	C#	Python	Java
Розвиненість інструментів розробки	+	+-	+
Можливості для кросплатформенності	+	+-	+
Зручність створення графічного інтерфейсу	+	+-	+-
Зручність для створення ПЗ для планування та організації щоденних справ	+	-	-
Продуктивність виконання коду	+	+-	+-

Отже, для реалізації програмного забезпечення для планування та організації щоденних справ обрано мову програмування C#, яка поєднує об'єктно-орієнтований підхід із широким набором бібліотек, що спрощує реалізацію складної бізнес-логіки та інтеграцію різноманітних сервісів. Завдяки підтримці платформи .NET відкривається можливість легко забезпечувати кросплатформенність і розгортати застосунок не лише на Windows, а й на інших популярних операційних системах.

2.2 Вибір середовища розробки для створення програмного забезпечення для планування та організації щоденних справ

В якості середовища розробки для створення програмного забезпечення для планування та організації щоденних справ було обрано середовище Visual Studio [19], [20].

Середовище розробки Visual Studio поєднує інтуїтивність роботи з багатим інструментарієм для реалізації різних завдань програмної інженерії. Цей інструмент дозволяє розробникам ефективно організовувати процес створення програм незалежно від обсягу чи складності проєкту, оскільки гнучкість налаштувань і підтримка великої кількості мов програмування значно розширюють межі використання. Вбудовані механізми інтеграції з популярними системами контролю версій роблять співпрацю над спільними завданнями зрозумілою та чітко структурованою, що особливо важливо для команд, де одночасно працюють кілька людей над одними і тими ж модулями коду. Автодоповнення синтаксису й потужна система підказок дозволяють суттєво скоротити час на написання і перевірку коду, оскільки можливість уникнути технічних помилок на етапі розробки позитивно впливає на якість кінцевого продукту. Широкий вибір інструментів для тестування програм безпосередньо у середовищі допомагає переконатися у працездатності застосунку ще до його розгортання на реальних пристроях, а вбудовані емулятори і симулятори стають у пригоді тоді, коли фізичний доступ до певного обладнання тимчасово відсутній. Особливий акцент робиться на зручності в управлінні структурою проєкту, адже Visual Studio надає всі необхідні засоби для грамотного упорядкування вихідного коду, що дозволяє швидко орієнтуватися навіть у багаторівневих рішеннях. До того ж середовище містить безліч функцій для полегшення рефакторингу, моніторингу швидкодії та оптимізації ресурсів, що стає визначальним у випадках, коли йдеться про довготривале супроводження програмних продуктів або поступове розширення їх функціоналу. З такою гнучкістю й різноманітністю можливостей Visual Studio залишається одним із найбільш надійних виборів для фахівців, які цінують стабільність, високу якість інструментів і готовність до масштабування проєктів відповідно до зростаючих вимог [19], [20].

Застосування середовища розробки Visual Studio у процесі розробки програм, які мають на меті впорядковувати повсякденні обов'язки й

допомагати людям керувати своїм часом, обґрунтовується багатьма технічними перевагами цієї платформи. Насамперед це середовище тісно поєднане з мовою C#, що гарантує стабільну та зручну роботу з усім спектром функціоналу, характерного для цієї технології. Широкий асортимент готових інструментів, який пропонує середовище розробки Visual Studio, дозволяє створювати навіть складні багаторівневі застосунки без потреби у залученні великої кількості сторонніх рішень. Інтуїтивно зрозумілий інтерфейс робить процес налаштування проєкту прозорим, а численні можливості для відлагодження та перевірки програмних модулів забезпечують своєчасне виявлення помилок ще на етапі написання коду. Якщо у розробника з'являється потреба у впровадженні додаткових функцій, то різноманітні плагіни та доповнення допомагають швидко розширювати робоче середовище, не втрачаючи при цьому продуктивності. Значну роль відіграє і підтримка зручних інструментів для створення користувацьких інтерфейсів, обробки даних і роботи з мережевими ресурсами, адже планувальники часто вимагають постійного обміну інформацією між різними пристроями та синхронізації подій у реальному часі [19], [20].

Важливо, що середовище Visual Studio оновлюється регулярно, що гарантує доступ до найновіших рішень у сфері програмування та дозволяє адаптувати проєкт до змін у технологічному середовищі без значних витрат часу. Стабільність і висока якість роботи цієї платформи позитивно позначаються на всіх етапах життєвого циклу продукту — від первинного проєктування до подальшої підтримки й удосконалення, що робить її незамінною для розробників, які працюють над створенням ефективних інструментів організації щоденних справ [19], [20].

Обґрунтування вибору середовища розробки для створення програмного забезпечення для планування та організації щоденних справ наведено у таблиці 2.2.

Таблиця 2.2 – Обґрунтування вибору середовища розробки для створення програмного забезпечення для планування та організації щоденних справ

Критерій порівняння середовищ розробки	Середовища розробки		
	Visual Studio	Rider	Eclipse
Інтеграція з мовою C#	+	+-	+-
Наявність вбудованих засобів відлагодження	+	+-	+
Підтримка складної архітектури рішень	+	+-	+-
Зручність засобів для створення графічного інтерфейсу	+	-	+-
Підтримка кросплатформеності	+	+-	+-

Таким чином, для створення програмного забезпечення для планування та організації щоденних справ обрано середовище розробки Visual Studio через поєднання глибокої інтеграції з мовою C# та широкого набору інструментів для розробки. Крім того, це середовище пропонує інтуїтивний інтерфейс, який спрощує налаштування складних рішень і пришвидшує перевірку коду завдяки вбудованим засобам відлагодження та тестування. Велика кількість розширень і плагінів дозволяє адаптувати робочий простір під конкретні потреби проєкту та легко впроваджувати додаткові функції без зайвих витрат часу. Іншою перевагою є те, що Visual Studio полегшує роботу з інтерфейсами користувача, базами даних і мережевими сервісами, що особливо важливо для планувальників, де потрібна синхронізація подій і нагадувань.

2.3 Розробка моделі взаємодії користувача з програмним забезпеченням для планування та організації щоденних справ

З метою можливості розробки програмного забезпечення для планування та організації щоденних справ створимо відповідну модель у вигляді діаграми прецедентів.

При розробленні моделі взаємодії користувача з програмним забезпеченням для планування та організації щоденних справ враховуємо те, що користувачі програми мають можливості редагування інформації про заплановані зустрічі та завдання, можливості авторизації та реєстрації, можливості зберігання та редагування інформації про контакти користувача, можливості редагування налаштувань.

Основний успішний сценарій користувача при роботі з програмним забезпеченням для планування та організації щоденних справ можна описати таким чином:

- користувач програмного забезпечення для планування та організації щоденних справ авторизується;
- після авторизації користувача сервер програмного забезпечення для планування та організації щоденних справ у відповідь повертає основну сторінку вебсайту для планування та організації щоденних справ;
- користувач бачить відповідну сторінку програмного забезпечення для планування та організації щоденних справ та шукає потрібну зустріч для редагування або створює нову зустріч у потрібну дату;
- користувач за необхідності змінює інформацію про заплановану зустріч;
- користувач натискає відповідну кнопку для підтвердження рішення про зміну інформації про зустріч;
- користувач завершує роботу з програмним забезпеченням для планування та організації щоденних справ.

Враховуючи сценарії роботи користувачів програмного забезпечення для планування та організації щоденних справ при роботі з програмою, модель взаємодії користувача з програмним забезпеченням для планування та організації щоденних справ подамо у вигляді діаграми прецедентів, зображеної на рис. 2.1.

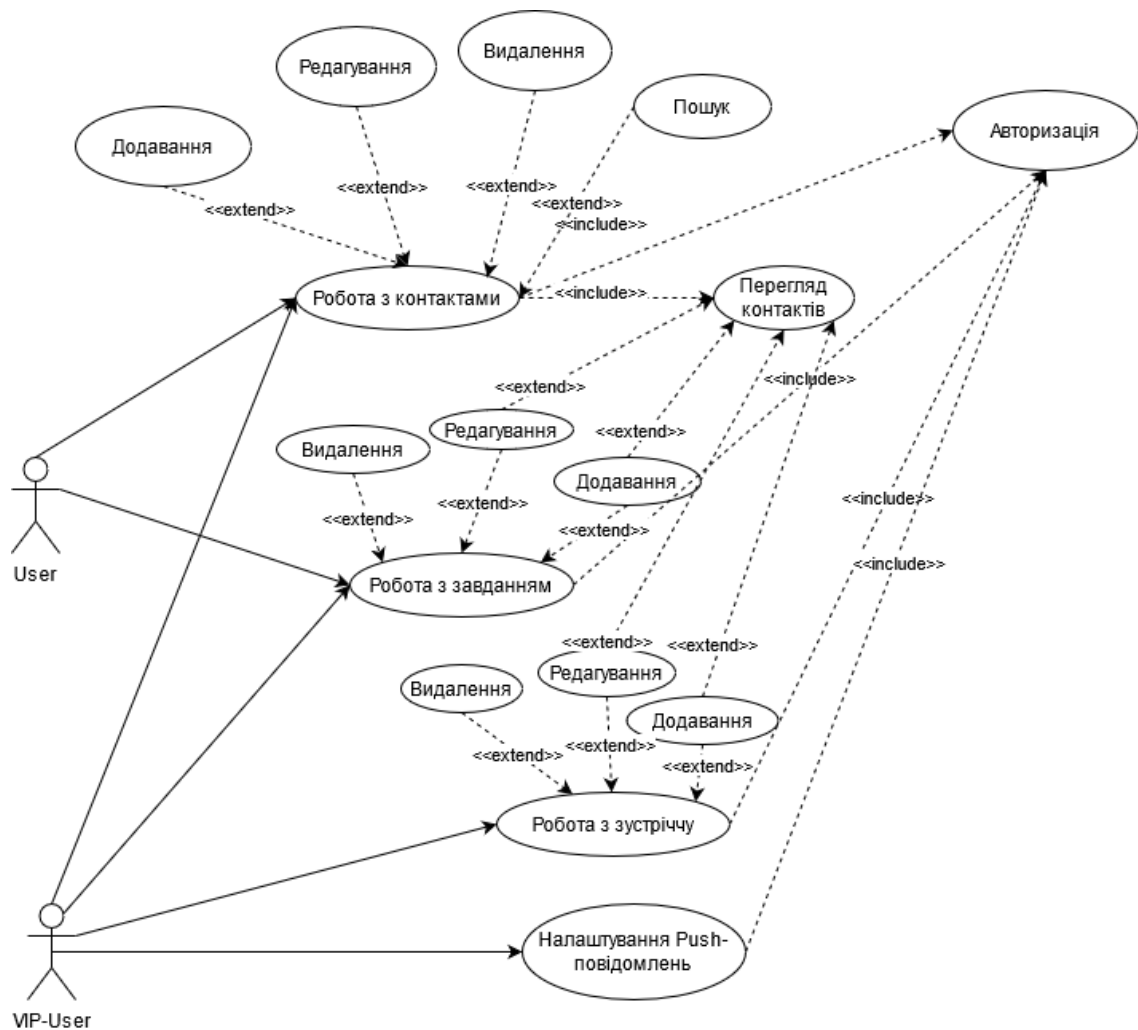


Рисунок 2.1 – Модель взаємодії користувача з програмним забезпеченням для планування та організації щоденних справ

Таким чином, створено модель взаємодії користувача з програмним забезпеченням для планування та організації щоденних справ, яка подана у вигляді діаграми прецедентів, та дозволяє розроблювати програмне забезпечення для планування та організації щоденних справ.

2.4 Висновки за розділом 2

Для реалізації програмного забезпечення для планування та організації щоденних справ обрано мову програмування C#, яка поєднує об'єктно-орієнтований підхід із широким набором бібліотек, що спрощує реалізацію складної бізнес-логіки та інтеграцію різноманітних сервісів. Завдяки підтримці платформи .NET відкривається можливість легко забезпечувати кросплатформенність і розгортати застосунок не лише на Windows, а й на інших популярних операційних системах.

Для створення програмного забезпечення для планування та організації щоденних справ обрано середовище розробки Visual Studio через поєднання глибокої інтеграції з мовою C# та широкого набору інструментів для розробки. Крім того, це середовище пропонує інтуїтивний інтерфейс, який спрощує налаштування складних рішень і пришвидшує перевірку коду завдяки вбудованим засобам відлагодження та тестування. Велика кількість розширень і плагінів дозволяє адаптувати робочий простір під конкретні потреби проєкту та легко впроваджувати додаткові функції без зайвих витрат часу.

Створено модель взаємодії користувача з програмним забезпеченням для планування та організації щоденних справ, яка подана у вигляді діаграми прецедентів, та дозволяє розроблювати програмне забезпечення для планування та організації щоденних справ.

3 ОПИС ПРОГРАМИ

3.1 Структура програмного забезпечення для планування та організації щоденних справ

Структуру програмного забезпечення для планування та організації щоденних справ у вигляді діаграми класів наведено на рис. 3.1.

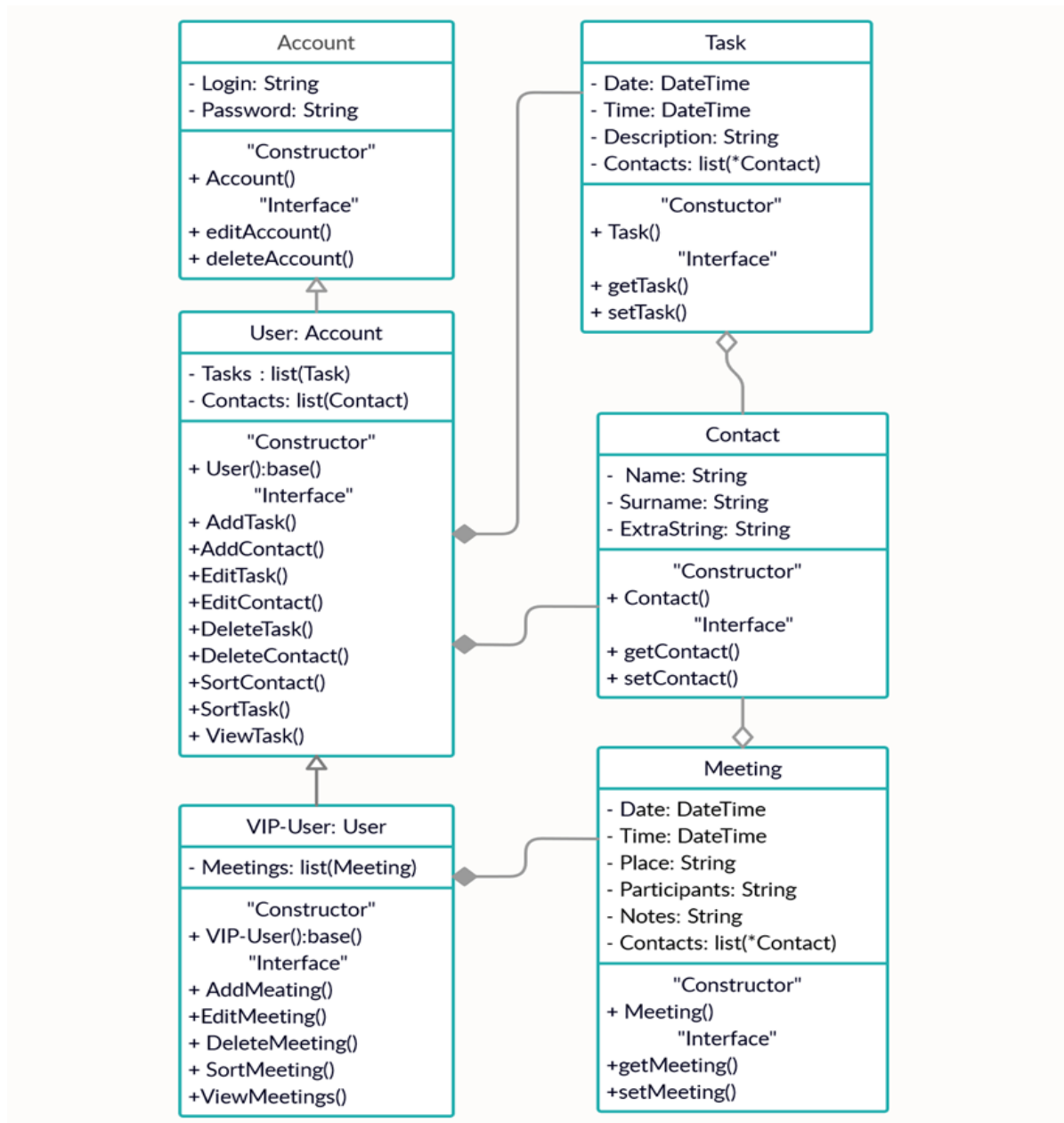


Рисунок 3.1 – Структура у вигляді діаграми класів програмного забезпечення для планування та організації щоденних справ

Основними класами програмного застосунку для планування та організації щоденних справ класи Task, User, Meeting, Contact та інші.

Діаграма класів для програмного рішення, що призначене для впорядкування щоденних справ і керування особистими чи робочими завданнями, відображає структуру основних об'єктів та зв'язків між ними, які забезпечують роботу ключових функцій планувальника. У такій моделі клас Task зазвичай виступає центральною ланкою — він описує окреме завдання, яке користувач створює для виконання у певний проміжок часу, і містить властивості для назви, опису, стану виконання, дедлайнів, пріоритетів та, за потреби, прикріплених файлів чи нагадувань.

Клас User виконує роль носія інформації про того, хто користується застосунком, і включає дані про обліковий запис, налаштування профілю та пов'язаний список усіх завдань, зустрічей чи контактів. Він пов'язаний з класом Task через відношення асоціації, яке показує, що кожен користувач може мати кілька завдань, але кожне завдання належить лише одному користувачу.

Окремий клас Meeting потрібен для того, щоб описувати події, які відрізняються від простих завдань, адже зустрічі передбачають конкретний час, тривалість і список запрошених контактів або учасників. Зазвичай цей клас має зв'язки як з User, так і з Contact, демонструючи, хто організовує подію і хто у ній бере участь.

Клас Contact служить для зберігання даних про сторонніх осіб чи колег, яких можна додати до зустрічі або прикріпити до завдання для зручності комунікації. Такий об'єкт часто містить базову контактну інформацію — ім'я, електронну пошту, номер телефону — і може бути пов'язаний одразу з кількома зустрічами чи завданнями.

Усі ці класи разом утворюють цілісну картину, що відображає логіку зберігання й взаємодії даних, завдяки чому стає зрозуміло, як користувач взаємодіє з програмою, планує робочий день, веде список контактів і контролює виконання всіх запланованих дій.

Діаграму компонентів програмного застосунку для планування та організації щоденних справ наведено на рис. 3.2.

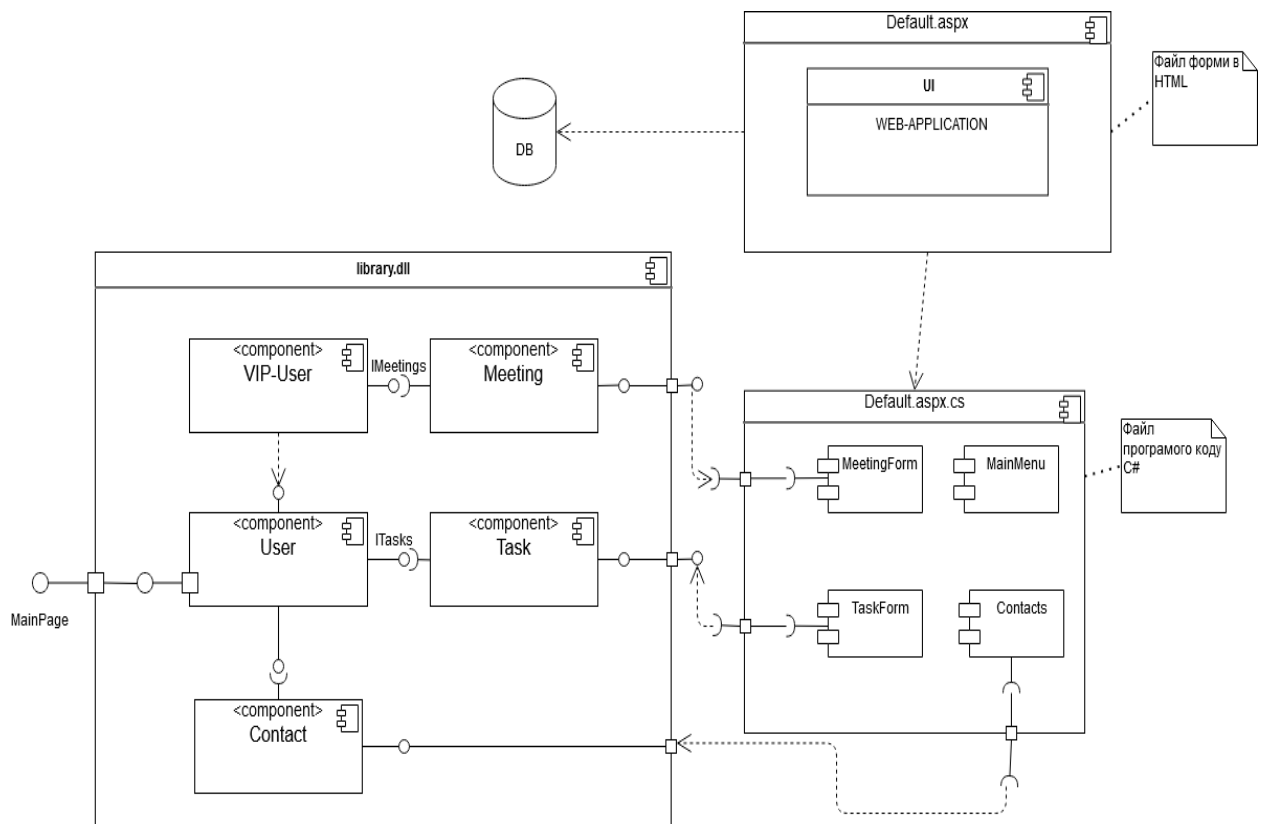


Рисунок 3.2 – Діаграма компонентів програмного застосунку для планування та організації щоденних справ

Діаграма компонентів програмного застосунку для планування та організації щоденних справ демонструє, як окремі частини програми взаємодіють між собою та які функції вони виконують у межах загальної архітектури.

У цьому випадку важливими елементами є Default.aspx, library.dll, MainPage і база даних, кожен з яких відповідає за окрему роль у роботі всього програмного комплексу.

Компонент Default.aspx у такій структурі виступає стартовою точкою для користувача програмного застосунку для планування та організації щоденних справ, коли той звертається до вебінтерфейсу застосунку. Він реалізує представлення початкової сторінки і відповідає за взаємодію із

клієнтом через браузер, обробляючи запити, ініціюючи відображення потрібних елементів і передаючи дані далі у внутрішню логіку.

Компонент `MainPage` програмного застосунку для планування та організації щоденних справ виконує функцію основної сторінки інтерфейсу користувача. Саме тут розміщується більшість ключових елементів — списки завдань, календарі, нагадування, форми створення нових подій чи зустрічей. `MainPage` взаємодіє з логікою, що реалізована у бібліотеці `library.dll`, і безпосередньо залежить від її функцій для обробки бізнес-правил.

Компонент `library.dll` є окремим модулем програмного застосунку для планування та організації щоденних справ, що містить основні класи та методи, які реалізують базову функціональність: створення, редагування, видалення завдань, опрацювання списків контактів, генерацію сповіщень і перевірку умов виконання. Саме бібліотека відповідає за зв'язок між представленням і збереженням інформації, працюючи як проміжна ланка, яка обробляє запити від `MainPage` або `Default.aspx` і передає потрібні дані до бази.

Важливим елементом програмного застосунку для планування та організації щоденних справ є база даних, що забезпечує надійне зберігання всієї інформації — облікових записів користувачів, завдань, розкладу зустрічей, контактів і журналів змін. Зв'язок між бібліотекою та базою даних гарантує, що усі зміни вносяться коректно, а запити від інтерфейсу отримують актуальні дані у відповідь.

Усі ці компоненти програмного застосунку для планування та організації щоденних справ разом формують чітко розмежовану структуру, де інтерфейс, бізнес-логіка та дані функціонують як окремі частини, проте взаємодіють у єдиній системі. Такий поділ полегшує супровід і розвиток програмного застосунку для планування та організації щоденних справ, дає змогу масштабувати окремі елементи й підвищує стабільність роботи всього інструменту для організації повсякденних справ.

3.2 Функціонування програмного застосунку для планування та організації щоденних справ

Функціонування програмного застосунку для планування та організації щоденних справ подамо за допомогою діаграми послідовностей, зображеної на рис. 3.3.

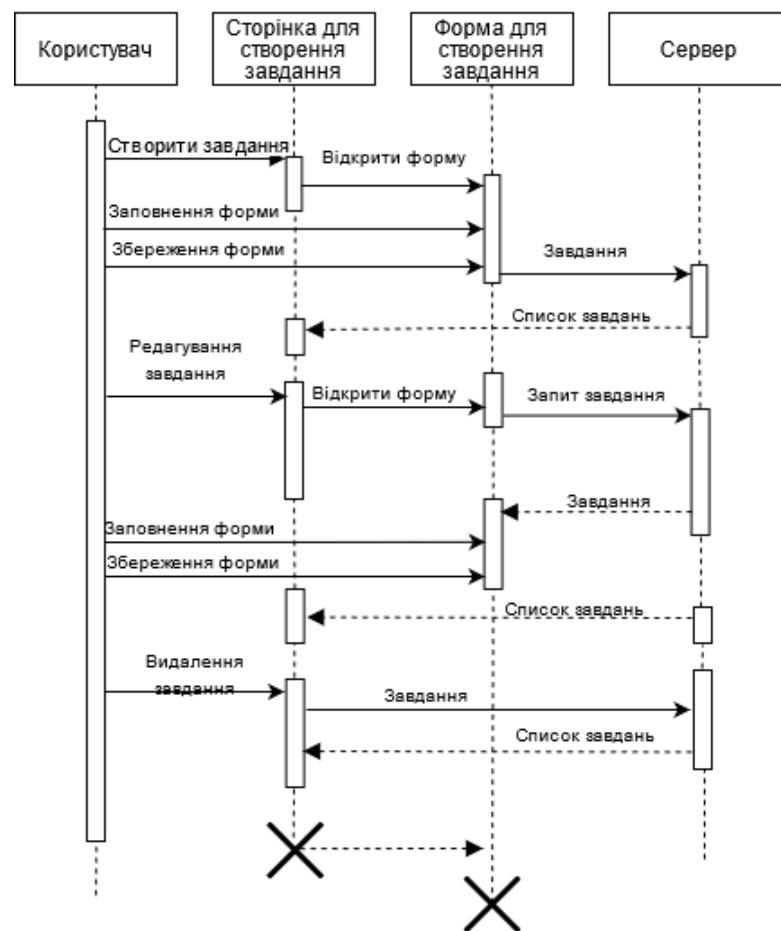


Рисунок 3.3 – Схема функціонування програмного застосунку для планування та організації щоденних справ у вигляді діаграми послідовностей

Діаграма послідовностей, що ілюструє процес взаємодії користувача із системою під час створення нового завдання у програмному застосунку для планування повсякденних справ, показує покроковий обмін повідомленнями між ключовими об'єктами. У цьому випадку головними учасниками взаємодії є Користувач, Сторінка для створення завдання, Форма для створення

завдання та Сервер, кожен із яких виконує чітко визначену роль у ланцюжку дій.

Процес починається з того, що користувач ініціює відкриття сторінки для створення нового завдання, надсилаючи відповідний запит. Сторінка завантажується і відображає форму, яка містить усі необхідні поля: назву, опис, дату, час нагадування, пріоритет та інші атрибути, що користувач планує заповнити. Як тільки користувач починає вводити дані у форму, взаємодія з боку системи може включати додаткові перевірки на коректність введеної інформації — наприклад, контроль правильності дати чи відсутності порожніх обов'язкових полів.

Після заповнення всіх полів форму надсилають для обробки. У цей момент формується запит, який переходить від форми до сервера. Сервер, отримавши інформацію, перевіряє валідність даних ще раз на своєму боці, оскільки контроль лише у браузері не гарантує повного захисту від помилок або спроб введення некоректних значень. Якщо перевірка пройдена успішно, сервер створює запис нового завдання у базі даних і генерує відповідь, що підтверджує успішне збереження.

Після обробки відповіді форма одержує підтвердження, що дані прийнято і завдання створено. Відповідно сторінка для створення завдання може автоматично оновлюватися або переадресувати користувача до списку всіх завдань, відображаючи новий запис серед інших. Користувач у результаті бачить підтвердження успішного додавання завдання та може одразу перейти до його редагування чи планування додаткових подій.

Завдяки такій послідовності взаємодій діаграма чітко показує, як крок за кроком передається інформація від користувача до сервера й назад, що дозволяє впевнено забезпечувати цілісність даних, швидкий зворотний зв'язок та зручність використання застосунку для щоденного планування справ.

Діаграму станів програмного застосунку для планування та організації щоденних справ наведено на рис. 3.4.

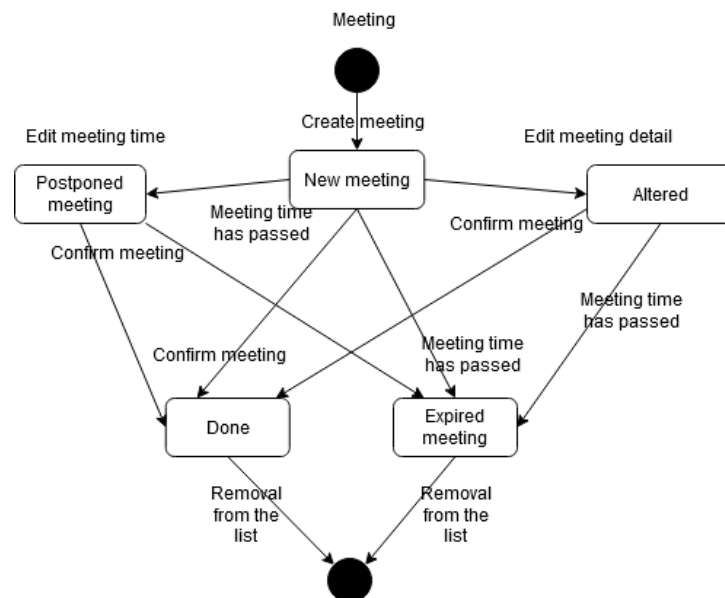


Рисунок 3.4 – Діаграма станів програмного застосунку для планування та організації щоденних справ

Діаграму кооперації програмного застосунку для планування та організації щоденних справ наведено на рис. 3.5.

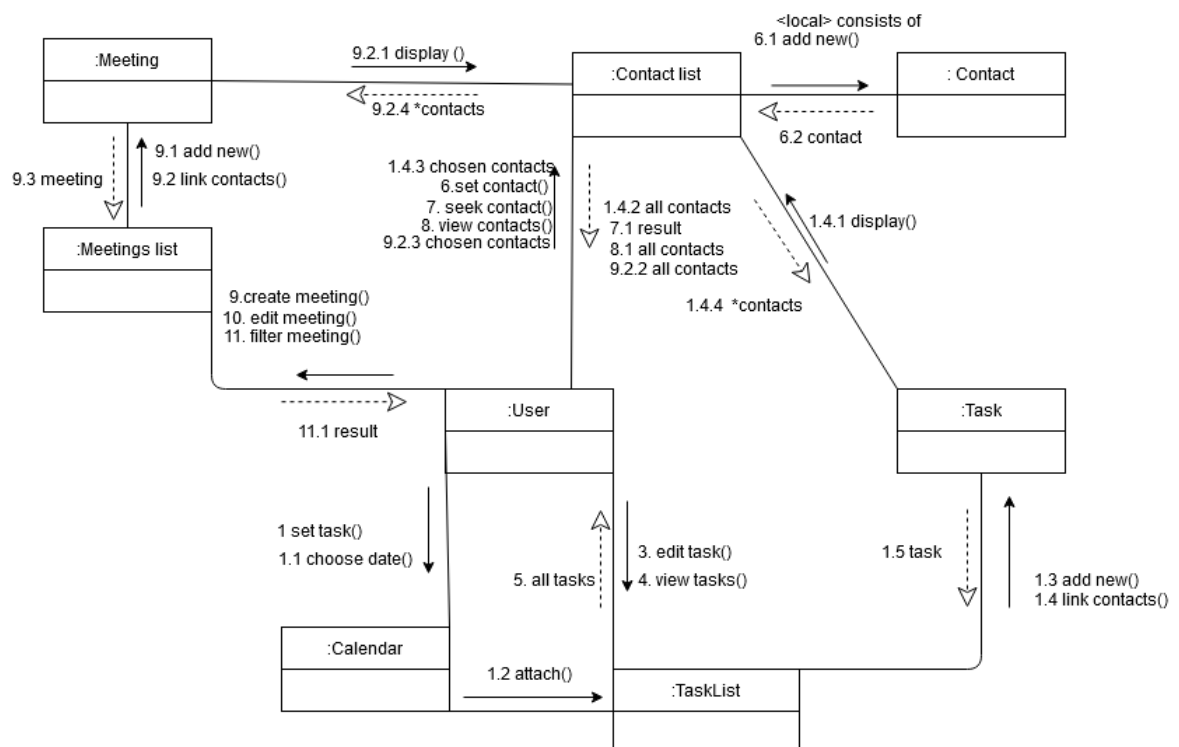


Рисунок 3.5 – Діаграма кооперації програмного застосунку для планування та організації щоденних справ

Діаграма кооперації використовується для опису взаємодії складових елементів застосунку, що допомагає впорядковувати повсякденні завдання та координувати особисті чи робочі плани, ілюструє, як об'єкти обмінюються повідомленнями для виконання однієї чи кількох типових функцій. На такій схемі зазвичай відображається, яким чином ініціатива користувача, наприклад, створити чи відредагувати елемент розкладу, передається між інтерфейсною частиною, модулем, що відповідає за бізнес-логіку, і сховищем даних або зовнішніми службами.

Кооперація демонструє не лише послідовність комунікацій, а й те, як різні частини системи спільно забезпечують виконання кожної дії — перевірку введених параметрів, збереження змін, оновлення відображення або відправлення сповіщень про заплановані події. Основна увага приділяється тому, щоб показати, як окремі об'єкти, що існують у межах застосунку, співпрацюють між собою для підтримки актуального стану розкладу користувача.

Кожен об'єкт на такій діаграмі виконує власну роль: одні відповідають за збір інформації з боку користувача, інші — за обробку та перевірку, ще інші — за збереження і передачу даних далі. Завдяки кооперації видно, як крок за кроком дії користувача перетворюються на конкретний результат — оновлення списку завдань, налаштування нагадувань чи організацію зустрічей із потрібними контактами. Така модель допомагає розробникам і аналітикам узгодити очікувану логіку роботи й забезпечити злагоджену взаємодію всіх частин інструменту для щоденного планування.

3.3 Розробка бази даних програмного забезпечення для планування та організації щоденних справ

Для організації надійного збереження та подальшої обробки всієї інформації, що використовується у процесі планування та впорядкування щоденних справ, доцільно застосовувати сервер керування базами даних

MySQL. Вся взаємодія з даними відбувається шляхом виконання запитів, складених мовою SQL, що забезпечує гнучкість і точність роботи з таблицями та зв'язками між ними. Структура даних охоплює кілька логічно пов'язаних таблиць, кожна з яких виконує свою окрему функцію та взаємодіє з іншими через зовнішні ключі.

Схему бази даних програмного забезпечення для планування та організації щоденних справ наведено на рис. 3.6.

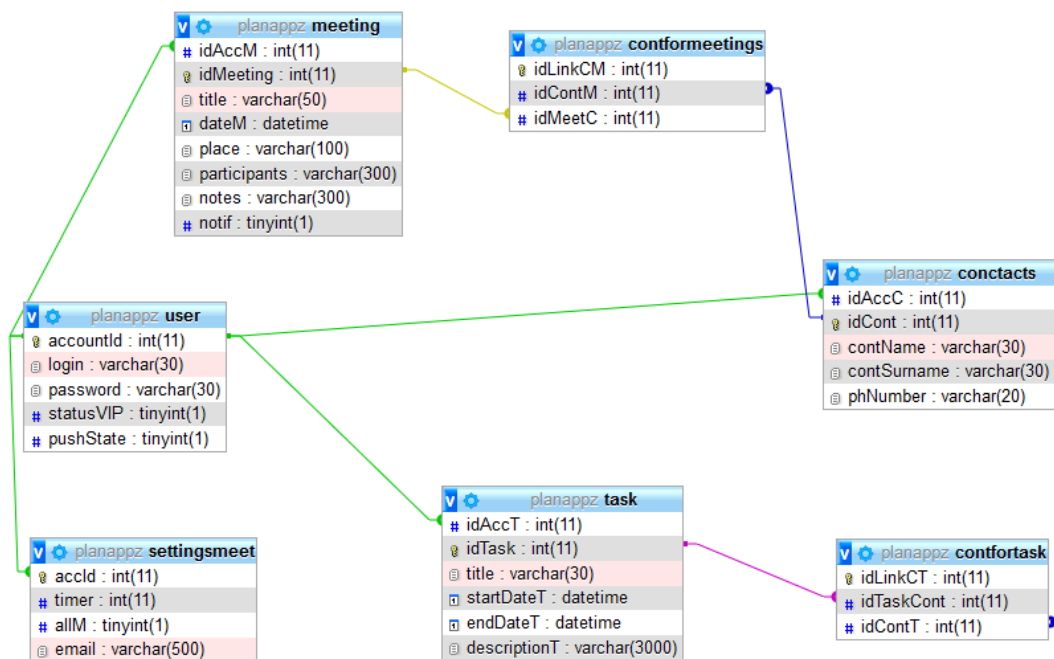


Рисунок 3.6 – Схема бази даних програмного забезпечення для планування та організації щоденних справ

Основними сутностями бази даних обрано такі: користувач, задача, зустріч, контакти та інші.

Основою сховища виступає таблиця, де накопичується вся інформація про користувачів, які працюють із застосунком. Кожне завдання, що додається або редагується, фіксується у таблиці, що містить усі необхідні атрибути справи та посиляється на користувача через відповідний зовнішній ключ, що поєднується з унікальним ідентифікатором облікового запису. Аналогічним чином побудовано таблицю для збереження зустрічей, які також

прив'язуються до профілю того, хто їх створив. Для забезпечення можливості пов'язувати кожен зустріч одразу з кількома контактами використовується спеціальна зв'язувальна структура, що описує багатозв'язні відношення між учасниками та подіями.

Контактна інформація зберігається окремо, однак усі записи, що стосуються певного користувача, мають посилання на обліковий запис, чим забезпечується чіткий поділ даних за власниками. Задля можливості приєднання контактів і до завдань, реалізовано ще одну таблицю, що дозволяє закріплювати потрібну кількість зв'язків між списком справ і контактами, яких вони стосуються. Додатково передбачено таблицю, що відповідає за збереження параметрів поштового сповіщення для окремої категорії користувачів, яким потрібна розширена система нагадувань і індивідуальні налаштування повідомлень.

Завдяки такій структурі кожен елемент даних (окреме завдання, запланована зустріч, список контактів) зберігається у логічно відокремленій частині бази, але у той же час усі таблиці об'єднані ключами для швидкого доступу та цілісної роботи всієї системи планування.

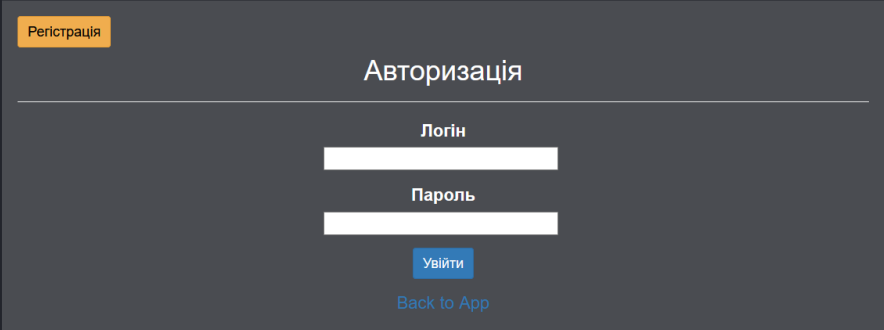
3.4 Проєктування інтерфейсу програмного забезпечення для планування та організації щоденних справ

Проєктування інтерфейсу взаємодії користувача з програмним забезпеченням для планування та організації щоденних справ виконано з урахуванням вимог технічного завдання. При розробленні програмного забезпечення для планування та організації щоденних справ враховані функціональні вимоги до програми:

- підтримка можливості подання інформації для планування та організації щоденних справ користувача;
- підтримка можливості авторизації та реєстрації користувачів;
- можливість відображення календаря зустрічей;

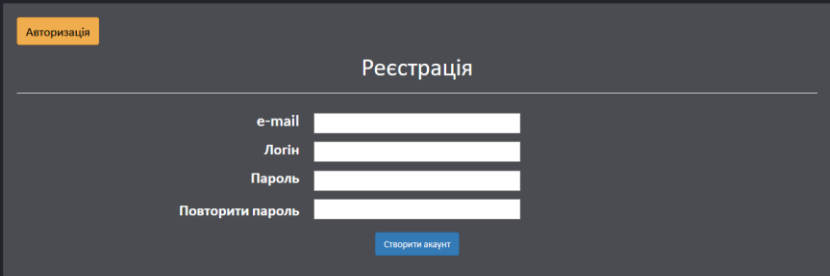
- можливість відображення інформації про діяльність та справи користувача;
- підтримка можливості зберігання інформації про контакти користувача;
- підтримка можливості редагування налаштувань.

Інтерфейси форм авторизації та реєстрації програмного забезпечення для планування та організації щоденних справ наведено на рис. 3.7 та 3.8.



The screenshot shows a login form titled "Авторизація" (Authorization) on a dark background. In the top left corner, there is an orange button labeled "Регістрація" (Registration). The form contains two white input fields: "Логін" (Login) and "Пароль" (Password). Below the password field is a blue button labeled "Увійти" (Login) and a blue link labeled "Back to App".

Рисунок 3.7 – Інтерфейс форми авторизації програмного забезпечення для планування та організації щоденних справ



The screenshot shows a registration form titled "Реєстрація" (Registration) on a dark background. In the top left corner, there is an orange button labeled "Авторизація" (Authorization). The form contains four white input fields: "e-mail", "Логін" (Login), "Пароль" (Password), and "Повторити пароль" (Repeat password). Below the last field is a blue button labeled "Створити акаунт" (Create account).

Рисунок 3.8 – Форма реєстрації програмного застосунку для планування та організації щоденних справ

Сторінка Календар зі справами користувача наведена на рис. 3.9.

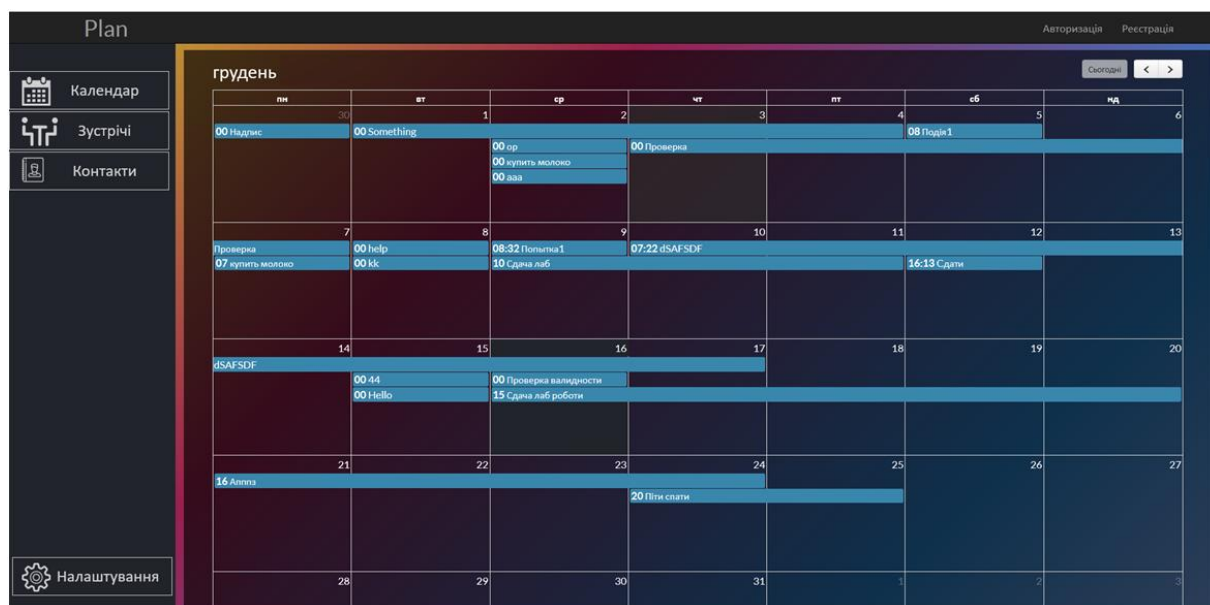


Рисунок 3.9 – Сторінка Календар зі справами користувача

Приклад сторінки з зустрічами користувача наведено на рис. 3.10.

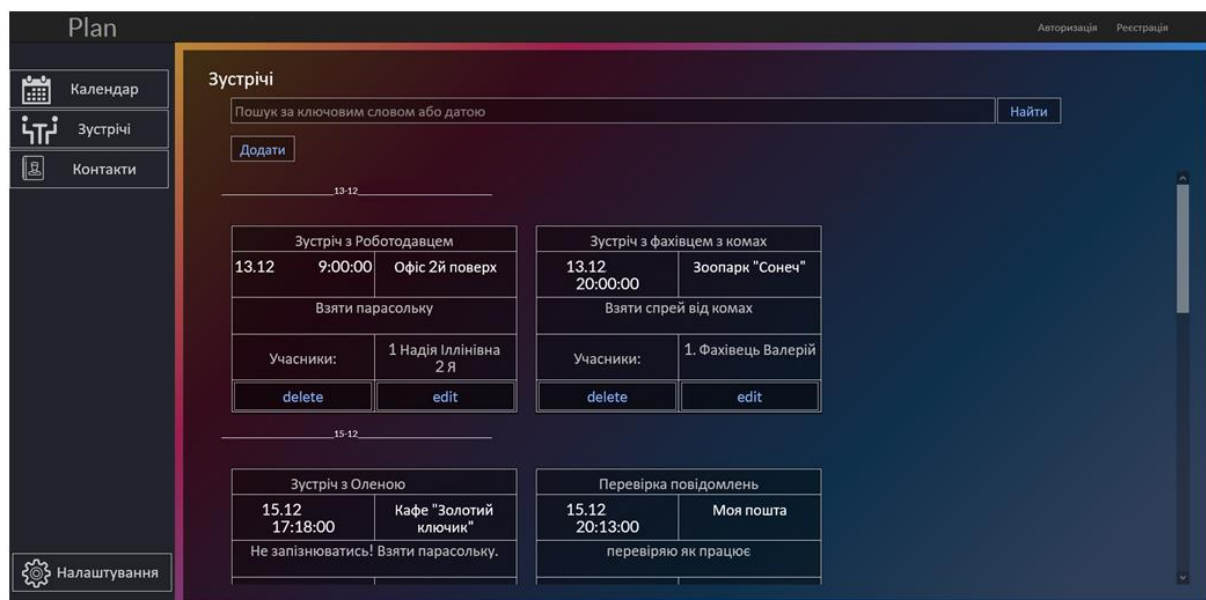


Рисунок 3.10 – Сторінка з зустрічами користувача

Форма редагування інформації про зустрічі користувача наведена на рис. 3.11.

Зустрічі

Пошук за ключовим словом або датою

Додати

16-12

Зустріч з батьками	
16.12 15:00:00	Ресторан "Мадрид"
Ювілей батька	
Учасники:	1 Тітка Надя 2 Мама
delete	edit

Опис зустрічі

Зустріч з батьками

Початок зустрічі

16-12 15:00

Місце зустрічі

Ресторан "Мадрид"

Учасники

1 Тітка Надя
2 Мама
3 Тато

Подробиці

Ювілей батька

Додати контакти

Надіслати повідомлення ☒

Delete Save

Рисунок 3.11 – Форма редагування інформації про зустрічі користувача

3.5 Висновки за розділом 3

Розроблено програмне забезпечення для планування та організації щоденних справ.

Запропоновано структуру програмного забезпечення для планування та організації щоденних справ. Визначено, що основними класами програмного застосунку для планування та організації щоденних справ є класи Task, User, Meeting, Contact та інші.

Функціонування програмного застосунку для планування та організації щоденних справ подано за допомогою діаграми послідовностей, що ілюструє процес взаємодії користувача із системою під час створення нового завдання у програмному застосунку для планування повсякденних справ, показує

покроковий обмін повідомленнями між ключовими об'єктами. Головними учасниками взаємодії є Користувач, Сторінка для створення завдання, Форма для створення завдання та Сервер, кожен із яких виконує чітко визначену роль у ланцюжку дій.

Розроблено базу даних програмного забезпечення для планування та організації щоденних справ. Основними сутностями бази даних обрано такі: користувач, задача, зустріч, контакти та інші.

Виконано проєктування інтерфейсу взаємодії користувача з програмним забезпеченням для планування та організації щоденних справ.

4 ЕКСПЛУАТАЦІЯ, ТЕСТУВАННЯ ТА ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ ПРОГРАМИ

4.1 Призначення й умови застосування програми

Програма призначена для планування та організації щоденних справ. Програма написана на мові C#, яка поєднує об'єктно-орієнтований підхід із широким набором бібліотек, що спрощує реалізацію складної бізнес-логіки та інтеграцію різноманітних сервісів. Завдяки підтримці платформи .NET відкривається можливість легко забезпечувати кросплатформенність і розгортати застосунок не лише на Windows, а й на інших популярних операційних системах. В якості середовища розробки для створення програмного забезпечення для планування та організації щоденних справ обрано Visual Studio через поєднання глибокої інтеграції з мовою C# та широкого набору інструментів для розробки. Крім того, це середовище пропонує інтуїтивний інтерфейс, який спрощує налаштування складних рішень і пришвидшує перевірку коду завдяки вбудованим засобам відлагодження та тестування.

Програмне забезпечення для планування та організації щоденних справ забезпечує виконання таких функцій:

- підтримка можливості подання інформації для планування та організації щоденних справ користувача;
- підтримка можливості авторизації та реєстрації користувачів;
- можливість відображення календаря зустрічей;
- можливість відображення інформації про діяльність та справи користувача;
- підтримка можливості зберігання інформації про контакти користувача;
- підтримка можливості редагування налаштувань.

Для безперебійного функціонування програмного застосунку для планування та організації щоденних справ необхідно виконати вимоги як до

апаратної, так і до програмної частини інфраструктури. Основна логіка та обробка користувацьких запитів можуть розгортатися на окремих серверах, що працюють у середовищі з достатнім резервом обчислювальних ресурсів. Ключову роль відіграють сучасні процесори, що підтримують частоти, достатні для швидкого опрацювання одночасних запитів, а обсяг оперативної пам'яті має покривати потреби багатопотокової обробки та кешування активних сесій. Особливі вимоги висуваються до характеристик накопичувачів, оскільки високі швидкості введення-виведення та низька затримка доступу гарантують швидке збереження та відновлення великого обсягу інформації, що постійно оновлюється користувачами під час планування завдань і подій.

Для основної серверної інфраструктури важливе дотримання балансу між кількістю віртуальних і фізичних ядер процесора, що впливає на стабільність при зростанні кількості одночасних запитів. Відмовостійкість досягається за рахунок дублювання ключових вузлів за схемою, що дозволяє у разі збою одного елемента автоматично передати навантаження на інший, зберігаючи працездатність усієї системи.

З боку програмного забезпечення обов'язковою умовою є встановлення актуальної редакції самої програми, що розробляється і підтримується у середовищі Microsoft Visual Studio, причому бажано використовувати найсвіжішу доступну версію для отримання всіх актуальних оновлень і виправлень. Ключову роль відіграє інтеграція з платформою ASP.NET, у межах якої забезпечується чітка логіка представлення даних. Для візуальної частини застосовуються перевірені пакети стилів, що відповідають за правильне форматування інтерфейсів. Сукупність цих компонентів формує основу стабільної роботи сервісу, дозволяючи користувачам безперервно керувати своїми справами та планами без втрат даних і технічних збоїв.

Функціональні характеристики розробленого програмного забезпечення для планування та організації щоденних справ наведено у

технічному завданні (додаток А). Фрагмент тексту програмного забезпечення для планування та організації щоденних справ наведено у додатку Б.

4.2 Характеристики програми для планування та організації щоденних справ

Програмний застосунок для планування та організації щоденних справ складається з серверної та клієнтської частин.

Архітектура рішення орієнтована на швидкий обмін даними між інтерфейсом і серверною частиною, що дозволяє своєчасно синхронізувати зміни у записах між різними пристроями. Фізичну структуру програмного застосунку для планування та організації щоденних справ наведено на рис. 4.1.

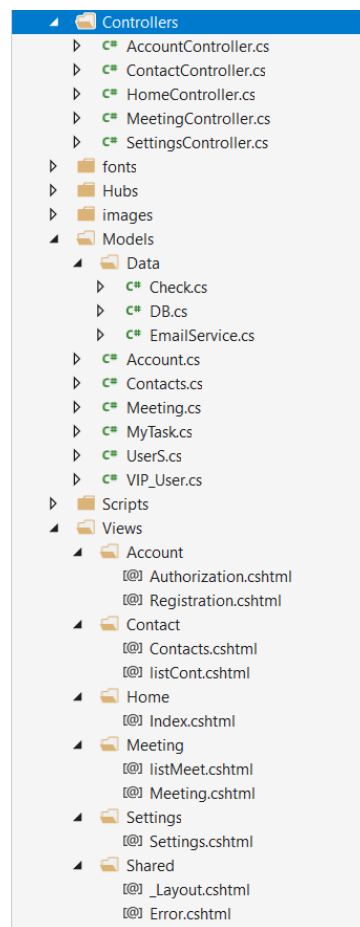


Рисунок 4.1 – Фізична структура програмного застосунку для планування та організації щоденних справ

Його основне призначення полягає у тому, щоб забезпечувати користувача зручними інструментами для фіксації завдань, налаштування нагадувань та організації зустрічей у зручному цифровому просторі.

Інтерфейс застосунку побудовано так, щоб навіть новий користувач міг легко зрозуміти принципи роботи та швидко знаходити потрібні функції. Основна логіка передбачає роботу з кількома модулями, що відповідають за різні аспекти управління часом — створення та редагування завдань, додавання зустрічей із зазначенням учасників, а також ведення списку контактів для спрощення організації спільних подій.

Надійність застосунку підтримується за рахунок реалізації багаторівневої перевірки введених даних і системи резервного копіювання, що унеможлиблює втрату важливої інформації навіть у разі технічних збоїв.

Завдяки такій сукупності характеристик програмне забезпечення перетворюється на надійного помічника, здатного підлаштовуватись під ритм життя користувача і знижувати ризики пропуску важливих подій або завдань.

4.3 Інструкція по експлуатації програми

4.3.1 Звернення до програми

Щоб отримати доступ до функцій вебзастосунку, користувачі звертаються до спеціального вебресурсу, відкривши його через браузер на своєму комп'ютері за умови підключення до Інтернету. Сам сайт передає звернення до сервера та приймає відповіді у вигляді оброблених запитів різних типів, серед яких головними є запити для отримання інформації та надсилання введених даних. Для запуску самої програми на стороні сервера необхідно знайти головну директорію, де зберігаються всі файли проєкту, і відкрити файл рішення, після чого активується виконання всієї програми для подальшої обробки запитів користувачів.

4.3.2 Вхідні й вихідні дані

Вхідними даними до програмного забезпечення для планування та організації щоденних справ є всі значення, які користувач зазначає у формах, зокрема, набір авторизаційних даних, інформація про задачі користувача, контакти та ін.

Вихідними даними програмного забезпечення для планування та організації щоденних справ є календарі, які групують події та завдання за відповідними датами й часом, сторінки для внесення правок до завдань, списки контактів та ін.

4.3.3 Повідомлення

Користувач програмного забезпечення для планування та організації щоденних справ може отримати такі повідомлення: повідомлення про неуспішну авторизацію, попередження про відсутність певних даних у процесі створення чи редагування задачі та ін.

4.4 Виконання програмного забезпечення для планування та організації щоденних справ

Після запуску програмного застосунку для планування та організації щоденних справ користувачу відображається форма авторизації (рис. 4.2), де необхідно ввести логін та пароль.

Якщо користувач програмного застосунку для планування та організації щоденних справ не зареєстрований, необхідно зареєструватися. Для цього необхідно натиснути на кнопку Регістрація та у новій формі (рис. 4.3) ввести потрібні дані.

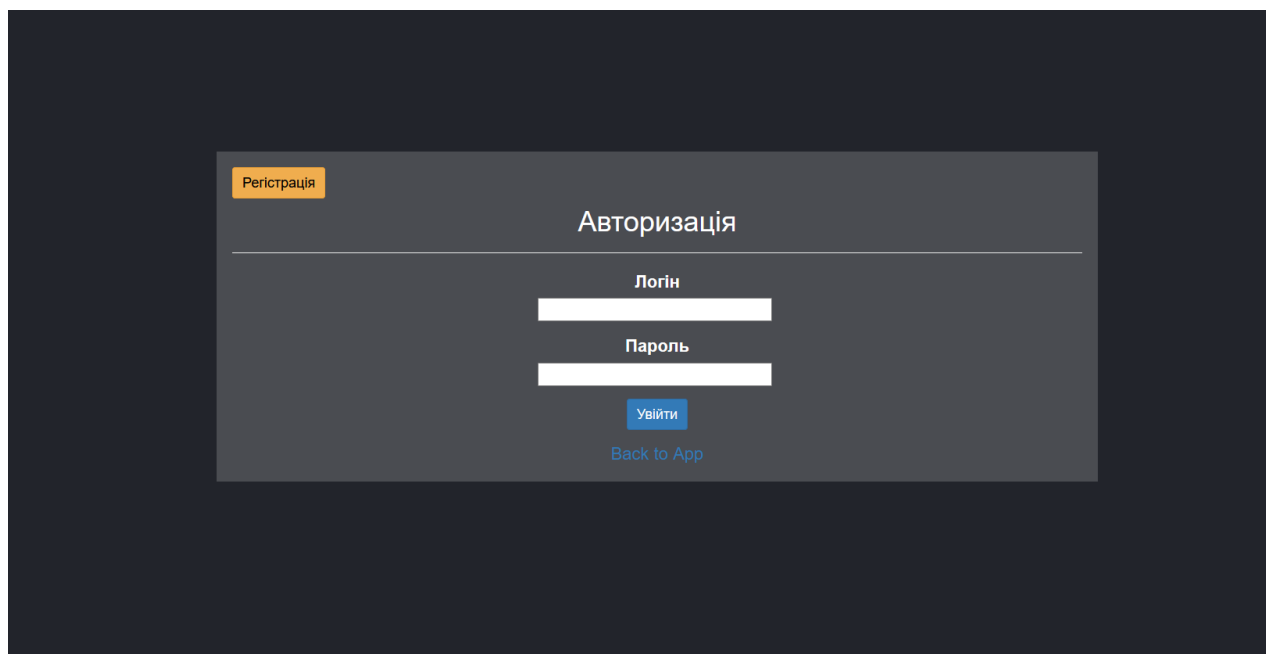


Рисунок 4.2 – Форма авторизації програмного застосунку для планування та організації щоденних справ

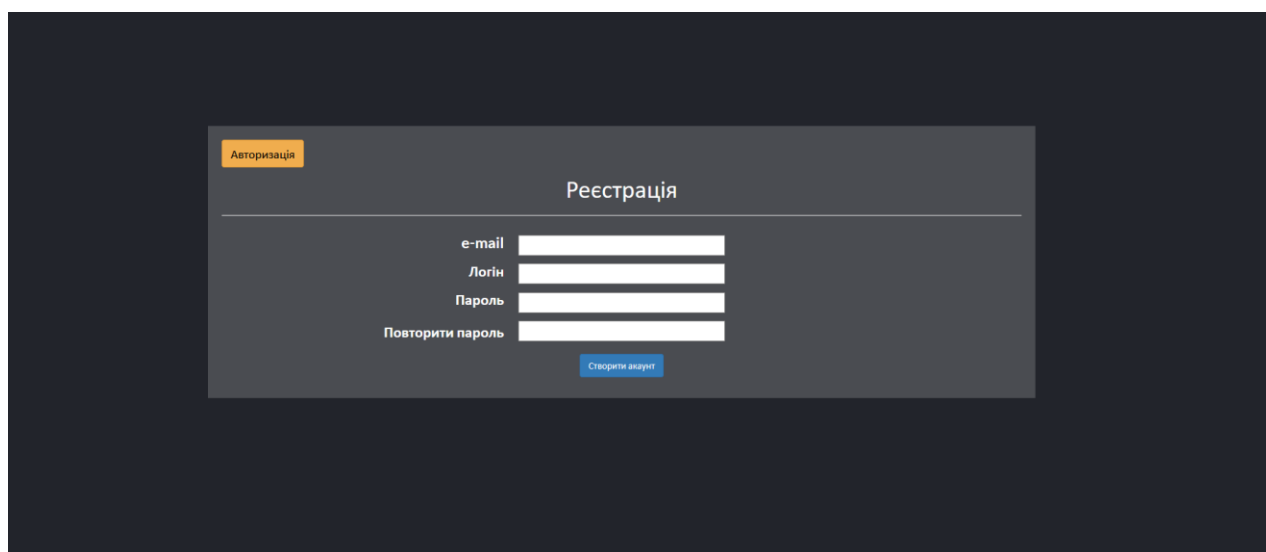


Рисунок 4.3 – Форма реєстрації програмного застосунку для планування та організації щоденних справ

Після реєстрації користувач може працювати з календарем справ (рис. 4.4). У календарі відображено інформацію про заплановані події у хронологічному порядку. Кожній даті відповідає відповідна чарунка календаря, де наведено назву подію та запланований час її початку.

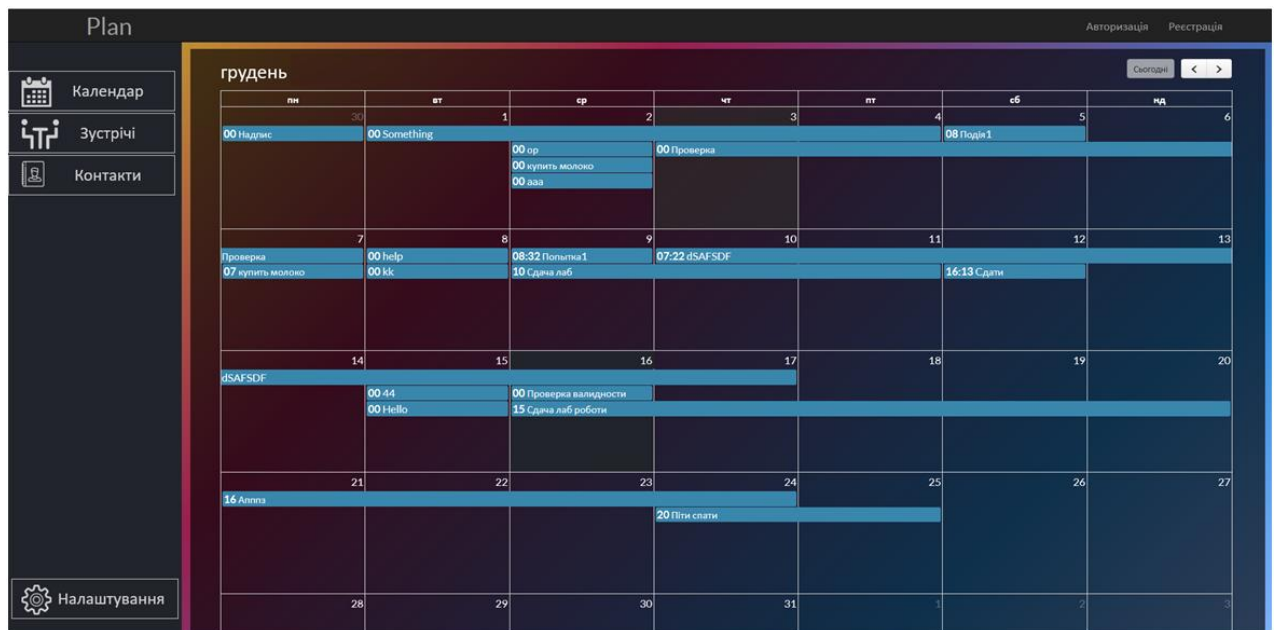


Рисунок 4.4 – Сторінка Календар зі справами користувача

При виборі пункту меню Зустрічі відображається сторінка зустрічей користувача (рис. 4.5), де надано детальну інформацію про кожну заплановану зустріч (назва, дата та час, опис, учасники).

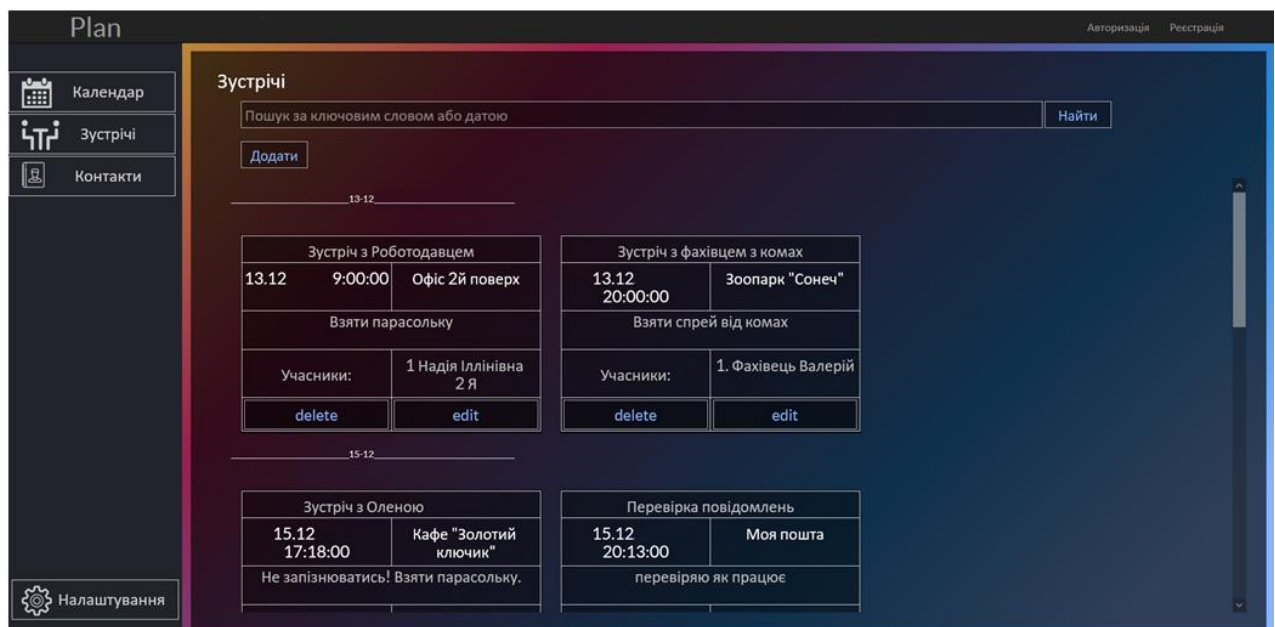


Рисунок 4.5 – Сторінка з зустрічами користувача

При натисканні на кнопку Edit обраної зустрічі користувач переходить до форми редагування інформації про зустріч (рис. 4.6), де є можливість

змінити опис зустрічі, дату та час її початку, місце, учасників, а також за необхідності додати відповідні контакти.

The image shows a web application interface for managing meetings. The main section, titled 'Зустрічі', contains a search bar with the placeholder 'Пошук за ключовим словом або датою' and a 'Додати' (Add) button. Below this is a table listing meetings. The first meeting is 'Зустріч з батьками' (Meeting with parents) on '16.12' at '15:00:00' in 'Ресторан "Мадрид"'. The second is 'Ювілей батька' (Father's birthday). The third lists participants: '1 Тітка Надя', '2 Мама'. Each row has 'delete' and 'edit' buttons. A right-hand sidebar provides a detailed view for editing a meeting. It includes fields for 'Опис зустрічі' (Meeting description), 'Початок зустрічі' (Meeting start) with date and time pickers, 'Місце зустрічі' (Meeting location), 'Учасники' (Participants) as a list, 'Подробиці' (Details) as a text area, 'Додати контакти' (Add contacts) as a text input, and a toggle for 'Надіслати повідомлення' (Send notifications). At the bottom of the sidebar are 'Delete' and 'Save' buttons.

Зустріч з батьками	
16.12 15:00:00	Ресторан "Мадрид"
Ювілей батька	
Учасники:	1 Тітка Надя 2 Мама
delete	edit

Опис зустрічі

Зустріч з батьками

Початок зустрічі

16-12 15:00

Місце зустрічі

Ресторан "Мадрид"

Учасники

1 Тітка Надя
2 Мама
3 Тато

Подробиці

Ювілей батька

Додати контакти

Надіслати повідомлення ☒

Delete Save

Рисунок 4.6 – Форма редагування інформації про зустрічі користувача

При виборі пункту меню Контакти відображаються контакти користувача (рис. 4.7). При натисканні відповідних кнопок можна редагувати контакти чи видаляти непотрібні з них.

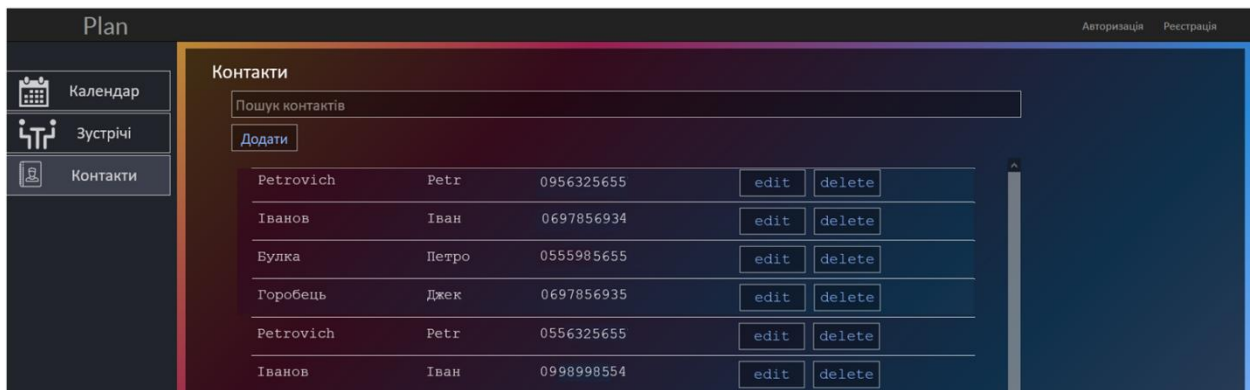


Рисунок 4.7 – Сторінка з контактами користувача

Користувач має можливість налаштувати деякі параметри роботи системи, зокрема, періодичність надсилання системних повідомлень, за допомогою спеціальної сторінки (рис. 4.8).

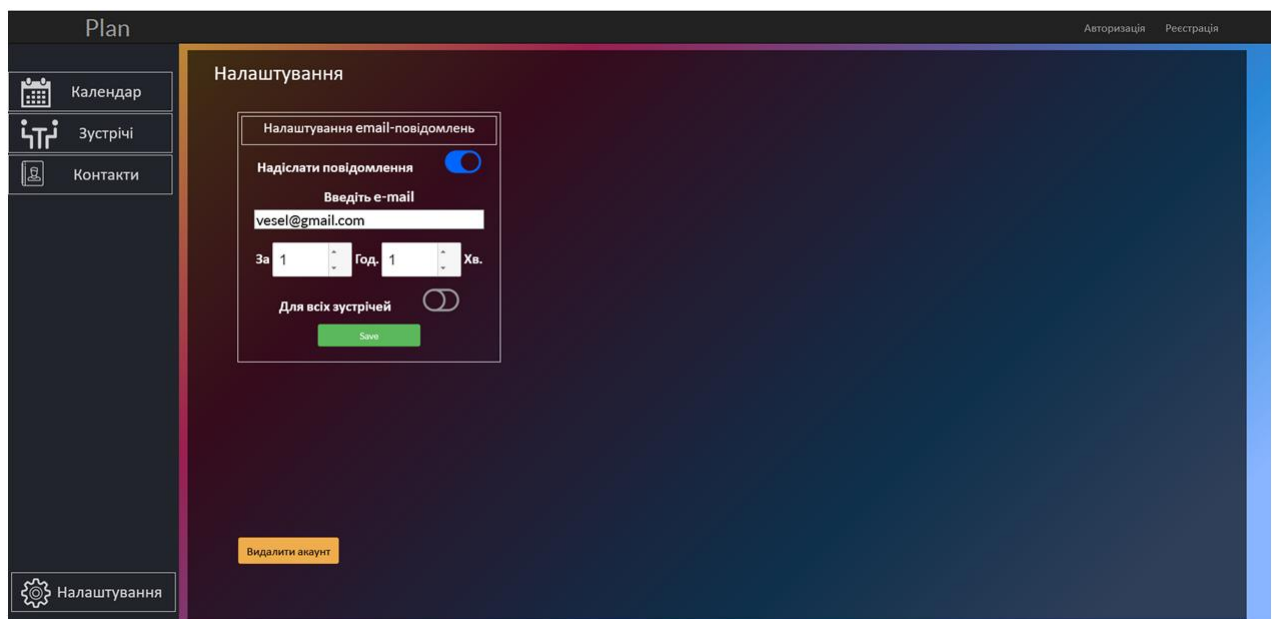


Рисунок 4.8 – Сторінка налаштувань

4.5 Тестування програмного забезпечення для планування та організації щоденних справ

Виконано тестування програмного забезпечення для планування та організації щоденних справ.

Проведемо тестування коректності посилань програмного забезпечення для планування та організації щоденних справ за допомогою застосунку Хепі Link Sleuth. Результати перевірки зображено на рисунку 4.9.

Address	Status	Type	Size	Title
http://code.jquery.com/jquery.min.js	ok	application/j...	95821	
http://localhost:52396/	ok	text/html	3708	Authorization
http://localhost:52396/Account/Registration	ok	text/html	4192	Registration
http://localhost:52396/Calndr/fullcalendar.css	ok	text/css	28215	
http://localhost:52396/Calndr/fullcalendar.js	ok	application/j...	338287	
http://localhost:52396/Calndr/images/ui-bg_glass_65_ffffff_1...	ok	image/png	207	
http://localhost:52396/Calndr/images/ui-bg_highlight-soft_1...	ok	image/png	277	
http://localhost:52396/Calndr/images/ui-bg_highlight-soft_2...	ok	image/png	338	
http://localhost:52396/Calndr/images/ui-bg_highlight-soft_5...	ok	image/png	280	
http://localhost:52396/Calndr/images/ui-icons_0073ea_256x2...	ok	image/png	4549	
http://localhost:52396/Calndr/images/ui-icons_454545_256x2...	ok	image/png	6992	
http://localhost:52396/Calndr/images/ui-icons_666666_256x2...	ok	image/png	6988	
http://localhost:52396/Calndr/images/ui-icons_ff0084_256x24...	ok	image/png	4549	
http://localhost:52396/Calndr/images/ui-icons_ffffff_256x240...	ok	image/png	6299	
http://localhost:52396/Calndr/jquery-ui.css	ok	text/css	34928	
http://localhost:52396/Calndr/jquery-ui.min.js	ok	application/j...	240422	
http://localhost:52396/Calndr/lang/uk.js	ok	application/j...	4215	
http://localhost:52396/Calndr/lib/jquery.min.js	ok	application/j...	84380	
http://localhost:52396/Calndr/lib/moment.min.js	ok	application/j...	39836	
http://localhost:52396/Calndr/main.js	ok	application/j...	10874	
http://localhost:52396/Contact/Contacts	ok	text/html	5819	Контакти
http://localhost:52396/Content/bootstrap.css	ok	text/css	146010	
http://localhost:52396/Content/bootstrap.min.css	ok	text/css	121200	
http://localhost:52396/Content/site.css	ok	text/css	537	
http://localhost:52396/Content/Style.css	ok	text/css	5216	
http://localhost:52396/Content/tiny-toggle.css	ok	text/css	2357	
http://localhost:52396/fonts/glyphicons-halflings-regular.eot	ok	application/v...	20127	
http://localhost:52396/fonts/glyphicons-halflings-regular.eot?	ok	application/v...	20127	
http://localhost:52396/fonts/glyphicons-halflings-regular.svg	ok	image/svg+xml	108738	
http://localhost:52396/fonts/glyphicons-halflings-regular.ttf	ok	application/o...	45404	
http://localhost:52396/fonts/glyphicons-halflings-regular.woff	ok	font/x-woff	23424	
http://localhost:52396/fonts/glyphicons-halflings-regular.woff2	ok	application/f...	18028	
http://localhost:52396/fonts/tinytoggle.eot	ok	application/v...	4588	
http://localhost:52396/fonts/tinytoggle.eot?	ok	application/v...	4588	

Рисунок 4.9 – Результати тестування коректності посилань програмного забезпечення для планування та організації щоденних справ

Виявлено, що програма для планування та організації щоденних справ функціонує правильно та злагоджено. Вона реалізує всі функціональні вимоги та успішно виконує свою основну задачу підтримки процесу планування діяльності.

Розроблене програмне забезпечення дозволяє забезпечувати автоматизацію процесів, пов'язаних з підтримкою планування щоденної діяльності.

4.6 Висновки за розділом 4

Описано програмне забезпечення для планування та організації щоденних справ. Виконано тестування розробленого програмного забезпечення для планування та організації щоденних справ. Результати тестування програмного забезпечення показали, що розроблена програма дозволяє забезпечити автоматизацію процесів, пов'язаних з підтримкою планування щоденної діяльності.

ВИСНОВКИ

В ході виконання дипломної кваліфікаційної роботи магістра було проаналізовано та досліджено процес розробки програмного забезпечення для планування та організації щоденних справ.

Визначено, що програмне забезпечення для планування та організації щоденних справ використовується для керування особистим та робочим розпорядком, допомагаючи користувачеві структурувати завдання, контролювати хід їх виконання та своєчасно реагувати на важливі події. Такі програмні застосунки зазвичай поєднують у собі можливості керування списками справ, інтеграцію календаря, а також засоби нагадувань і сповіщень, завдяки яким можна уникнути пропущених граничних строків виконання задач чи незавершених пунктів плану.

За результатами проведеного аналізу зроблено висновок, що у наш час існує досить багато програмних засобів для планування та організації щоденних справ. Проте деякі програмні засоби для планування та організації щоденних справ вимагають додаткового часу на налаштування та регулярне оновлення даних, що може стати рутинним і навіть обтяжливим для користувача. Деякі програми надто складні у використанні або перенасичені функціями, які не завжди потрібні, що іноді призводить до перевантаження інтерфейсу і втрати мотивації їх використовувати. Окремою проблемою є ризики, пов'язані з конфіденційністю особистої інформації, оскільки дані про зустрічі, контакти чи завдання часто зберігаються у хмарних сервісах і можуть бути вразливими до несанкціонованого доступу. Крім того, надмірне використання цифрових програмних планувальників може призвести до втрати навичок самоорганізації без допоміжних інструментів і сформувати залежність від автоматизованих нагадувань. Тому актуальною є розробка програмного забезпечення для планування та організації щоденних справ.

Сформульовано функціональні вимоги до програмного забезпечення для планування та організації щоденних справ.

Для реалізації програмного забезпечення для планування та організації щоденних справ обрано мову програмування C#, яка поєднує об'єктно-орієнтований підхід із широким набором бібліотек, що спрощує реалізацію складної бізнес-логіки та інтеграцію різноманітних сервісів. Завдяки підтримці платформи .NET відкривається можливість легко забезпечувати кросплатформенність і розгортати застосунок не лише на Windows, а й на інших популярних операційних системах.

Для створення програмного забезпечення для планування та організації щоденних справ обрано середовище розробки Visual Studio через поєднання глибокої інтеграції з мовою C# та широкого набору інструментів для розробки. Крім того, це середовище пропонує інтуїтивний інтерфейс, який спрощує налаштування складних рішень і пришвидшує перевірку коду завдяки вбудованим засобам відлагодження та тестування. Велика кількість розширень і плагінів дозволяє адаптувати робочий простір під конкретні потреби проєкту та легко впроваджувати додаткові функції без зайвих витрат часу.

Новизна роботи полягає у тому, що створено модель взаємодії користувача з програмним забезпеченням для планування та організації щоденних справ, яка подана у вигляді діаграми прецедентів, та дозволяє розроблювати програмне забезпечення для планування та організації щоденних справ.

Практична цінність роботи полягає у тому, що створено програмне забезпечення для планування та організації щоденних справ.

Запропоновано структуру програмного забезпечення для планування та організації щоденних справ. Визначено, що основними класами програмного застосунку для планування та організації щоденних справ класи Task, User, Meeting, Contact та інші.

Функціонування програмного застосунку для планування та організації щоденних справ подано за допомогою діаграми послідовностей, що ілюструє процес взаємодії користувача із системою під час створення нового завдання у

програмному застосунку для планування повсякденних справ, показує покроковий обмін повідомленнями між ключовими об'єктами. Головними учасниками взаємодії є Користувач, Сторінка для створення завдання, Форма для створення завдання та Сервер, кожен із яких виконує чітко визначену роль у ланцюжку дій.

Розроблено базу даних програмного забезпечення для планування та організації щоденних справ. Основними сутностями бази даних обрано такі: користувач, задача, зустріч, контакти та інші.

Виконано проєктування інтерфейсу взаємодії користувача з програмним забезпеченням для планування та організації щоденних справ.

Виконано тестування розробленого програмного забезпечення для планування та організації щоденних справ. Результати тестування програмного забезпечення показали, що розроблена програма дозволяє забезпечити автоматизацію процесів, пов'язаних з підтримкою планування щоденної діяльності.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Task Management [Electronic resource]. – Access mode: <https://www.projectmanager.com/guides/task-management>.
2. Maximize Your Productivity With an Effective Daily Plan [Electronic resource]. – Access mode: <https://www.kumospace.com/blog/daily-plan>.
3. Manage the Planner app for your organization in Microsoft Teams [Electronic resource]. – Access mode: <https://learn.microsoft.com/en-us/microsoftteams/manage-planner-app>.
4. Weekly Planner More Than Just a To Do List [Electronic resource]. – Access mode: <https://weekplan.net/>.
5. How to make a daily schedule: Template and examples [Electronic resource]. – Access mode: <https://asana.com/resources/daily-schedule-template>.
6. Daily planner templates [Electronic resource]. – Access mode: <https://www.canva.com/planners/templates/daily/?msockid=31c9e1a14eb869ce3f80f78e4f196836>.
7. How to Keep Track of Tasks at Work [Electronic resource]. – Access mode: <https://teamhood.com/project-management/how-to-keep-track-of-tasks-at-work/>.
8. Best Free Daily Planner Apps [Electronic resource]. – Access mode: <https://affine.pro/blog/15-best-free-daily-planner-apps-features-free>.
9. The Best Daily Planner Apps [Electronic resource]. – Access mode: <https://theprocesshacker.com/blog/daily-planner-apps/>.
10. Daily Task Trackers to Organize Your Work [Electronic resource]. – Access mode: <https://teamhood.com/productivity/daily-task-tracker-software/>.
11. Best Planner Apps To Organize Work Better [Electronic resource]. – Access mode: <https://toggl.com/blog/best-planner-apps>.
12. Daily Planner Apps for Productivity [Electronic resource]. – Access mode: <https://plaky.com/blog/best-daily-planner-apps/>.

13. The Best Organization Apps for Personal & Professional Use: Set Recurring Tasks, Create a To-Do List & More for Free [Electronic resource]. – Access mode: <https://www.cloudwards.net/best-organization-apps/>.

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

15. Devonthink [Electronic resource]. – Access mode: <https://www.devontechnologies.com/apps/devonthink>.

16. Sunsama [Electronic resource]. – Access mode: <https://www.sunsama.com/>.

17. C# 7.0 All-In-One For Dummies Cheat Sheet [Electronic resource]. – Access mode: <https://www.dummies.com/article/technology/programming-web-design/csharp/c-7-0-one-dummies-cheat-sheet-248715/>.

18. The Comprehensive, Expert Guide to C# Language Programming [Electronic resource]. – Access mode: <https://essentialcsharp.com/home>.

19. What can you do with Visual Studio [Electronic resource]. – Access mode: <https://visualstudio.microsoft.com/vs/features/>.

20. What is Visual Studio? All You Need to Know [Electronic resource]. – Access mode: <https://blog.hubspot.com/website/what-is-visual-studio>.

ДОДАТОК А
Технічне завдання

Вступ

Програмне забезпечення може використовуватися для планування та організації щоденних справ.

A.1 Підстава для розробки

Підставою для розробки є завдання на дипломну кваліфікаційну роботу на тему «Дослідження та програмна реалізація застосунку для планування та організації щоденних справ», затверджене наказом Національного університету «Запорізька політехніка» № 447 від 30 вересня 2025 р.

A.2 Призначення розробки

Програмний продукт призначений для планування та організації щоденних справ.

A.3 Основні вимоги до програми, що розробляється

A.3.1 Вимоги до функціональних характеристик

Програмне забезпечення для планування та організації щоденних справ забезпечує виконання таких функцій:

- підтримка можливості подання інформації для планування та організації щоденних справ користувача;
- підтримка можливості авторизації та реєстрації користувачів;
- можливість відображення календаря зустрічей;
- можливість відображення інформації про діяльність та справи користувача;
- підтримка можливості зберігання інформації про контакти користувача;

– підтримка можливості редагування налаштувань.

А.3.2 Вимоги до інтерфейсу програми

Інтерфейс програмного забезпечення для планування та організації щоденних справ повинен бути зручним для користувачів. Повинна бути забезпечена можливість роботи в візуальному режимі.

А.3.3 Вимоги до надійності

Програмне забезпечення для планування та організації щоденних справ повинно забезпечити надійне функціонування.

А.3.4 Умови експлуатації

Для експлуатації програмного забезпечення для планування та організації щоденних справ необхідна наявність персонального комп'ютера.

А.3.5 Вимоги до складу та параметрів технічний засобів

Для безперебійного функціонування програмного застосунку для планування та організації щоденних справ необхідно виконати вимоги як до апаратної, так і до програмної частини інфраструктури. Основна логіка та обробка користувацьких запитів можуть розгортатися на окремих серверах, що працюють у середовищі з достатнім резервом обчислювальних ресурсів. Ключову роль відіграють сучасні процесори, що підтримують частоти, достатні для швидкого опрацювання одночасних запитів, а обсяг оперативної пам'яті має покривати потреби багатопотокової обробки та кешування

активних сесій. Особливі вимоги висуваються до характеристик накопичувачів, оскільки високі швидкості введення-виведення та низька затримка доступу гарантують швидке збереження та відновлення великого обсягу інформації, що постійно оновлюється користувачами під час планування завдань і подій.

A.3.6 Вимоги до маркування і пакування

Програма для планування та організації щоденних справ може бути записана на будь-якому носії інформації.

На пакуванні повинна бути назва програми – «Програмне забезпечення для планування та організації щоденних справ».

A.4 Вхідні дані до роботи

Вхідними даними до програмного забезпечення для планування та організації щоденних справ є всі значення, які користувач зазначає у формах, зокрема, набір авторизаційних даних, інформація про задачі користувача, контакти та ін.

Вихідними даними програмного забезпечення для планування та організації щоденних справ є календарі, які групують події та завдання за відповідними датами й часом, сторінки для внесення правок до завдань, списки контактів та ін.

ДОДАТОК Б
Фрагмент тексту програми

```

namespace ApPlan.Controllers
{
    public class AcntController : Controller
    {
        public ActionResult Regstrn()
        {
            return View();
        }

        public ActionResult Verifc(Acnt Acnt)
        {
            DB bd = new BD();
            try
            {
                DataTabl tabl = new DataTabl();
                MySqlDataAdaptr adaptr = new MySqlDataAdaptr();
                MySqlCommand cmd = new MySqlCommand("SELECT *FROM korstv WHERE
emAdrs=@uEm", bd.getConnection());
                cmd.Paramtrs.Add("@uEm", MySqlBdType.VarChar).Value =
Acnt.emAdrs;
                adaptr.SelectCommand = cmd;
                adaptr.Fill(tabl);
                if (tabl.Rows.Count == 0)
                {
                    cmd = new MySqlCommand("INSERT INTO `korstv`
(`login`, `psswrđ`, `emAdrs`, `sttsVEP`, `pshStt`) VALUES (@ulog,
@upass,@uEm, @uVep, @uPush)", bd.getConnection());
                    cmd.Paramtrs.Add("@ulog", MySqlBdType.VarChar).Value =
Acnt.login;
                    cmd.Paramtrs.Add("@upass", MySqlBdType.VarChar).Value =
Acnt.psswrđ;
                    cmd.Paramtrs.Add("@uVep", MySqlBdType.VarChar).Value =
Acnt.sttsVEP;
                    cmd.Paramtrs.Add("@uPush", MySqlBdType.VarChar).Value =
Acnt.pshStt;
                    cmd.Paramtrs.Add("@uEm", MySqlBdType.VarChar).Value =
Acnt.emAdrs;

                    bd.openConnection();
                    if (cmd.ExecuteNonQuery() == 1)
                    {

                        tabl = new DataTabl();
                        adaptr = new MySqlDataAdaptr();
                        cmd = new MySqlCommand("SELECT *FROM korstv
WHERE emAdrs=@uEm ", bd.getConnection());
                        cmd.Paramtrs.Add("@uEm",
MySqlBdType.VarChar).Value = Acnt.emAdrs;

```

```

        adaptr.SelectCommand = cmd;
        adaptr.Fill(tabl);
        if (tabl.Rows.Count == 1)
        {

HttpContext.Rspns.Klgths["AcntId"].Value =
Convert.ToString(tabl.Rows[0]["AcntId"]);

        }
        bd.closeConnection();

        return RedirectToAction("Index", "Home");
    }

    else
    {
        bd.closeConnection();
        ViewData["Messg"] = "Проблеми з
сервером";

        return
View("~/Views/Acnt/Regstrn.cshtml");
    }
}
else
{
    ViewData["Messg"] = "Користувач існує";
    return View("~/Views/Acnt/Regstrn.cshtml");
}
}
catch
{
    ViewData["Messg"] = "Problems with server";
    return View("~/Views/Acnt/Regstrn.cshtml");
}
}

public ActionResult Authorization()
{

    return View();
}

public ActionResult LogIn(Acnt Acnt)
{
    BD bd = new BD();
    try
    {
        DataTabl tabl = new DataTabl();

```

```

        MySqlDataAdapter adaptr = new MySqlDataAdapter();
        MySqlCommand cmd = new MySqlCommand("SELECT *FROM korstv
WHERE login=@uLog AND psswr=@uPs", bd.getConnection());
        cmd.Parameters.Add("@uLog",
MySqlDbType.VarChar).Value = Acnt.login;
        cmd.Parameters.Add("@uPs",
MySqlDbType.VarChar).Value = Acnt.psswr;
        adaptr.SelectCommand = cmd;
        adaptr.Fill(tbl);
        if (tbl.Rows.Count == 1)
        {
            Rspns.Klghs["AcntId"].Value =
Convert.ToString(tbl.Rows[0]["AcntId"]);
            return RedirectToAction("Index", "Home");
        }
        else
        {
            ViewData["Messg"] = "Wrong login";
            return
View("~/Views/Acnt/Authorization.cshtml");
        }
    }
    catch
    {
        ViewData["Messg"] = "Problems with server";
        return View("~/Views/Acnt/Authorization.cshtml");
    }
}
}
}

```

```

namespace ApPlan.Controllers
{
    public class KontaktController : Controller
    {
        public ActionResult Kontacts()
        {
            HttpKlgh klgh = Request.Klghs.Get("AcntId");
            if (klgh != null) {
                int vep = 0;
                List<Kontacts> cntLst = new List<Kontacts>();
                BD bd = new BD();
                string TmpId = Request.Klghs["AcntId"].Value;
                DataTable tbl = new DataTable();
                MySqlDataAdapter adaptr = new MySqlDataAdapter();
            }
        }
    }
}

```

```

        MySqlCommand cmd = new MySqlCommand("SELECT *FROM
konktacts WHERE idAccC=@usId ORDER BY knstsrnm",
bd.getConnection());
        cmd.Paramtrs.Add("@usId",
MySQLBdType.VarChar).Value = Convert.ToInt16(TmpId);
        adaptr.SelectCommand = cmd;
        adaptr.Fill(tabl);
        foreach (DataRow r in tabl.Rows)
        {
            Kontacts tmCnt = new Kontacts();
            tmCnt.CntId = Convert.ToInt16(r["idCnt"]);
            tmCnt.cntNm = Convert.ToString(r["cntNm"]);
            tmCnt.knstsrnm =
Convert.ToString(r["knstsrnm"]);
            tmCnt.phNmr = Convert.ToString(r["phNmr"]);
            cntLst.Add(tmCnt);
        }

        bd.closeConnection();
        tabl = new DataTabl();
        cmd = new MySqlCommand("SELECT *FROM korstv WHERE
AcntId=@uIdU", bd.getConnection());
        cmd.Paramtrs.Add("@uIdU",
MySQLBdType.VarChar).Value =
Convert.ToInt16(Request.Klghs["AcntId"].Value);
        adaptr.SelectCommand = cmd;
        adaptr.Fill(tabl);
        foreach (DataRow r in tabl.Rows)
        {
            vep = Convert.ToInt32(r["sttsVEP"]);
        }
        ViewBag.VEP = vep;

        return View(cntLst);
    }
    else return View();
}

public ActionResult lstCnt()
{
    List<Kontacts> cntLst = new List<Kontacts>();
    BD bd = new BD();
    string TmpId = Request.Klghs["AcntId"].Value;
    DataTabl tabl = new DataTabl();
    MySqlDataAdapter adaptr = new MySqlDataAdapter();
    MySqlCommand cmd = new MySqlCommand("SELECT *FROM konktacts
WHERE idAccC=@usId ORDER BY knstsrnm", bd.getConnection());

```

```

        cmd.Paramtrs.Add("@usId", MySqlDbType.VarChar).Value
= Convert.ToInt16(TmpId);
        adaptr.SelectCommand = cmd;
        adaptr.Fill(tabl);
        foreach (DataRow r in tabl.Rows)
        {
            Kontacts tmCnt = new Kontacts();
            tmCnt.CntId = Convert.ToInt16(r["idCnt"]);
            tmCnt.cntNm = Convert.ToString(r["cntNm"]);
            tmCnt.knstsrm = Convert.ToString(r["knstsrm"]);
            tmCnt.phNmr = Convert.ToString(r["phNmr"]);
            cntLst.Add(tmCnt);
        }

        return PartialView( "lstCnt",cntLst);
    }

    public ActionRslt searchCnt(string searchRslt)
    {
        List<Kontacts> cntLst = new List<Kontacts>();
        List<Kontacts> rslt = new List<Kontacts>();
        BD bd = new BD();
        string TmpId = Request.Klghs["AcntId"].Value;
        DataTabl tabl = new DataTabl();
        MySqlDataAdapter adaptr = new MySqlDataAdapter();
        MySqlCommand cmd = new MySqlCommand("SELECT *FROM konktacts
WHERE idAccC=@usId ORDER BY knstsrm", bd.getConnection());
        cmd.Paramtrs.Add("@usId", MySqlDbType.VarChar).Value
= Convert.ToInt16(TmpId);
        adaptr.SelectCommand = cmd;
        adaptr.Fill(tabl);
        foreach (DataRow r in tabl.Rows)
        {
            Kontacts tmCnt = new Kontacts();
            tmCnt.CntId = Convert.ToInt16(r["idCnt"]);
            tmCnt.cntNm = Convert.ToString(r["cntNm"]);
            tmCnt.knstsrm = Convert.ToString(r["knstsrm"]);
            tmCnt.phNmr = Convert.ToString(r["phNmr"]);
            cntLst.Add(tmCnt);
        }

        foreach (Kontacts r in cntLst)
        {
            string s = r.cntNm + " " + r.knstsrm + " "+r.phNmr;

            if (s.Contains(searchRslt))
            {

```

```

        rslt.Add(r);
    }
    else if (Convert.ToString(r.knstsrm+ " "
+r.cntNm + " "+r.phNmr).Contains(searchRslt))
    { rslt.Add(r); }
    else if (Convert.ToString(r.knstsrm + " " +
r.phNmr+ " "+ r.cntNm ).Contains(searchRslt))
    { rslt.Add(r); }
    else if (Convert.ToString( r.phNmr+ " "
+r.knstsrm + " " + r.cntNm ).Contains(searchRslt))
    { rslt.Add(r); }
    else if (Convert.ToString(r.cntNm+" "+ r.phNmr+ "
" +r.knstsrm ).Contains(searchRslt))
    { rslt.Add(r); }
    else if (Convert.ToString(r.phNmr +" " + r.cntNm+
" " + r.knstsrm).Contains(searchRslt))
    { rslt.Add(r); }

    }
    return PartialView("lstCnt", rslt);
}

public ActionResult AddCont(Kontacts cont)
{
    BD bd = new BD();
    MySqlCommand cmd= new MySqlCommand("INSERT INTO `kontacts`
(`idAccC`,`idCnt`,`cntNm`,`knstsrm`,`phNmr`) VALUES
(@uUs,NULL, @uNam, @uSurn, @uPhoneNum)", bd.getConnection());
    cmd.Paramtrs.Add("@uUs", MySqlDbType.VarChar).Value =
Request.Klghs["AcntId"].Value;
    cmd.Paramtrs.Add("@uNam", MySqlDbType.VarChar).Value
= cont.cntNm;
    cmd.Paramtrs.Add("@uSurn", MySqlDbType.VarChar).Value
= cont.knstsrm;
    cmd.Paramtrs.Add("@uPhoneNum",
MySqlDbType.VarChar).Value = cont.phNmr;
    return RedirectToAction("Kontacts");
}

public JsonResult getKontakt(int CntId)
{
    List<Kontacts> cntLst = new List<Kontacts>();
    BD bd = new BD();
    DataTabl tabl = new DataTabl();
    MySqlDataAdapter adaptr = new MySqlDataAdapter();

```

```

        MySqlCommand cmd = new MySqlCommand("SELECT *FROM konktacts
WHERE idCnt=@uIdC ", bd.getConnection());
        cmd.Paramtrs.Add("@uIdC", MySqlBdType.VarChar).Value
= CntId;

        adaptr.SelectCommand = cmd;
        adaptr.Fill(tabl);
        foreach (DataRow r in tabl.Rows)
        {
            Kontacts tmCnt = new Kontacts();

            tmCnt.cntNm = Convert.ToString(r["cntNm"]);
            tmCnt.knstsrm = Convert.ToString(r["knstsrm"]);
            tmCnt.phNmr = Convert.ToString(r["phNmr"]);

            cntLst.Add(tmCnt);
        }
        bd.closeConnection();

        return Json(cntLst.AsEnumerable().Select(e => new {

            name = e.cntNm,
            prizv = e.knstsrm,
            phNmr = e.phNmr,
        })
        ).ToList(), JsonRequestBehavior.AllowGet);
    }

    public ActionRslt SaveKontakt(Kontacts e)
    {
        bool sttds = false;
        e.idAccC =
Convert.ToInt16(Request.Klghs["AcntId"].Value);

        if (e.CntId == 0)
        {
            BD bd = new BD();
            DataTabl tabl = new DataTabl();
            MySqlDataAdaptr adaptr = new MySqlDataAdaptr();
            MySqlCommand cmd = new MySqlCommand("INSERT INTO
`konktacts` (`idAccC`,`idCnt`,`cntNm`,`knstsrm`,`phNmr`) VALUES
(@uIdU,NULL,@uNam, @uSur, @uPhNum)", bd.getConnection());
            cmd.Paramtrs.Add("@uIdU",
MySqlBdType.VarChar).Value = e.idAccC;
            cmd.Paramtrs.Add("@uNam",
MySqlBdType.VarChar).Value = e.cntNm;
            cmd.Paramtrs.Add("@uSur",
MySqlBdType.VarChar).Value = e.knstsrm;

```

```

        cmd.Paramtrs.Add("@uPhNum",
MySQLBdType.VarChar).Value = e.phNmr;
        bd.openConnection();
        if (cmd.ExecuteNonQuery() == 1)
        {
            sttds = true;
        }
        bd.closeConnection();
    }
    else
    {
        BD bd = new BD();
        MySqlCommand cmd = new MySqlCommand("UPDATE `konktacts`
SET idAccC=@uIdU , cntNm=@uNam, knstsrnm=@uSur, phNmr=@uPhNum
WHERE idCnt=@uIdC ", bd.getConnection());
        cmd.Paramtrs.Add("@uIdU",
MySQLBdType.VarChar).Value = e.idAccC;
        cmd.Paramtrs.Add("@uIdC",
MySQLBdType.VarChar).Value = e.CntId;
        cmd.Paramtrs.Add("@uNam",
MySQLBdType.VarChar).Value = e.cntNm;
        cmd.Paramtrs.Add("@uSur",
MySQLBdType.VarChar).Value = e.knstsrnm;
        cmd.Paramtrs.Add("@uPhNum",
MySQLBdType.VarChar).Value = e.phNmr;

        bd.openConnection();
        if (cmd.ExecuteNonQuery() == 1)
        {
            bd.closeConnection();
            sttds = true;
        }
    }

    return new JsonRslt { Data = new { sttds = sttds } };
}

public JsonRslt DeleteKontakt( int CntId)
{
    bool sttds = false;
    int CntId =
Convert.ToInt16(Rspns.Klghs["idCntact"].Value);
    List<Kontacts> cntLst = new List<Kontacts>();
    BD bd = new BD();
    MySqlCommand cmd = new MySqlCommand("DELETE FROM konktacts
WHERE idCnt=@uIdC ", bd.getConnection());

```

```

        cmd.Paramtrs.Add("@uIdC", MySqlBdType.VarChar).Value
= CntId;
        bd.openConnection();
        if (cmd.ExecuteNonQuery() == 1)
        {
            bd.closeConnection();
            sttds = true;
        }
        else { bd.closeConnection(); }

        return new JsonRslt { Data = new { sttds = sttds } };
    }
}

```

```

namespace ApPlan.Controllers
{
    public class HomeController : Controller
    {

        public ActionRslt Index()
        {
            HttpKlgh klgh = Request.Klghs.Get("AcntId");
            if (klgh != null)
            {
                List<MTsk> cntLst = new List<MTsk>();
                BD bd = new BD();
                int vep = 0;
                DataTabl tabl = new DataTabl();
                MySqlDataAdaptr adaptr = new MySqlDataAdaptr();
                MySqlCmd cmd = new MySqlCmd("SELECT *FROM Tsk
WHERE idAccT=@uIdU ", bd.getConnection());
                cmd.Paramtrs.Add("@uIdU",
MySqlBdType.VarChar).Value =
Convert.ToInt16(Request.Klghs["AcntId"].Value);
                adaptr.SelectCommand = cmd;
                adaptr.Fill(tabl);
                foreach (DataRow r in tabl.Rows)
                {
                    MTsk tmCnt = new MTsk();
                    tmCnt.TskId = Convert.ToInt16(r["idTsk"]);
                    tmCnt.title = Convert.ToString(r["title"]);
                    tmCnt.strtDtT =
Convert.ToDateTime(Convert.ToString(r["strtDtT"]));
                    tmCnt.endDtT =
Convert.ToDateTime(Convert.ToString(r["endDtT"]));
                    tmCnt.descrT = Convert.ToString(r["descrT"]);
                }
            }
        }
    }
}

```

```

        cntLst.Add(tmCnt);
    }
    bd.closeConnection();
    tabl = new DataTabl();
    cmd = new MySqlCommand("SELECT *FROM korstv WHERE
AcntId=@uIdU", bd.getConnection());
    cmd.Paramtrs.Add("@uIdU",
MySQLBdType.VarChar).Value =
Convert.ToInt16(Request.Klghs["AcntId"].Value);
    adaptr.SelectCommand = cmd;
    adaptr.Fill(tabl);
    foreach (DataRow r in tabl.Rows)
    {
        vep = Convert.ToInt32(r["sttsVEP"]);
    }
    ViewBag.VEP = vep;
    return View(cntLst);
}
else return View();

}
public JsonResult getJSData()
{
    List<MTsk> cntLst = new List<MTsk>();
    BD bd = new BD();
    DataTabl tabl = new DataTabl();
    MySqlDataAdapter adaptr = new MySqlDataAdapter();
    MySqlCommand cmd = new MySqlCommand("SELECT *FROM Tsk WHERE
idAccT=@uIdU ", bd.getConnection());
    cmd.Paramtrs.Add("@uIdU", MySQLBdType.VarChar).Value
= Convert.ToInt16(Request.Klghs["AcntId"].Value);
    adaptr.SelectCommand = cmd;
    adaptr.Fill(tabl);
    foreach (DataRow r in tabl.Rows)
    {
        MTsk tmCnt = new MTsk();
        tmCnt.TskId = Convert.ToInt16(r["idTsk"]);
        tmCnt.title = Convert.ToString(r["title"]);
        tmCnt.strtDtT =
Convert.ToDateTime(Convert.ToString(r["strtDtT"]));

        tmCnt.endDtT =
Convert.ToDateTime(Convert.ToString(r["endDtT"]));
        tmCnt.descrT = Convert.ToString(r["descrT"]);
        cntLst.Add(tmCnt);
    }
    bd.closeConnection();
}

```

```

        return Json(cntLst.AsEnumerable().Select(e => new {
            id = e.TskId,
            start = e.strtDtT.ToString("yyyy-MM-dd HH:mm"),
            end = e.endDtT.ToString("yyyy-MM-dd HH:mm"),
            title = e.title,
            descrptn = e.descrT,

        }

        ).ToList(), JsonRequestBehavior.AllowGet);

    }

    public ActionRslt SaveEvent(MTsk e)
    {

        e.AccId =
        Convert.ToInt16(Request.Klghs["AcntId"].Value);
        if (e.TskId == 0)
        {
            BD bd = new BD();
            DataTabl tabl = new DataTabl();
            MySqlDataAdaptr adaptr = new MySqlDataAdaptr();
            MySqlCommand cmd = new MySqlCommand("INSERT INTO `Tsk`
            (`idAccT`,`idTsk`,`title`,`strtDtT`,`endDtT`,`descrT`) VALUES
            (@uIdU,NULL,@uTit, @uStaT, @uEndT,@uDes)", bd.getConnection());
            cmd.Paramtrs.Add("@uIdU",
            MySqlBdType.VarChar).Value =
            Convert.ToInt16(Request.Klghs["AcntId"].Value);
            cmd.Paramtrs.Add("@uTit",
            MySqlBdType.VarChar).Value = e.title;
            cmd.Paramtrs.Add("@uStaT",
            MySqlBdType.VarChar).Value = e.strtDtT.ToString("yyyy-MM-dd
            HH:mm:ss");
            cmd.Paramtrs.Add("@uEndT",
            MySqlBdType.VarChar).Value = e.endDtT.ToString("yyyy-MM-dd
            HH:mm:ss"); ;
            cmd.Paramtrs.Add("@uDes",
            MySqlBdType.VarChar).Value = e.descrT;
            bd.openConnection();
            if (cmd.ExecuteNonQuery() == 1)
            {
                tabl = new DataTabl();
                adaptr = new MySqlDataAdaptr();
                cmd = new MySqlCommand("SELECT *FROM Tsks WHERE
                idKorstv=@uIdU AND idTsk=@uNam AND title=@uTit AND descrptn=@uDes
                AND strtDtT= @uDate ", bd.getConnection());
                bd.closeConnection();
            }
        }
    }

```

```

    }
    bd.closeConnection();
    var sttds = true;
    return new JsonRslt { Data = new { sttds = sttds
} };

}
else
{
    BD bd = new BD();
    MySqlCommand cmd = new MySqlCommand("UPDATE `Tsk` SET
idAccT=@uIdU, title=@uTit , strtDtT=@uStaT,endDtT=@uEndT,
descrT=@uDes WHERE idTsk=@uIdT ", bd.getConnection());
    cmd.Paramtrs.Add("@uIdU",
MySQLBdType.VarChar).Value =
Convert.ToInt16(Request.Klghs["AcntId"].Value);
    cmd.Paramtrs.Add("@uIdT",
MySQLBdType.VarChar).Value = e.TskId;
    cmd.Paramtrs.Add("@uTit",
MySQLBdType.VarChar).Value = e.title;
    cmd.Paramtrs.Add("@uStaT",
MySQLBdType.VarChar).Value = e.strtDtT.ToString("yyyy-MM-dd
HH:mm:ss");
    cmd.Paramtrs.Add("@uEndT",
MySQLBdType.VarChar).Value = e.endDtT.ToString("yyyy-MM-dd
HH:mm:ss");
    cmd.Paramtrs.Add("@uDes",
MySQLBdType.VarChar).Value = Convert.ToString(e.descrT);

    bd.openConnection();
    if (cmd.ExecuteNonQuery() == 1)
    {
        bd.closeConnection();
    }

    var sttds = true;
    return new JsonRslt { Data = new { sttds = sttds
} };

}

}

public JsonRslt getCntLst()
{
    List<Kontacts> cntLst = new List<Kontacts>();
    BD bd = new BD();
    DataTabl tabl = new DataTabl();

```

```

        MySqlDataAdapter adaptr = new MySqlDataAdapter();
        MySqlCommand cmd = new MySqlCommand("SELECT *FROM konktacts
WHERE idAccC=@uIdU ", bd.getConnection());
        cmd.Parameters.Add("@uIdU", MySqlDbType.VarChar).Value
= Convert.ToInt16(Request.Klghs["AcntId"].Value);
        adaptr.SelectCommand = cmd;
        adaptr.Fill(tabl);
        foreach (DataRow r in tabl.Rows)
        {
            Kontacts tmCnt = new Kontacts();
            tmCnt.CntId = Convert.ToInt16(r["idCnt"]);
            tmCnt.cntNm = Convert.ToString(r["cntNm"]);
            tmCnt.knstsrm = Convert.ToString(r["knstsrm"]);
            tmCnt.phNmr = Convert.ToString(r["phNmr"]);
            cntLst.Add(tmCnt);
        }
        bd.closeConnection();

        return Json(cntLst.AsEnumerable().Select(e => new {
            CntId = e.CntId,
            prizv = Convert.ToString(e.knstsrm+" " +
e.cntNm+" " + e.phNmr)
        }).ToList(), JsonRequestBehavior.AllowGet);
    }

    public JsonResult getSelected( )
    {
        List<string> cntLst = new List<string>();
        BD bd = new BD();
        DataTabl tabl = new DataTabl();
        MySqlDataAdapter adaptr = new MySqlDataAdapter();
        MySqlCommand cmd = new MySqlCommand("SELECT
konktacts.idCnt,konktacts.cntNm,konktacts.knstsrm,konktacts.phNm
r FROM konktacts INNER JOIN contforTsk ON
konktacts.idCnt=contforTsk.idCntT WHERE
((contforTsk.idTskCont)=@uIdT) ", bd.getConnection());
        cmd.Parameters.Add("@uIdT", MySqlDbType.VarChar).Value
= Convert.ToInt16(Request.Klghs["TskId"].Value);
        adaptr.SelectCommand = cmd;
        adaptr.Fill(tabl);
        if (tabl.Rows.Count >0)
        { foreach(DataRow r in tabl.Rows)
            {
                string tmCnt = Convert.ToString(r["idCnt"]);
                cntLst.Add(tmCnt);
            }
        }
    }

```

```

    }
    return Json(cntLst.AsEnumerable().Select(e => new {
        CntId = e,
    })
        .ToList(), JsonRequestBehavior.AllowGet);
}

public ActionRslt SaveKontacts(KorstvS Cont)
{
    var sttds = false;
    BD bd = new BD();
    DataTabl tabl = new DataTabl();
    MySqlDataAdaptr adaptr = new MySqlDataAdaptr();
    int TskId;
    MySqlCmd cmd;
    if (Request.Klghs["TskId"].Value == "0")
    {
        cmd = new MySqlCmd("SELECT* FROM Tsk ORDER BY
idTsk DESC LIMIT 1", bd.getConnection());

        adaptr.SelectCmd = cmd;
        adaptr.Fill(tabl);
        TskId = Convert.ToInt16(tabl.Rows[0]["idTsk"]);
    }
    else
    {
        TskId = Convert.ToInt16(
Request.Klghs["TskId"].Value);
    }
    if (Cont.forCont != "-1")
    {

        List<int> CntId = new List<int>();
        string[] idCntacts = Cont.forCont.Split(new[] {
'/' }, StringSplitOptions.RemoveEmptyEntries);
        foreach (string r in idCntacts)
        {
            CntId.Add(Convert.ToInt32(r));
        }

        List<int> exst = new List<int>();
        List<int> add = new List<int>();
        List<Kontacts> cntLst = new List<Kontacts>();
        bd = new BD();
        tabl = new DataTabl();
        adaptr = new MySqlDataAdaptr();
    }
}

```

```

        cmd = new MySqlCommand("SELECT *FROM contforTsk WHERE
idTskCont=@uIdC ", bd.getConnection());
        cmd.Parameters.Add("@uIdC",
MySQLDbType.VarChar).Value = TskId;
        adaptr.SelectCommand = cmd;
        adaptr.Fill(tabl);
        if (tabl.Rows.Count > 0)
        {
            foreach (DataRow r in tabl.Rows)
            {
                exst.Add(Convert.ToInt16(r["idCntT"]));
            }
            foreach (int r in CntId)
            {
                if (!exst.Exists(item => item == r))
                {
                    add.Add(r);
                }
                else
                {
                    exst.Remove(r);
                }
            }
            foreach (int r in add)
            {
                bd = new BD();
                cmd = new MySqlCommand("INSERT INTO
`contforTsk` (`idLinkCT`,`idTskCont`,`idCntT`) VALUES( NULL,
@uidT, @uidC)", bd.getConnection());
                cmd.Parameters.Add("@uidT",
MySQLDbType.VarChar).Value = TskId;
                cmd.Parameters.Add("@uidC",
MySQLDbType.VarChar).Value = r;
                bd.openConnection();
                if (cmd.ExecuteNonQuery() == 1)
                {
                    sttds = true;
                    bd.closeConnection();
                }
                else { bd.closeConnection(); sttds =
false ; }
            }
            foreach (int r in exst)
            {
                bd = new BD();
                cmd = new MySqlCommand("DELETE FROM
contforTsk WHERE idTskCont=@uIdT AND idCntT=@uIdC",
bd.getConnection());

```

```

        cmd.Paramtrs.Add("@uIdT",
MySqlDbType.VarChar).Value = TskId;
        cmd.Paramtrs.Add("@uIdC",
MySqlDbType.VarChar).Value = r;
        bd.openConnection();
        if (cmd.ExecuteNonQuery() == 1)
        {
            sttds = true;
            bd.closeConnection();
        }
        else { bd.closeConnection(); sttds =
false ; }
    }

}
else
{
    foreach (int r in CntId)
    {
        bd = new BD();
        cmd = new MySqlCommand("INSERT INTO
`contforTsk` (`idLinkCT`,`idTskCont`,`idCntT`) VALUES( NULL,
@uIdT, @uIdC)", bd.getConnection());
        cmd.Paramtrs.Add("@uIdT",
MySqlDbType.VarChar).Value = TskId;
        cmd.Paramtrs.Add("@uIdC",
MySqlDbType.VarChar).Value = r;
        bd.openConnection();
        if (cmd.ExecuteNonQuery() == 1)
        {
            bd.closeConnection();
            sttds = true;
        }
        else { bd.closeConnection(); sttds =
false; }
    }
}

}
else
{
    bd = new BD();
    cmd = new MySqlCommand("DELETE FROM contforTsk WHERE
idTskCont=@uIdT", bd.getConnection());
    cmd.Paramtrs.Add("@uIdT",
MySqlDbType.VarChar).Value = TskId;
    bd.openConnection();
    if (cmd.ExecuteNonQuery() == 1)

```

```

        {
            bd.closeConnection();
            sttds = true;
        }
        else { bd.closeConnection(); }
    }

    return new JsonRslt { Data = new { sttds = sttds } };

}

public JsonRslt DeleteEvent(int TskId)
{
    var sttds = false;

    BD bd = new BD();
    MySqlCommand cmd = new MySqlCommand("DELETE FROM Tsk WHERE
idTsk=@uIdT", bd.getConnection());
    cmd.Paramtrs.Add("@uIdT", MySqlBdType.VarChar).Value
= TskId;
    bd.openConnection();
    if (cmd.ExecuteNonQuery() == 1)
    {
        bd.closeConnection();
        sttds = true;
    }
    else { bd.closeConnection(); }

    return new JsonRslt { Data = new { sttds = sttds } };
}
}
}

```

ДОДАТОК В
Слайди презентації

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

**Дослідження та програмна реалізація
застосування для планування та організації
щоденних справ**

Виконав:
студент групи КНТ-114м

Олег ГАЛИШ

Керівник:
д.т.н., професор кафедри ПЗ

Андрій ОЛШНИК

Рисунок В.1 – Слайд 1

Об'єкт, предмет та мета роботи

Об'єкт дослідження – процес розробки програмного забезпечення для планування та організації щоденних справ.

Предмет дослідження – програмні засоби для планування та організації щоденних справ.

Мета роботи – дослідження та розробка програмного застосування для планування та організації щоденних справ.

Рисунок В.2 – Слайд 2

Завдання роботи

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

- виконати аналіз предметної області та програмних засобів для планування та організації щоденних справ;
- розробити модель взаємодії користувача з програмним забезпеченням для планування та організації щоденних справ;
- здійснити проектування програмного забезпечення для планування та організації щоденних справ;
- створити програмне забезпечення для планування та організації щоденних справ;
- виконати тестування розробленого програмного забезпечення для планування та організації щоденних справ.

3

Рисунок В.3 – Слайд 3

Порівняння ПЗ для планування та організації щоденних справ

Критерій порівняння	Twobird	Devonthink	Sunsama
Зручність планування завдань	+	+–	+
Інтеграція з поштою	+	–	+–
підтримка командної роботи	+–	+	+–
Наявність безкоштовної версії	–	+–	–
Простота інтерфейсу	+–	+–	–
Засоби збереження та редагування документів	+–	+	+–

4

Рисунок В.4 – Слайд 4

Порівняння мов програмування

Критерій порівняння мов програмування	Мова програмування		
	C#	Python	Java
Розвиненість інструментів розробки	+	+-	+
Можливості для кросплатформенності	+	+-	+
Зручність створення графічного інтерфейсу	+	+-	+-
Зручність для створення ПЗ для планування та організації щоденних справ	+	-	-
Продуктивність виконання коду	+	+-	+-

5

Рисунок В.5 – Слайд 5

Порівняння середовищ розробки

Критерій порівняння середовищ розробки	Середовища розробки		
	Visual Studio	Rider	Eclipse
Інтеграція з мовою C#	+	+-	+-
Наявність вбудованих засобів відлагодження	+	+-	+
Підтримка складної архітектури рішень	+	+-	+-
Зручність засобів для створення графічного інтерфейсу	+	-	+-
Підтримка кросплатформенності	+	+-	+-

6

Рисунок В.6 – Слайд 6

Модель взаємодії користувача з ПЗ

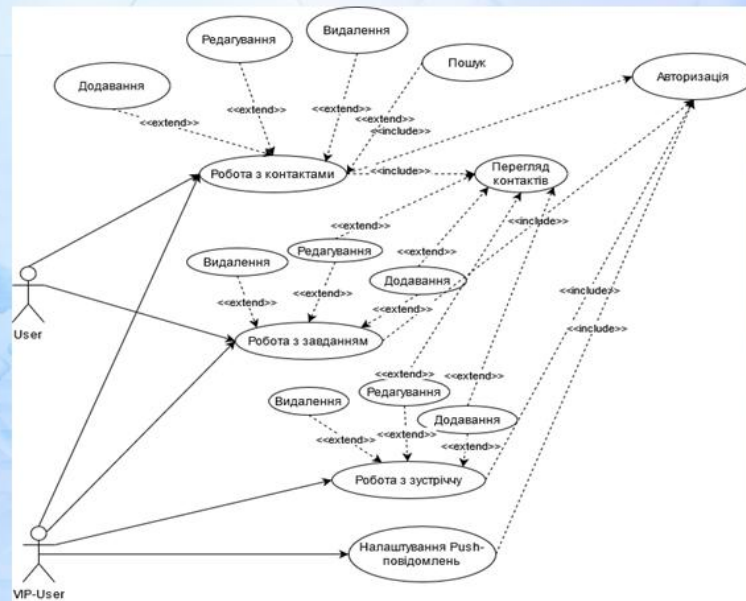


Рисунок В.7 – Слайд 7

Структура ПЗ у вигляді діаграми класів

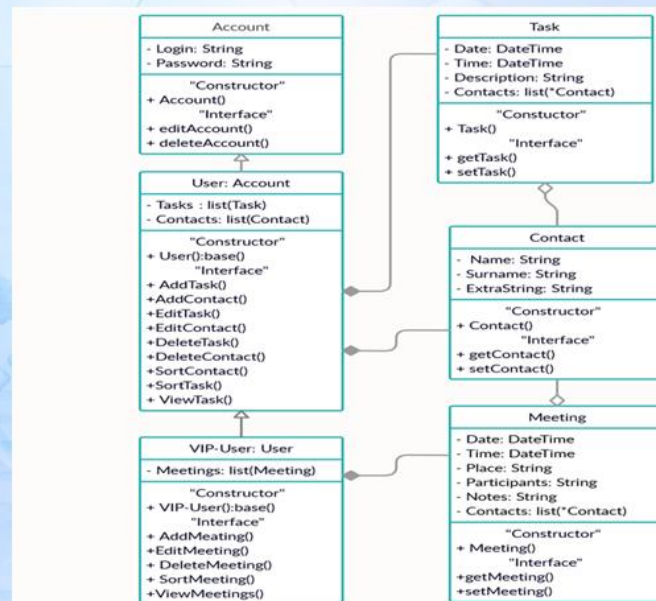
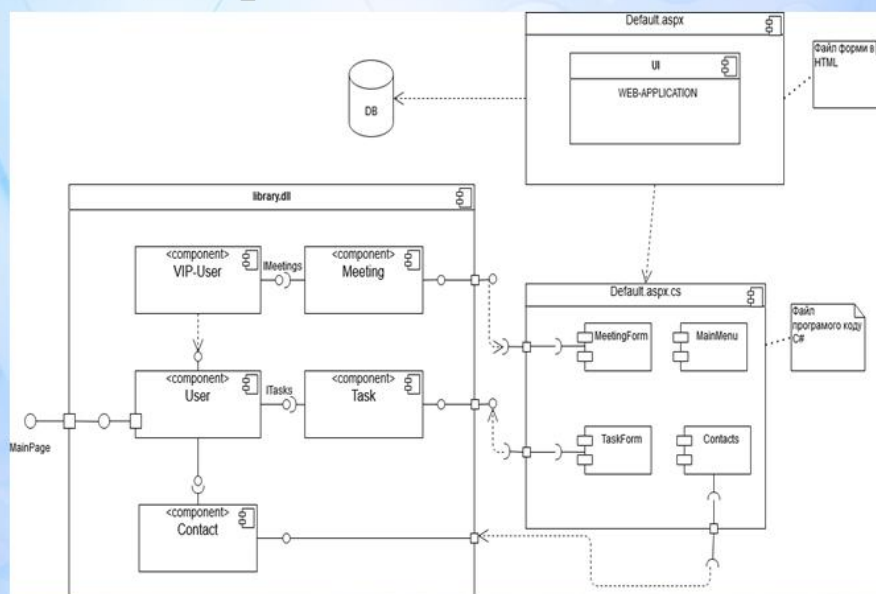


Рисунок В.8 – Слайд 8

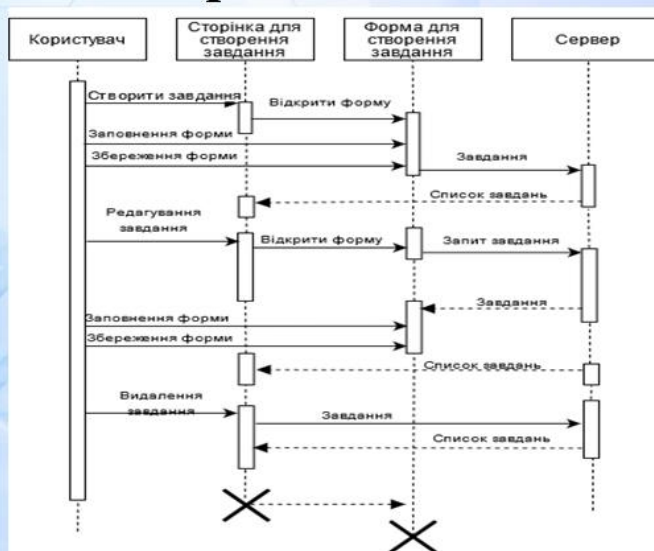
Діаграма компонентів



9

Рисунок В.9 – Слайд 9

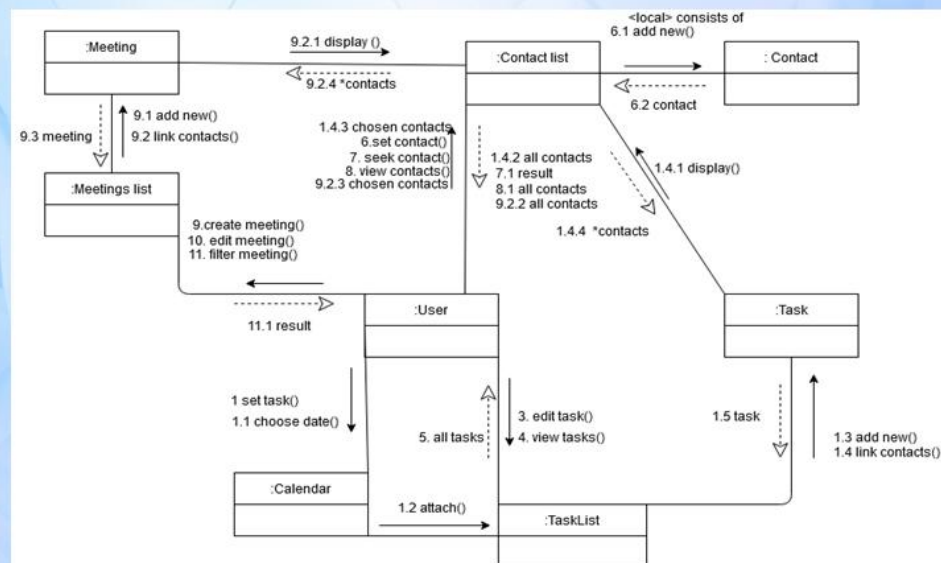
Схема функціонування застосунку у вигляді діаграми послідовностей



10

Рисунок В.10 – Слайд 10

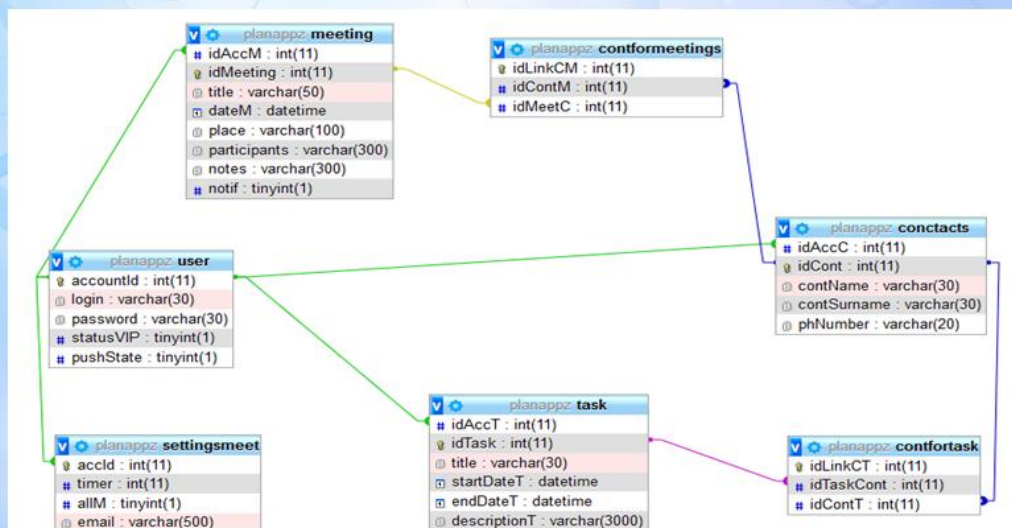
Діаграма кооперації



11

Рисунок В.11 – Слайд 11

Схема бази даних ПЗ для планування та організації щоденних справ



12

Рисунок В.12 – Слайд 12

Робота з програмою. Авторизація та реєстрація. Календар подій

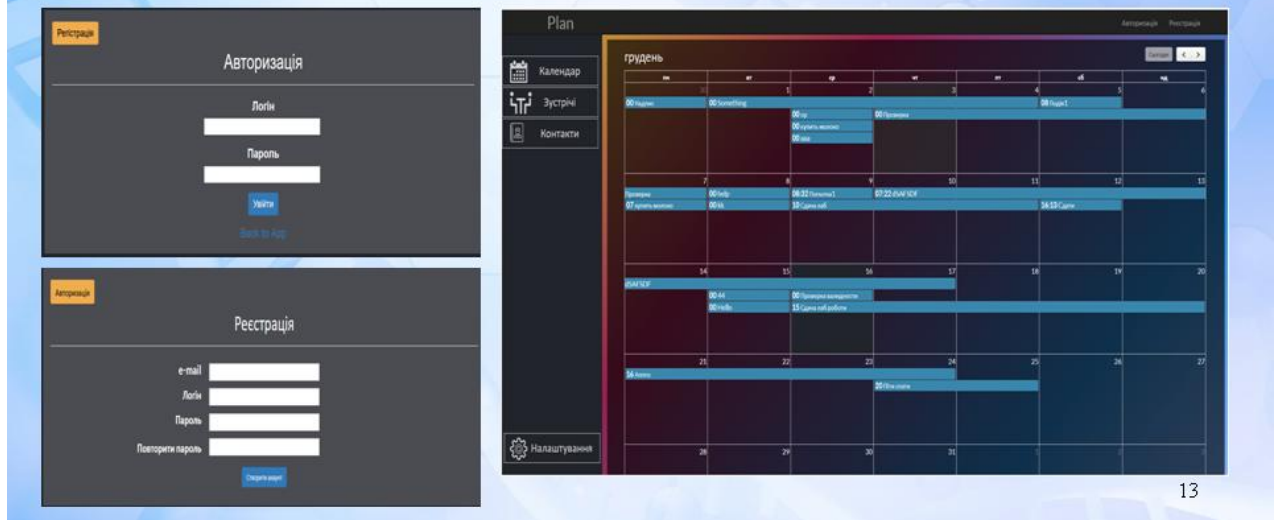


Рисунок В.13 – Слайд 13

Робота з програмою. Зустрічі та контакти

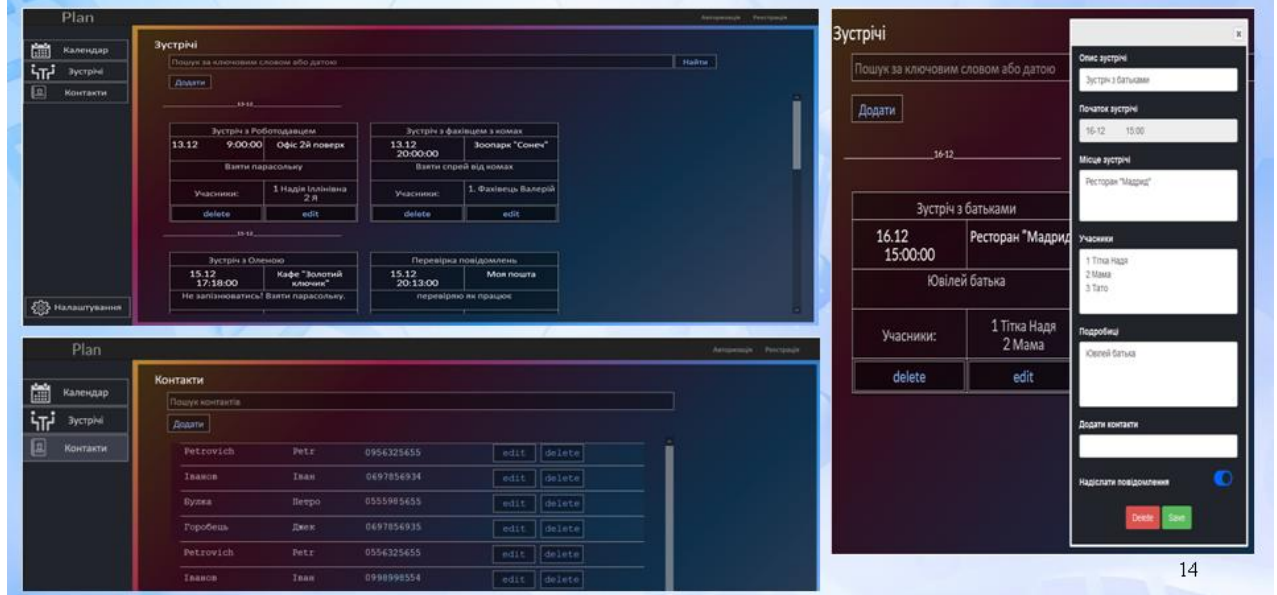


Рисунок В.14 – Слайд 14

Висновки

В ході виконання дипломної кваліфікаційної роботи магістра було проаналізовано та досліджено процес розробки програмного забезпечення для планування та організації щоденних справ.

Новизна роботи полягає у тому, що створено модель взаємодії користувача з програмним забезпеченням для планування та організації щоденних справ, яка подана у вигляді діаграми прецедентів, та дозволяє розроблювати програмне забезпечення для планування та організації щоденних справ.

Практична цінність роботи полягає у тому, що створено програмне забезпечення для планування та організації щоденних справ.

Виконано тестування розробленого програмного забезпечення для планування та організації щоденних справ. Після проведення тестування встановлено, що програма для планування та організації щоденних справ відповідає технічним вимогам і може успішно використовуватися за призначенням.

15

Рисунок В.15 – Слайд 15