

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет комп'ютерних наук і технологій

(повне найменування факультету)

Кафедра програмних засобів

(повне найменування кафедри)

Пояснювальна записка

до дипломного проєкту (роботи)

бакалавр

(ступінь вищої освіти)

на тему

ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ

ДЛЯ УПРАВЛІННЯ ЗАДАЧАМИ КОРИСТУВАЧА

SOFTWARE FOR USER

TASK MANAGEMENT

Виконав(ла): студент(ка) 4 курсу, групи КНТ-113сп

Спеціальності 121 Інженерія програмного

(код і найменування спеціальності)

забезпечення

Освітня програма (спеціалізація)

Інженерія програмного забезпечення

НОСКОВА А.О.

(ПРИЗВИЩЕ та ініціали)

Керівник КОЦУР М.І.

(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 бакалавр
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
“ ” 2026 року

З А В Д А Н Н Я

НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА(КИ)

НОСКОВОЇ Анастасії Олексіївни

(ПРІЗВИЩЕ, ім'я, по батькові)

1. Тема проєкту (роботи) Програмне забезпечення для управління задачами користувача. Software for User Task Management

керівник проєкту (роботи) к.т.н., доцент, КОЦУР Михайло Ігорович,
(науковий ступінь, вчене звання, ПРІЗВИЩЕ, ім'я, по батькові)

затверджені наказом закладу вищої освіти від “7” квітня 2026 року № 139

2. Строк подання студентом проєкту (роботи) 03 червня 2026 року

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

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

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

6. Консультанти розділів проєкту (роботи)

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

7. Дата видачі завдання “7” квітня 2026 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.	2 тиждень	Розділ 3
5	Розробка програми.	3-4 тижні	Розділи 3, 4
6	Тестування та експериментальне дослідження програмного забезпечення.	5 тиждень	Розділ 4
7	Оформлення пояснювальної записки та документів до неї.	6 тиждень	Додатки
8	Нормоконтроль та рецензування.	7 тиждень	
9	Захист роботи.	8 тиждень	

Студент(ка)

(підпис) Анастасія НОСКОВА
(Ім'я ПРИЗВИЩЕ)

Керівник проєкту (роботи)

(підпис) Михайло КОЦУР
(Ім'я ПРИЗВИЩЕ)

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи бакалавра:
86 с., 3 табл., 29 рис., 3 дод., 18 джерел.

JAVASCRIPT, NODE.JS, ЗАДАЧА, КОРИСТУВАЧ, МОВА
ПРОГРАМУВАННЯ, ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ.

Об'єкт дослідження – процес розробки програмного забезпечення для управління задачами користувача.

Предмет дослідження – програмні засоби для управління задачами користувача.

Мета роботи – розробка програмного забезпечення для управління задачами користувача, що забезпечує зменшення часу на планування та виконання завдань.

Матеріали, методи та технічні засоби: мова програмування JavaScript, серверне середовище виконання Node.js.

Результати. Розроблено проєктні рішення для створення програмного забезпечення для управління задачами користувача. Створено програмне забезпечення для управління задачами користувача. Проведено тестування розробленої програмної системи для управління задачами користувача.

Висновки. Мету роботи досягнуто. Розроблено програмне забезпечення для управління задачами користувача за допомогою мови програмування JavaScript та серверного середовища виконання Node.js.

Галузь використання – управління робочим часом.

ABSTRACT

Explanatory note to the diploma qualifying work of the bachelor: 86 pages, 3 tables, 29 figures, 3 appendixes, 18 sources.

JAVASCRIPT, NODE.JS, TASK, USER, PROGRAMMING LANGUAGE, SOFTWARE.

The object of research is the process of developing software managing user tasks.

The subject of the research is software for managing user tasks.

The purpose of this work is to develop the software for managing user tasks, which provides a reduction in the time for planning and executing tasks.

Materials, methods and technical tools: JavaScript programming language, Node.js server runtime environment.

Results. Project solutions for the creation of software for managing user tasks has been designed. The software for managing user tasks has been developed. The developed software system for managing user tasks has been tested.

Conclusions. The goal of the work has been achieved. The software tools for managing user tasks using the JavaScript programming language and the Node.js server runtime environment have been developed.

Scope of use – working time management.

ЗМІСТ

	С.
Перелік скорочень та умовних позначок	8
Вступ.....	9
1 Аналіз предметної області.....	11
1.1 Програмне забезпечення для управління задачами користувача	11
1.2 Аналіз програмного забезпечення для управління задачами користувача	20
1.3 Висновки за розділом 1	29
2 Матеріали і методи	31
2.1 Вибір мови програмування	31
2.2 Вибір серверного середовища виконання для програмного забезпечення для управління задачами користувача.....	34
2.3 Висновки за розділом 2	38
3 Опис програми	39
3.1 Структура програмного забезпечення для управління задачами користувача	39
3.2 Функціонування програмного забезпечення для управління задачами користувача	42
3.3 Розробка бази даних програмного забезпечення для управління задачами користувача	46
3.4 Проєктування інтерфейсу програмного забезпечення для управління задачами користувача.....	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 управління задачами користувача	54
4.3 Інструкція по експлуатації програми.....	55
4.3.1 Звернення до програми.....	55
4.3.2 Вхідні й вихідні дані	56

4.3.3 Повідомлення.....	56
4.4 Виконання програмного забезпечення для управління задачами користувача	56
4.5 Тестування програмного забезпечення для управління задачами користувача	59
4.6 Висновки за розділом 4	60
Висновки	61
Перелік джерел посилання	64
Додаток А Технічне завдання	66
Додаток Б Фрагмент тексту програми	71
Додаток В Слайди презентації	80

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

CRM – Customer Relationship Management;

ERP – Enterprise Resource Planning;

БД – база даних;

ПЗ – програмне забезпечення;

ШІ – штучний інтелект.

ВСТУП

Актуальність програмного забезпечення для управління задачами користувача зумовлюється сучасними тенденціями розвитку цифрового суспільства, зростанням обсягів інформації та ускладненням професійної й навчальної діяльності. Умови постійної багатозадачності, стислих термінів виконання робіт і необхідності оперативного прийняття рішень вимагають застосування ефективних інструментів організації та контролю діяльності. У зв'язку з цим використання спеціалізованого програмного забезпечення розглядається як один із ключових чинників підвищення ефективності управління індивідуальними та колективними процесами виконання завдань [1], [2].

Використання програмних засобів для управління задачами забезпечує впорядкування інформації про поточні та заплановані дії, що сприяє раціональному розподілу часу і ресурсів. Завдяки можливості фіксації завдань, визначення термінів їх виконання та встановлення пріоритетів досягається зниження ймовірності пропуску важливих етапів роботи. Це особливо актуально в умовах інтенсивної діяльності, коли традиційні способи планування не забезпечують належного рівня контролю [2]-[4].

Актуальність програмного забезпечення для управління задачами також посилюється розвитком дистанційних форм роботи та навчання. У таких умовах виникає потреба в централізованому середовищі для координації дій, синхронізації процесів і забезпечення прозорості виконання завдань. Використання відповідних інформаційних систем сприяє підвищенню рівня відповідальності та узгодженості дій між учасниками спільної діяльності [5].

Таким чином, програмне забезпечення для управління задачами користувача є важливим інструментом сучасної інформаційної інфраструктури. Його актуальність визначається здатністю підвищувати ефективність планування, оптимізувати використання ресурсів і забезпечувати контроль за виконанням завдань у динамічних умовах сучасної діяльності [6].

Незважаючи на значні переваги, використання таких програмних систем не завжди забезпечує очікуваний результат та в окремих випадках створює додаткові ускладнення в процесі діяльності користувача. Одним із суттєвих недоліків є складність освоєння окремих програмних рішень. Надмірна функціональність, складний інтерфейс або недостатньо зрозуміла логіка роботи можуть ускладнювати процес адаптації користувачів, особливо за відсутності спеціальної підготовки. У таких випадках час, витрачений на навчання та налаштування системи, може перевищувати вигоди від її використання. Ще одним недоліком є ризик інформаційного перевантаження. Постійні сповіщення, нагадування та оновлення статусів здатні знижувати концентрацію уваги та спричиняти фрагментацію робочого процесу. Замість підвищення продуктивності це може призводити до зростання рівня стресу та відчуття постійної зайнятості без реального прогресу у виконанні завдань. Крім того, питання захисту персональних даних і конфіденційності інформації залишаються актуальними, особливо при використанні хмарних сервісів.

Тому актуальною є розробка програмного забезпечення для управління задачами користувача. У дипломній кваліфікаційній роботі бакалавра розв'язується актуальне завдання розробки програмного забезпечення для управління задачами користувача.

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

- виконати аналіз предметної області та програмних засобів для управління задачами користувача;
- здійснити проєктування програмного забезпечення для управління задачами користувача;
- створити програмне забезпечення для управління задачами користувача;
- виконати тестування розробленого програмного забезпечення для управління задачами користувача.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Програмне забезпечення для управління задачами користувача

Програмне забезпечення для управління задачами користувача представляє собою комплекс цифрових інструментів, призначених для організації, контролю та оптимізації процесів виконання завдань у рамках індивідуальної чи колективної діяльності. Такі програмні засоби забезпечують централізоване відстеження робочих процесів, дозволяють структурувати інформацію про завдання, терміни їх виконання та відповідальних осіб, а також надають можливість координувати діяльність між різними учасниками проєктів. Застосування подібного програмного забезпечення сприяє підвищенню продуктивності, оскільки користувач отримує змогу планувати завдання, контролювати прогрес і своєчасно реагувати на зміни у робочих процесах [1].

До особливостей таких систем належить інтеграція засобів комунікації та обміну інформацією в межах одного середовища, що зменшує потребу у сторонніх платформах та спрощує обмін даними між членами команди. Програмне забезпечення дозволяє автоматизувати повторювані операції, що знижує рутинне навантаження і сприяє концентрації на більш стратегічних чи творчих аспектах діяльності. Завдяки візуалізації завдань через різноманітні представлення, наприклад, дошки в стилі Канбан або діаграми Ганта, користувачі мають змогу легко оцінювати стан проєктів, виявляти вузькі місця та ефективно розподіляти ресурси [1]-[4].

Переваги використання таких програмних систем проявляються у зручності планування та моніторингу робочих процесів, підвищенні прозорості та підзвітності виконання завдань, а також у можливості отримувати аналітичні дані щодо продуктивності. Програмні продукти для управління задачами здатні адаптуватися під різні моделі роботи, підтримуючи як індивідуальне управління завданнями, так і координацію

роботи команд різного масштабу. Вони забезпечують узгодженість дій, скорочують втрати часу на комунікацію та дозволяють ефективніше досягати поставлених цілей, створюючи комплексне середовище для організованої та структурованої діяльності [1].

Програмне забезпечення для управління завданнями користувача використовується для організації, контролю та впорядкування діяльності, пов'язаної з виконанням завдань у межах проєктів або робочих процесів. Такі системи дозволяють структурувати завдання, встановлювати пріоритети, слідкувати за термінами виконання та контролювати прогрес, забезпечуючи при цьому централізований доступ до необхідної інформації. Хоч управління завданнями та управління проєктами мають спільні риси, перше зосереджене на окремих завданнях, надаючи можливість приділяти увагу конкретним діям, відповідальним особам та часовим рамкам, що спрощує організацію робочого процесу [1].

Використання програмного забезпечення для управління завданнями підвищує контроль над інформацією та знижує ризик її втрати, оскільки всі дані про завдання, комунікацію та документи збираються в одному середовищі. Така система полегшує делегування та сприяє більш ефективній взаємодії членів команди, оскільки призначення завдання автоматично повідомляє відповідального, а можливість залишати коментарі та оновлення дозволяє координувати дії віддалено. Крім того, визначення пріоритетів допомагає співробітникам раціонально розподіляти свій час і виконувати завдання у правильному порядку, що значно підвищує продуктивність. Можливість розбивати великі завдання на менші частини дозволяє зосередитися на конкретних кроках і уникати перевантаження [1].

Програмні системи управління завданнями мають різноманітні функціональні можливості. Планування та складання графіків дозволяють організувати робочий процес у вигляді окремих завдань із встановленими термінами та можливістю повторення. Для командної роботи передбачені інструменти співпраці, які полегшують обмін інформацією, відстеження

виконання та взаємодію з керівництвом без потреби в постійних зустрічах. Автоматизація рутинних процесів дозволяє значно скоротити ручну роботу, забезпечуючи більш ефективне управління складними завданнями та підвищуючи загальну продуктивність. Представлення завдань у різних форматах, таких як календар, дошки Канбан або діаграми Ганта, сприяє візуальному контролю над прогресом і спрощує делегування завдань на різних етапах. Відстеження часу дозволяє визначати, скільки ресурсів витрачається на виконання кожного завдання, виявляти вузькі місця та оптимізувати робочі процеси, а також забезпечує облік робочого часу для погодинної оплати або контролю витрат у межах проєктів. У підсумку, таке програмне забезпечення створює ефективне середовище для організованого, прозорого та структурованого виконання завдань, сприяючи підвищенню продуктивності та контролю над робочими процесами [1].

Управління завданнями здійснюється за допомогою програмних засобів та певного структурованого процесу, що забезпечує послідовне виконання робочих дій та контроль над прогресом. Спершу створюються завдання, у які додаються всі необхідні деталі для виконання, визначаються терміни та відповідальні особи, після чого завдання призначається конкретним виконавцям. Наприклад, при адаптації нового співробітника завдання можуть включати підготовку документації, надання інформації про ресурси та координацію робочого місця з відділом персоналу. Для зручності деякі системи дозволяють імпортувати вже існуючі завдання з електронних таблиць та зберігати їх як шаблони для подальшого використання, що усуває необхідність створювати їх повторно [1], [2].

Подальший етап передбачає взаємодію та співпрацю учасників процесу. Виконавець отримує можливість уточнити деталі завдання або повідомити про можливі затримки, використовуючи вбудовані коментарі та інструменти обміну повідомленнями. Такий підхід дозволяє зменшити потребу у сторонніх каналах комунікації та забезпечує прозорість робочого процесу [1], [2].

Контроль виконання завдань здійснюється через оновлення статусу роботи у системі управління. Кожен учасник може відмічати завершені пункти завдань, додавати коментарі щодо труднощів і звертатися за допомогою до керівника, що забезпечує постійний моніторинг прогресу. Наприклад, при підготовці до орієнтаційної сесії для нових співробітників учасники команди можуть виконувати різні завдання одночасно, відмічаючи завершені кроки та повідомляючи керівника про необхідність підтримки, що дозволяє координувати дії та своєчасно реагувати на зміни в процесі виконання [1], [2].

Програмне забезпечення менеджера завдань призначений для організації щоденної діяльності компанії, забезпечуючи чітку структуру робочих процесів та своєчасне виконання завдань у команді. Його основне значення полягає у забезпеченні результативності, координації колективної роботи та контролю над строками виконання. Завдяки системі управління завданнями кожне завдання розташовується у відповідному місці з визначеною логікою та відповідальною особою, що дозволяє уникнути хаотичного ведення інформації через електронну пошту чи таблиці. Інструменти менеджера завдань надають можливість контролювати стан проектів, відстежувати критичні завдання та виявляти потенційні затримки, а також підтримують дисципліну виконання завдань завдяки суворому контролю термінів і обов'язків. Делегування завдань відбувається швидко, дозволяючи перепризначати їх окремим учасникам або розподіляти між кількома співробітниками. Зберігання всієї інформації в єдиному середовищі підвищує прозорість комунікацій між різними відділами, а чітке визначення обов'язків та термінів виконання зміцнює відповідальність кожного члена команди. Структуровані бізнес-процеси та інструменти для візуалізації та аналітики дають керівництву можливість відслідковувати прогрес команди та оцінювати наближення до досягнення поставлених цілей [2], [3].

Сучасні програмні системи управління завданнями забезпечують широкий функціонал, який адаптується під потреби різних підрозділів та типів команд, від IT і маркетингу до HR і виробництва. Вони охоплюють

планування завдань із можливістю встановлення термінів, пріоритетів і призначення відповідальних, а також створення підзадач для поетапного досягнення результатів. Управління пріоритетами виконується гнучко, з можливістю змін у режимі реального часу залежно від прогресу проекту або зовнішніх обставин. Для підтримки командної взаємодії передбачено призначення ролей, обмін коментарями, проведення обговорень, додавання файлів та інтеграцію з комунікаційними платформами. Автоматичні сповіщення та нагадування допомагають стежити за наближенням термінів та змінами у завданнях, а історія змін дозволяє відстежувати, хто і коли вносив корективи, забезпечуючи прозорість виконання. Крім того, інтеграція з CRM, календарями, електронною поштою та ERP-системами сприяє безперебійному обміну даними та ефективній роботі у єдиній цифровій екосистемі [2], [3].

Програмне забезпечення для управління задачами користувача виступає незамінним інструментом для будь-якого сучасного підрозділу або бізнес-напрямку, незалежно від сфери діяльності, будь то торгівля, виробництво, інформаційні технології чи надання послуг. Його застосування забезпечує підвищення продуктивності, покращення координації між відділами та своєчасне виконання завдань у межах встановлених термінів. Для керівництва це означає можливість отримувати оперативну інформацію про прогрес проектів, своєчасно реагувати на виниклі проблеми та розподіляти ресурси відповідно до пріоритетів. Для комерційних підрозділів менеджер завдань спрощує роботу з клієнтами, забезпечує контроль за виконанням повторюваних завдань та нагадує про ключові етапи роботи. У маркетингових відділах він дозволяє систематизувати минулі кампанії, координувати нові проекти та оцінювати ефективність проведених заходів. Для HR-спеціалістів система гарантує прозорість адаптації нових співробітників, контроль виконання завдань та своєчасне надходження нагадувань щодо оцінок, опитувань та навчальних заходів. IT-відділи отримують чітку структуру управління інцидентами та запитами на підтримку, де встановлюються пріоритети та відстежується стан кожного завдання [3], [4].

Незалежно від специфіки роботи відділу, усі вони стикаються зі схожими викликами, такими як суворі терміни, необхідність ефективної комунікації, планування та організація завдань. Саме тому менеджер завдань стає універсальним рішенням, здатним адаптуватися до будь-яких робочих процесів. У виробничих процесах він дозволяє координувати цикли роботи, контролювати дотримання стандартів і оперативно реагувати на непередбачені ситуації. У торгівлі система спрощує управління асортиментом, підтримує координацію маркетингових заходів та забезпечує швидке переналаштування завдань у відповідь на зміну попиту. У фінансових підрозділах менеджер завдань сприяє плануванню та контролю операційних процесів, забезпечує прозорість і підтримує взаємодію з іншими відділами. Для ІТ та команд розробки він забезпечує надійне відстеження проєктів, розподіл пріоритетів та ефективну комунікацію між внутрішніми групами та іншими підрозділами, що дозволяє оперативно обробляти запити, оновлювати функції та контролювати доступ. У сфері послуг система підтримує планування графіків, контроль якості завдань, швидке реагування на запити клієнтів та підвищення рівня їх задоволеності. Якщо менеджер завдань оснащений можливістю гнучкого налаштування та широкими інтеграційними функціями з іншими бізнес-системами, такими як CRM, ERP, поштові сервіси або месенджери, він стає універсальним засобом, що забезпечує ефективність, координацію та прозорість робочих процесів у будь-якому бізнесі та команді [3], [4].

Сучасний ринок рішень для програмного управління завданнями надзвичайно різноманітний, і інструмент, який підходить для невеликої команди, може виявитися непрактичним для великої організації з численними відділами. При виборі системи для управління завданнями варто враховувати не лише кількість користувачів, а й складність процесів, потребу в аналітиці, можливості регулювання доступу та ролей, а також варіанти інтеграції з іншими платформами [3], [4].

Існують програмні інструменти, орієнтовані на особисте планування, які дозволяють контролювати стан завдань від моменту створення до

завершення. Вони відрізняються простотою інтерфейсу, мінімальною конфігурацією та швидкістю додавання завдань, що робить їх зручними для фрілансерів та невеликих команд. Для середніх та великих компаній краще підходять корпоративні системи, що пропонують гнучкі налаштування робочих груп, контроль доступу на основі ролей та інтеграцію з CRM, ERP, календарями та іншими платформами. Проектно-орієнтовані рішення акцентують увагу на довгострокових проектах, відстеженні залежностей завдань, етапів і прогресу через візуалізацію у вигляді діаграм Ганта та часових шкал. Галузево-специфічні системи створені для унікальних потреб певних сфер діяльності, таких як ІТ, виробництво або роздріб, і часто постачаються з готовими робочими процесами та шаблонами. Також популярні менеджери завдань, інтегровані в універсальні платформи, де управління завданнями поєднане з документами, проектами та запитами на обслуговування, що дозволяє централізувати операції та скоротити час реагування [3], [4].

Вибираючи систему для управління завданнями, важливо оцінити масштабованість платформи та здатність адаптуватися до зростання кількості завдань і користувачів, зручність інтерфейсу та легкість освоєння командою, а також можливість гнучко керувати ролями та рівнями доступу. Важливим критерієм є інтеграція з існуючими бізнес-системами, включно з CRM, ERP, поштовими сервісами та месенджерами. Аналітика та звітність дозволяють відстежувати прогрес, аналізувати робоче навантаження та приймати обґрунтовані рішення на основі даних. Впровадження правильно обраного програмного інструмента підвищує прозорість процесів, покращує організацію роботи та сприяє досягненню бізнес-результатів [4], [5].

Управління завданнями може здійснюватися безперервно, проте просте виконання робіт і просування проектів не завжди гарантує ефективність бізнесу. Для досягнення реальних результатів потрібна систематична аналітика, яка дозволяє бачити не лише факт виконання, а й якість процесу та продуктивність команди. Інструменти аналітики в сучасних системах

диспетчерів завдань надають можливість відстежувати прогрес проектів у режимі реального часу, визначати стан кожного завдання та виявляти затримки або ризики у виконанні термінів. Крім того, вони показують навантаження на співробітників і дозволяють оптимально розподіляти завдання, забезпечуючи збалансованість роботи всередині команди [4], [5].

Аналітичні модулі дають змогу оцінювати виконання ключових показників ефективності, визначати, наскільки близькі команди до досягнення ключових цілей, і порівнювати результати за різні періоди. Візуалізація даних дозволяє легко сприймати навіть складні статистичні показники, відстежувати відхилення, тенденції та ефективність роботи за термінами й ролями. На основі цих даних можна прогнозувати завантаження, планувати ресурси та приймати обґрунтовані управлінські рішення, підвищуючи прозорість процесів на всіх рівнях [4], [5].

Використання штучного інтелекту (ШІ) значно розширює функціональні можливості систем диспетчера завдань. Інтелектуальні помічники та чат-боти автоматизують комунікацію, нагадують про терміни та швидко знаходять необхідну інформацію. ШІ може аналізувати робоче навантаження та компетенції співробітників для оптимального розподілу завдань, мінімізуючи ризик перевантаження або простоїв. Крім того, він здатен передбачати потенційні проблеми, виявляти вузькі місця в процесі та допомагати коригувати пріоритети завдань у динамічних умовах. Аналітика продуктивності, що підтримується ШІ, дозволяє отримати детальне уявлення про роботу команди та окремих співробітників, виявляти ефективні та слабкі сторони процесу та пропонувати шляхи для підвищення результативності. У комплексі це робить управління завданнями більш точним, швидким і прозорим, забезпечуючи основу для прийняття рішень на фактичних даних, а не припущеннях [4], [5].

У програмних системах управління завданнями кожен учасник повинен мати чітко визначену роль, оскільки це запобігає плутанині, підвищує відповідальність і сприяє злагодженій командній роботі. Правильне

налаштування ролей є основою адаптації менеджера завдань до специфіки проекту чи внутрішніх бізнес-процесів. Адміністратор отримує повний контроль над системою, керує користувачами, ролями та дозволами, відповідаючи за організацію всієї роботи. Керівник проекту зосереджується на плануванні, розподілі завдань і відстеженні прогресу, маючи доступ до звітів і аналітики для своєчасного реагування на ризики. Виконавці отримують призначені завдання, виконують їх та звітують про результати, маючи доступ лише до ресурсів, необхідних для виконання їхніх обов'язків. Спостерігачі можуть переглядати хід робіт та коментарі, не змінюючи інформацію, що зручно для зацікавлених сторін або керівництва, яке хоче залишатися в курсі проекту. Коментатори можуть брати участь в обговореннях і залишати свої відмітки, проте не змінюють статуси чи налаштування завдань [4], [5].

Ефективність програмного забезпечення для управління задачами користувача підвищується за умови чіткого опису кожного завдання, реалістичних термінів і регулярного оновлення статусу. Пріоритезація завдань допомагає зосереджуватися на найважливішому, а відкритий обмін інформацією через коментарі або інтеграції з месенджерами полегшує уточнення деталей та спільне вирішення проблем. Постійний аналіз ефективності процесів та впровадження оптимізацій дозволяє системі еволюціонувати разом із потребами команди, роблячи управління завданнями більш прозорим, структурованим і результативним [4], [5].

Визначено, що програмне забезпечення для управління задачами користувача представляє собою комплекс цифрових інструментів, призначених для організації, контролю та оптимізації процесів виконання завдань у рамках індивідуальної чи колективної діяльності. Такі програмні засоби забезпечують централізоване відстеження робочих процесів, дозволяють структурувати інформацію про завдання, терміни їх виконання та відповідальних осіб, а також надають можливість координувати діяльність між різними учасниками проєктів. Застосування подібного програмного забезпечення сприяє підвищенню продуктивності, оскільки користувач

отримує змогу планувати завдання, контролювати прогрес і своєчасно реагувати на зміни у робочих процесах.

1.2 Аналіз програмного забезпечення для управління задачами користувача

Проаналізуємо програмне забезпечення для управління задачами користувача [6]-[14].

Програмне забезпечення Nifty (рис. 1.1) є комплексним середовищем для організації задач, командної взаємодії та внутрішньої комунікації, у межах якого всі робочі процеси зосереджені в єдиному цифровому просторі. Такий підхід дозволяє зменшити розрізненість інструментів і забезпечити цілісне управління проектною діяльністю. Система орієнтована на використання в різних сферах професійної діяльності, зокрема в технічних, креативних та управлінських напрямках, де важливо координувати дії учасників і відстежувати прогрес виконання робіт. Завдяки візуальному поданню планів та можливостям спільної роботи в інтерактивному режимі забезпечується постійна узгодженість дій між членами команди [6], [11], [12].

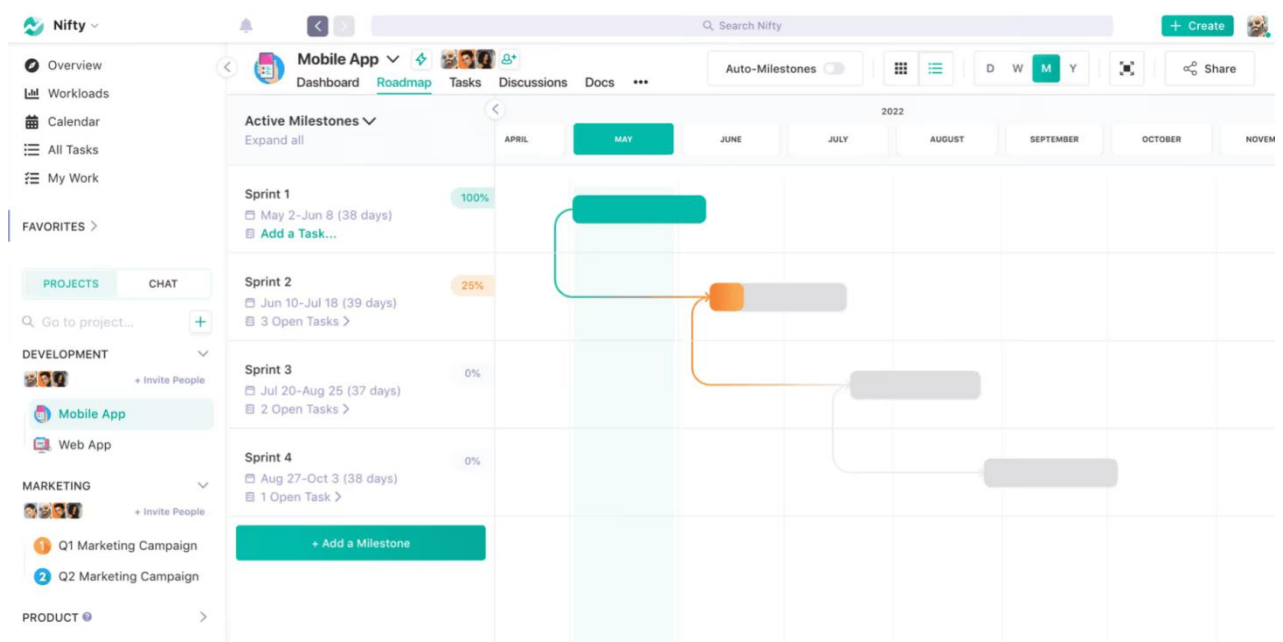


Рисунок 1.1 – Програмне забезпечення Nifty [12]

Програмне забезпечення Nifty характеризується ефективною реалізацією механізмів контролю етапів проєкту, які є однією з ключових відмінних рис платформи. Інтерфейс дорожніх карт надає змогу наочно відображати логіку розвитку проєкту та взаємозв'язок між завданнями, що сприяє кращому розумінню загальної мети та проміжних результатів. Пряме поєднання задач із відповідними етапами підвищує прозорість процесів і відповідальність виконавців. Додаткову цінність становить система аналітичного відображення прогресу, яка дозволяє оперативно оцінювати стан виконання робіт і своєчасно реагувати на відхилення [6], [11], [12].

Функціональні можливості Nifty не обмежуються лише плануванням і контролем завдань, а охоплюють також засоби оперативного спілкування, що інтегровані безпосередньо в робоче середовище. Це сприяє зменшенню залежності від сторонніх каналів зв'язку та підвищує концентрацію на проєктній діяльності. Програмне забезпечення Nifty підтримує фіксацію витрачених зусиль на виконання задач, що дозволяє більш усвідомлено підходити до розподілу ресурсів. Вбудовані засоби роботи з документацією забезпечують збереження та спільне використання матеріалів у межах одного простору, що спрощує доступ до актуальної інформації. Гнучкість налаштувань робочих процесів дає змогу адаптувати платформу до специфіки конкретної команди [6], [11], [12].

Важливою рисою програмного забезпечення Nifty є підтримка взаємодії з іншими програмними продуктами, що дозволяє інтегрувати її в уже наявну цифрову екосистему організації. Наявність програмного інтерфейсу розширює можливості індивідуального налаштування та автоматизації. Разом із тим, поряд із перевагами, такими як зручна візуальна організація задач, стабільна синхронізація даних та централізоване управління документацією, система має певні обмеження. Частина розширених функцій програмного забезпечення Nifty доступна лише в межах комерційних варіантів використання, що може впливати на вибір платформи для окремих категорій користувачів [6], [11], [12].

Основними характеристиками програмного забезпечення Nifty є такі [6], [11], [12]:

- централізоване управління проєктами: програмне забезпечення Nifty забезпечує об'єднання задач, комунікацій і матеріалів користувачів у межах єдиного робочого програмного середовища, що спрощує координацію командної діяльності [6], [11], [12];

- візуальне планування робіт: програмне забезпечення Nifty надає інструменти наочного відображення структури проєктів і взаємозв'язків між етапами, що полегшує контроль виконання завдань [6], [11], [12];

- інтегрована командна взаємодія: у межах платформи реалізовані засоби обміну повідомленнями та спільної роботи, що зменшує потребу у використанні сторонніх сервісів [6], [11], [12];

- підтримка контролю виконання задач: програмне забезпечення Nifty дозволяє відстежувати стан робіт та оперативно оцінювати прогрес реалізації проєкту [6], [11], [12];

- управління супровідною документацією: програмне забезпечення Nifty забезпечує зберігання, редагування та спільне використання файлів, пов'язаних із проєктною діяльністю [6], [11], [12];

- адаптивність робочих процесів: програмне забезпечення Nifty дозволяє налаштовувати логіку виконання задач відповідно до специфіки команди та характеру проєктів [6], [11], [12];

- можливості інтеграції: програмне забезпечення Nifty підтримує взаємодію з іншими цифровими інструментами та сервісами, що розширює функціональність робочого середовища [6], [11], [12].

Серед недоліків програмного забезпечення Nifty можна відзначити обмежену доступність окремих функціональних можливостей у базових режимах використання, а також залежність повного набору інструментів від вибору комерційного плану, що може знижувати привабливість системи для окремих категорій користувачів [6], [11], [12].

Програмне забезпечення MeisterTask (рис. 1.2) орієнтоване на використання у креативних колективах і сферах для планування задач, де важливу роль відіграє наочність та швидка взаємодія між учасниками робочого процесу. Система MeisterTask реалізує підхід візуального управління задачами через дошки, побудовані за принципами Канбан, що дозволяє легко відстежувати стан робіт і структурувати проекти без тривалого періоду навчання. Завдяки простому та зрозумілому інтерфейсу MeisterTask знижується потреба у постійних нарадах і надмірному листуванні, а основна увага зосереджується на виконанні завдань і підтриманні впорядкованості робочих процесів [6], [9], [13].

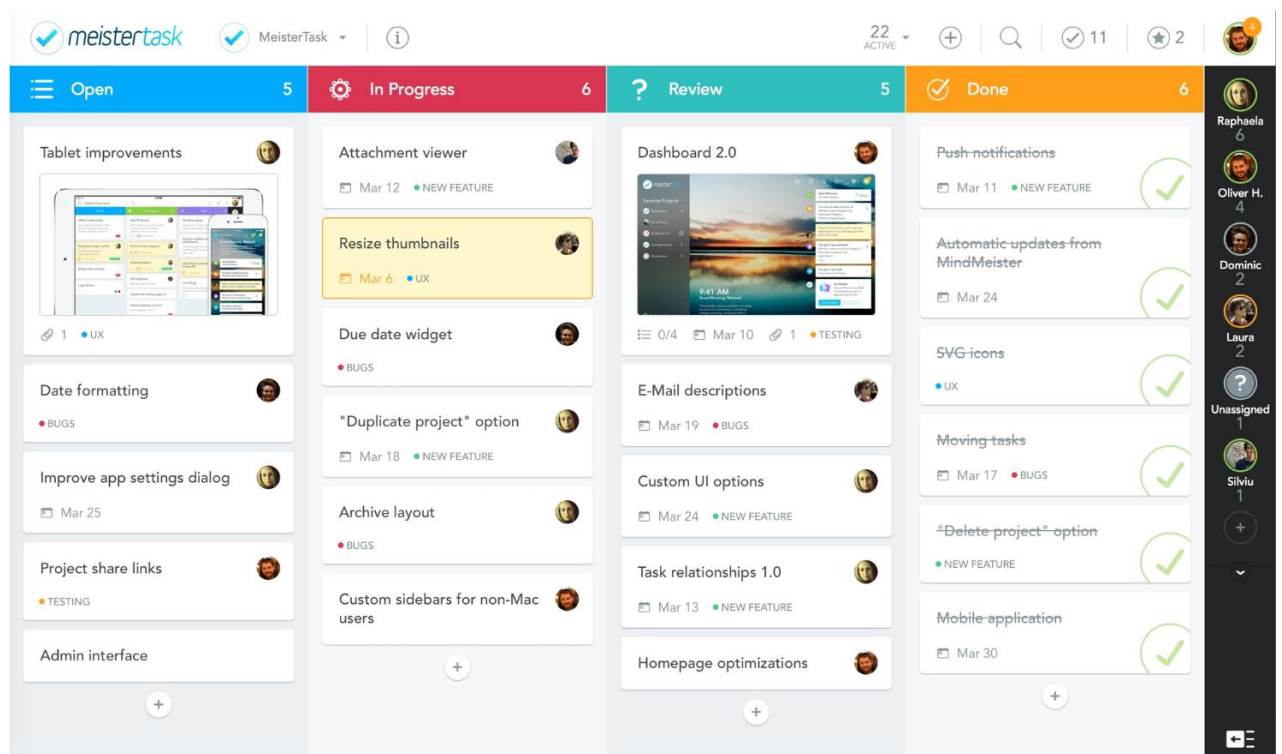


Рисунок 1.2 – Програмне забезпечення MeisterTask [13]

Програмне забезпечення MeisterTask спрямовано на ефективну командну взаємодію, яка є визначальним фактором для сучасних систем управління задачами. Візуальне представлення завдань сприяє прозорості процесів і полегшує координацію дій між учасниками команди. Актуалізація інформації відбувається синхронно для всіх користувачів, що дозволяє

постійно мати уявлення про стан виконання проєкту без необхідності регулярних обговорень. Додаткову ефективність забезпечують засоби автоматизації типових дій, які зменшують рутинне навантаження та дозволяють зосередитися на більш важливих аспектах діяльності [6], [9], [13].

Функціональні можливості MeisterTask охоплюють організацію відповідальності за виконання завдань, контроль витрачених зусиль і гнучке налаштування відображення інформації відповідно до потреб користувача. Система дозволяє формувати персоналізоване робоче середовище, у якому відображаються найбільш актуальні задачі та проєкти. Автоматичне відтворення повторюваних дій спрощує підтримку стабільних процесів і сприяє підвищенню загальної продуктивності [6], [9], [13].

Важливою особливістю програмного забезпечення MeisterTask є можливість інтеграції з популярними сервісами для комунікації, зберігання даних та розробки програмного забезпечення, що забезпечує безперервність роботи в межах існуючої цифрової інфраструктури. Поряд із такими перевагами, як зручність використання, підтримка автоматизації та наявність засобів календарного планування, система має і певні обмеження. Частина функціональних можливостей MeisterTask доступна лише у розширених режимах використання, а ієрархічна деталізація завдань реалізована у спрощеному вигляді, що може не повністю відповідати потребам складних проєктів [6], [9], [13].

Програмне забезпечення MeisterTask має такі особливості [6], [9], [13]:

- візуальне керування задачами: програмне забезпечення MeisterTask використовує підхід канбан-дошок для наочного подання станів завдань і спрощення контролю виконання робіт у межах проєкту [6], [9], [13];

- орієнтація на командну співпрацю: програмне забезпечення MeisterTask забезпечує спільний доступ до задач користувачів і миттєве оновлення інформації для всіх учасників робочого процесу, що покращує узгодженість дій [6], [9], [13];

- інтуїтивно зрозумілий інтерфейс: інтерфейс створено з урахуванням мінімального порогу входження, що дозволяє швидко розпочати роботу без тривалого навчання [6], [9], [13];
- автоматизація повторюваних дій: програмне забезпечення MeisterTask підтримує налаштування автоматичних сценаріїв для типових операцій, зменшуючи ручну працю користувачів [6], [9], [13];
- контроль витраченого часу: програмне забезпечення MeisterTask надає можливість фіксувати час виконання задач, що сприяє аналізу ефективності та оптимізації робочих процесів [6], [9], [13];
- персоналізація робочого простору: користувач програмного забезпечення MeisterTask може налаштовувати відображення задач і проєктів відповідно до власних потреб та пріоритетів [6], [9], [13];
- підтримка інтеграцій: програмне забезпечення MeisterTask взаємодіє з іншими цифровими сервісами, що дозволяє вбудовувати його у наявну інформаційну інфраструктуру [6], [9], [13].

Серед недоліків програмного забезпечення MeisterTask можна відзначити обмежений доступ до окремих функцій автоматизації у базових режимах використання, а також спрощену модель деталізації завдань, яка не завжди є достатньою для управління складними багаторівневими проєктами [6], [9], [13].

Програмне забезпечення FunctionFox (рис. 1.3) є спеціалізованим інструментом для організації проєктної діяльності, орієнтований переважно на представників креативних професій, які працюють одночасно з кількома завданнями та клієнтськими проєктами. Програмне забезпечення FunctionFox спрямоване на підтримку впорядкованості робочих процесів і контролю витраченого часу, що дозволяє уникати порушень графіків і втрати важливої інформації. Завдяки зрозумілій логіці взаємодії з інтерфейсом програмного забезпечення FunctionFox забезпечується швидке освоєння функціоналу та зручне керування поточними проєктами [6], [14].

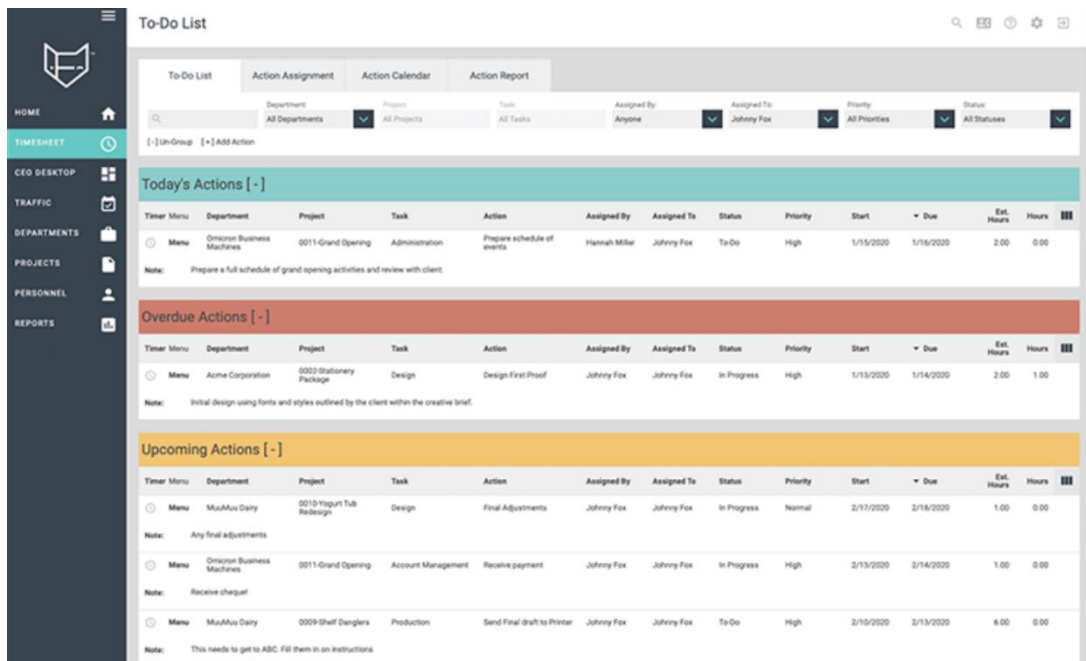


Рисунок 1.3 – Програмне забезпечення FunctionFox [14]

Програмне забезпечення FunctionFox пристосовано до потреб творчих команд, для яких важливими є точний облік виконаної роботи та прозоре планування завдань. Програмне забезпечення FunctionFox дозволяє одночасно контролювати кілька напрямів діяльності без перевантаження користувача, а розвинуті механізми фіксації часу сприяють коректному врахуванню як оплачуваної, так і внутрішньої роботи. Окрему цінність становить можливість програмного забезпечення FunctionFox постійного контролю фінансових меж проєктів, що допомагає підтримувати баланс між витратами та запланованими ресурсами в умовах багатозадачності [6], [14].

Функціональне наповнення FunctionFox сприяє покращенню управління завданнями шляхом візуального представлення планів і строків виконання робіт. Система підтримує календарне планування, що забезпечує наочний огляд завантаженості та послідовності дій. Засоби автоматичного сповіщення зменшують ризик пропуску важливих етапів і допомагають підтримувати стабільний ритм роботи. Механізми детального планування проєктів дозволяють більш ефективно розподіляти ресурси та відповідальність між учасниками команди [6], [14].

Водночас програмне забезпечення характеризується обмеженою взаємодією з зовнішніми сервісами, що може ускладнювати інтеграцію з іншими інструментами обліку та фінансового управління. Частина розширених можливостей доступна лише у певних варіантах використання, що може впливати на гнучкість застосування системи. Попри це, FunctionFox залишається ефективним рішенням для організацій, які потребують надійного контролю часу, завдань і бюджету в межах креативної діяльності [6], [14].

Програмне забезпечення FunctionFox має такі особливості [6], [14]:

- орієнтація на креативні команди: програмне забезпечення розроблене з урахуванням специфіки роботи дизайнерів, маркетологів та інших творчих фахівців, які одночасно працюють із кількома проєктами [6], [14];
- управління проєктами та задачами: програмне забезпечення FunctionFox забезпечує централізоване створення, розподіл і контроль виконання завдань у межах різних проєктів [6], [14];
- вбудоване відстеження часу: програмне забезпечення підтримує фіксацію витраченого часу на виконання задач, що дозволяє контролювати робоче навантаження та коректно обліковувати виконану роботу [6], [14];
- бюджетний контроль: реалізовані механізми моніторингу витрат у межах проєктів сприяють дотриманню фінансових обмежень і підвищенню прозорості використання ресурсів [6], [14];
- календарне планування: програмне забезпечення FunctionFox надає візуальне представлення графіків виконання робіт, що допомагає впорядковувати задачі та координувати строки реалізації проєктів [6], [14];
- автоматичні нагадування: програмне забезпечення FunctionFox підтримує сповіщення про важливі події та завдання, зменшуючи ризик пропуску ключових етапів роботи [6], [14];
- призначення відповідальних осіб: програмне забезпечення FunctionFox дозволяє закріплювати завдання за конкретними виконавцями, що підвищує рівень відповідальності та організованості команди [6], [14].

Серед недоліків програмного забезпечення FunctionFox можна відзначити обмежену кількість інтеграцій із зовнішніми сервісами, а також доступність окремих інструментів планування й аналізу лише у розширених варіантах використання, що може знижувати гнучкість застосування системи в комплексних робочих середовищах [6], [14].

Порівняльну характеристику програмного забезпечення для управління задачами користувача наведено у таблиці 1.1.

Таблиця 1.1 – Порівняння програмного забезпечення для управління задачами користувача

Критерій порівняння	Nifty [12]	MeisterTask [13]	FunctionFox [14]
Інтуїтивний інтерфейс	+	+	+—
Візуальне планування	+	+—	+—
Відстеження часу	+—	+	+
Аналіз продуктивності та звітність	+—	+	+—
Персоналізація робочого простору	+	+—	+—
Інтеграції із зовнішніми сервісами	+	+	—

За результатами проведеного аналізу можна зробити висновок, що у наш час існує досить багато програмних засобів для управління задачами користувача. Проте незважаючи на значні переваги, використання таких програмних систем не завжди забезпечує очікуваний результат та в окремих випадках створює додаткові ускладнення в процесі діяльності користувача. Одним із суттєвих недоліків є складність освоєння окремих програмних рішень. Надмірна функціональність, складний інтерфейс або недостатньо

зрозуміла логіка роботи можуть ускладнювати процес адаптації користувачів, особливо за відсутності спеціальної підготовки. У таких випадках час, витрачений на навчання та налаштування системи, може перевищувати вигоди від її використання. Ще одним недоліком є ризик інформаційного перевантаження. Постійні сповіщення, нагадування та оновлення статусів здатні знижувати концентрацію уваги та спричиняти фрагментацію робочого процесу. Крім того, питання захисту персональних даних і конфіденційності інформації залишаються актуальними, особливо при використанні хмарних сервісів. Тому актуальною є розробка програмного забезпечення для управління задачами користувача.

При розробці програмного забезпечення для управління задачами користувача необхідно забезпечити такі функціональні вимоги:

- підтримка можливості створення, перегляду, редагування та видалення дошок для організації задач користувача;
- підтримка можливості реєстрації та авторизації користувачів для забезпечення контролю доступу до персональних даних;
- можливість збереження інформації про користувачів, дошки, колонки, картки та замітки у централізованій базі даних;
- можливість можливості створення, редагування, переміщення та видалення карток завдань користувача;
- підтримка можливості додавання та перегляду заміток, пов'язаних із окремими картками;
- підтримка можливості сортування дошок відповідно до обраних параметрів відображення.

1.3 Висновки за розділом 1

Визначено, що програмне забезпечення для управління задачами користувача представляє собою комплекс цифрових інструментів, призначених для організації, контролю та оптимізації процесів виконання

завдань у рамках індивідуальної чи колективної діяльності. Такі програмні засоби забезпечують централізоване відстеження робочих процесів, дозволяють структурувати інформацію про завдання, терміни їх виконання та відповідальних осіб, а також надають можливість координувати діяльність між різними учасниками проєктів. Застосування подібного програмного забезпечення сприяє підвищенню продуктивності, оскільки користувач отримує змогу планувати завдання, контролювати прогрес і своєчасно реагувати на зміни у робочих процесах.

За результатами проведеного аналізу зроблено висновок, що у наш час існує досить багато програмних засобів для управління задачами користувача. Проте незважаючи на значні переваги, використання таких програмних систем не завжди забезпечує очікуваний результат та в окремих випадках створює додаткові ускладнення в процесі діяльності користувача. Одним із суттєвих недоліків є складність освоєння окремих програмних рішень. Надмірна функціональність, складний інтерфейс або недостатньо зрозуміла логіка роботи можуть ускладнювати процес адаптації користувачів, особливо за відсутності спеціальної підготовки. У таких випадках час, витрачений на навчання та налаштування системи, може перевищувати вигоди від її використання. Ще одним недоліком є ризик інформаційного перевантаження. Постійні сповіщення, нагадування та оновлення статусів здатні знижувати концентрацію уваги та спричиняти фрагментацію робочого процесу. Замість підвищення продуктивності це може призводити до зростання рівня стресу та відчуття постійної зайнятості без реального прогресу у виконанні завдань. Крім того, питання захисту персональних даних і конфіденційності інформації залишаються актуальними, особливо при використанні хмарних сервісів. Тому актуальною є розробка програмного забезпечення для управління задачами користувача.

Сформульовано функціональні вимоги до програмного забезпечення для управління задачами користувача.

2 МАТЕРІАЛИ І МЕТОДИ

2.1 Вибір мови програмування

При розробці програмного забезпечення для управління задачами користувача було використано мову програмування JavaScript [15], [16].

Мова програмування JavaScript орієнтована на подієву модель програмування та асинхронну обробку даних. Це дозволяє ефективно працювати з користувацькими діями, мережевими запитами та фоновими процесами без блокування основного потоку виконання програми. Для прикладних систем, що потребують швидкого реагування на зміни стану даних або взаємодії з користувачем у режимі реального часу, така особливість є принципово важливою [15], [16].

Окремої уваги заслуговує стандартизація мови JavaScript, яка здійснюється у межах специфікації ECMAScript. Регулярне оновлення стандарту сприяє впровадженню нових мовних конструкцій, підвищенню безпеки та покращенню читабельності коду. Це забезпечує сумісність програмних рішень між різними платформами та сприяє довготривалому супроводу програмного забезпечення [15], [16].

JavaScript також характеризується високим рівнем підтримки з боку спільноти розробників. Наявність великої кількості відкритих програмних бібліотек, навчальних матеріалів і технічної документації створює сприятливі умови для вивчення мови та впровадження її у практичних проєктах. Така екосистема сприяє зниженню порогу входження та підвищенню якості програмних продуктів завдяки використанню перевірених рішень [15], [16].

Таким чином, мова програмування JavaScript може розглядатися не лише як технічний інструмент реалізації вебзастосунків, а як універсальна платформа для розробки сучасних програмних систем. Її гнучкість, постійний розвиток і відповідність актуальним вимогам програмної інженерії роблять JavaScript доцільним вибором для створення різноманітних прикладних рішень, зокрема систем управління задачами користувача [15], [16].

Доцільність використання мови програмування JavaScript для розробки програмного забезпечення з управління задачами користувача обґрунтовується її технологічними можливостями та відповідністю сучасним вимогам до інформаційних систем. У сучасних умовах такі програмні продукти мають бути інтерактивними, кросплатформними, зручними у використанні та доступними з різних пристроїв, що безпосередньо визначає вибір інструментів реалізації [15], [16].

JavaScript є базовою мовою веброзробки, що забезпечує створення динамічних та адаптивних інтерфейсів користувача. Оскільки програмне забезпечення для управління задачами орієнтоване на постійну взаємодію з користувачем, оперативне оновлення даних та візуальне відображення стану завдань, використання JavaScript дозволяє реалізувати ці вимоги без необхідності встановлення додаткового програмного забезпечення. Доступ до системи через веббраузер суттєво розширює аудиторію користувачів та підвищує зручність експлуатації [15], [16].

Важливою перевагою JavaScript є можливість використання цієї мови як на стороні клієнта, так і на стороні сервера. Застосування серверних платформ на основі JavaScript, зокрема середовища Node.js, дає змогу створювати повнофункціональні клієнт-серверні застосунки з єдиною мовою програмування. Це спрощує процес розробки, зменшує складність супроводу програмного забезпечення та підвищує узгодженість логіки обробки даних [15], [16].

Крім того, JavaScript має розвинену екосистему бібліотек і фреймворків, які значно прискорюють розробку програмного забезпечення. Використання сучасних інструментів дає змогу ефективно реалізовувати функціональність управління задачами, синхронізацію даних у реальному часі, роботу з календарями та повідомленнями, а також інтеграцію з іншими інформаційними сервісами. Це є особливо актуальним для систем, орієнтованих на індивідуальне та колективне планування діяльності [15], [16].

Таким чином, використання мови програмування JavaScript для розробки програмного забезпечення з управління задачами користувача є доцільним з огляду на її універсальність, поширеність та відповідність вимогам сучасних веборієнтованих інформаційних систем. Застосування цієї мови дозволяє створювати гнучкі, масштабовані та зручні у використанні програмні продукти, що забезпечують ефективну організацію діяльності користувача [15], [16].

Обґрунтування вибору мови програмування для розробки програмного забезпечення для управління задачами користувача наведено у таблиці 2.1.

Таблиця 2.1 – Обґрунтування вибору мови програмування для розробки програмного забезпечення для управління задачами користувача

Критерій порівняння мов програмування	Мова програмування		
	JavaScript	C#	PHP
Підтримка інтерактивних вебінтерфейсів	+	+-	-
Можливість використання на клієнті і сервері	+	-	-
Кросплатформність	+	+	+-
Швидкість розробки вебзастосунків	+	+-	+-
Простота інтеграції з зовнішніми сервісами	+	+-	+-

Отже, для реалізації програмного забезпечення для управління задачами користувача обрано мову програмування JavaScript, що обумовлено її універсальністю, можливістю повноцінного використання як на стороні клієнта, так і на стороні сервера, а також тісною інтеграцією з вебтехнологіями. Крім того, JavaScript має розвинену екосистему бібліотек і

фреймворків, які значно прискорюють розробку програмного забезпечення. Використання сучасних інструментів дає змогу ефективно реалізовувати функціональність управління задачами, синхронізацію даних у реальному часі, роботу з календарями та повідомленнями, а також інтеграцію з іншими інформаційними сервісами.

2.2 Вибір серверного середовища виконання для програмного забезпечення для управління задачами користувача

В якості серверного середовища виконання для програмного забезпечення для управління задачами користувача було обрано середовище Node.js [17], [18].

Node.js є серверним середовищем виконання, яке забезпечує запуск коду мовою програмування JavaScript поза межами веббраузера. Його поява стала важливим етапом розвитку вебтехнологій, оскільки дозволила використовувати JavaScript не лише для реалізації інтерфейсів користувача, а й для побудови повноцінної серверної логіки. В основі Node.js лежить рушій V8, що забезпечує високу швидкодію та ефективне виконання програмного коду [17], [18].

Особливістю Node.js є подієво-орієнтована та неблокувальна модель введення-виведення, яка дозволяє обробляти значну кількість одночасних запитів без створення окремих потоків для кожного з них. Такий підхід є доцільним для розробки вебзастосунків, орієнтованих на активну взаємодію з користувачами, часті запити до сервера та роботу з даними у реальному часі. У системах управління задачами це забезпечує оперативне оновлення стану завдань і стабільну роботу при збільшенні кількості користувачів [17], [18].

Додатково про Node.js можна зазначити його роль як платформи для побудови масштабованих мережових застосунків. Завдяки модульній архітектурі та підтримці пакетного менеджера npm створюються умови для повторного використання коду, стандартизації підходів до розробки та

швидкого впровадження нових функціональних можливостей. Це робить Node.js привабливим рішенням не лише для комерційних проєктів, а й для навчальних і наукових розробок [17], [18].

Node.js доцільно порівнювати з іншими серверними технологіями, які виконують аналогічні функції. Зокрема, його часто зіставляють з Java-платформами на основі сервлетів або фреймворків, такими як Spring, які використовують багатопотокову модель обробки запитів. У порівнянні з ними Node.js пропонує простішу архітектуру для високонавантажених, подієвих застосунків, хоча може поступатися в реалізації складної бізнес-логіки корпоративного рівня [17], [18].

Аналогом Node.js також є серверні середовища на основі мови Python, наприклад застосунки з використанням Django або Flask. Такі рішення відзначаються високою читабельністю коду та зручністю розробки, однак у задачах, що потребують обробки великої кількості одночасних з'єднань, Node.js часто демонструє кращу масштабованість. Крім того, порівняння можливе з платформами на основі C# та .NET, які забезпечують високу стабільність і тісну інтеграцію з екосистемою Microsoft, але зазвичай мають складніший процес налаштування [17], [18].

Таким чином, Node.js є серверним середовищем виконання JavaScript, орієнтованим на створення масштабованих та високопродуктивних вебзастосунків. У порівнянні з іншими серверними технологіями воно вирізняється подієвою моделлю, єдністю технологічного стеку та широкою екосистемою інструментів, що робить його доцільним вибором для розробки програмного забезпечення з управління задачами користувача [17], [18].

Доцільність використання середовища виконання Node.js для розробки програмного забезпечення з управління задачами користувача визначається його архітектурними особливостями та відповідністю вимогам сучасних клієнт-серверних і веборієнтованих систем. Такі програмні продукти передбачають постійну взаємодію з користувачем, обробку великої кількості подій, синхронізацію даних у реальному часі та одночасну роботу багатьох

користувачів, що безпосередньо впливає на вибір серверної платформи [17], [18].

Node.js побудований на асинхронній, подієво-орієнтованій моделі виконання, яка забезпечує ефективну обробку численних запитів без створення великої кількості потоків. Це дозволяє досягти високої продуктивності та масштабованості серверної частини системи, що є особливо актуальним для програмного забезпечення управління задачами, де користувачі часто змінюють стан завдань, отримують оновлення та працюють з даними в режимі реального часу [17], [18].

Важливою перевагою Node.js є можливість використання мови програмування JavaScript як для клієнтської, так і для серверної частини застосунку. Такий підхід забезпечує єдність технологічного стеку, спрощує процес розробки та супроводу програмного забезпечення, а також зменшує кількість помилок, пов'язаних з різною реалізацією бізнес-логіки на різних рівнях системи. Для навчальних і прикладних проєктів це має особливе значення, оскільки знижує складність архітектури [17], [18].

Node.js також характеризується розвиненою екосистемою модулів і пакетів, що поширюються через менеджер пакетів npm. Це дає змогу швидко інтегрувати функціональність для роботи з базами даних, автентифікації користувачів, обробки запитів, обміну даними та взаємодії з зовнішніми сервісами. У контексті систем управління задачами це сприяє скороченню часу розробки та підвищенню якості програмного продукту [17], [18].

Таким чином, використання Node.js для розробки програмного забезпечення з управління задачами користувача є доцільним завдяки його продуктивності, масштабованості та тісній інтеграції з JavaScript. Це середовище забезпечує ефективну реалізацію серверної логіки, відповідає сучасним вимогам до вебсистем і створює сприятливі умови для побудови надійних та гнучких інформаційних рішень [17], [18].

Обґрунтування вибору серверного середовища виконання для програмного забезпечення для управління задачами користувача наведено у таблиці 2.2.

Таблиця 2.2 – Обґрунтування вибору середовища виконання для програмного забезпечення для управління задачами користувача

Критерій порівняння середовищ виконання	Середовища виконання		
	Node.js	Deno	Cloudflare Workers
Зрілість та стабільність платформи	+	+—	+—
Поширеність у промисловій розробці	+	+—	+—
Екосистема пакетів і бібліотек	+	+—	—
Масштабованість для багатокористувацьких систем	+	+—	+
Простота розгортання та хостингу	+	+—	+—

Таким чином, в якості серверного середовища виконання для програмного забезпечення для управління задачами обрано Node.js, що обумовлено його архітектурними особливостями та відповідністю вимогам сучасних клієнт-серверних і веборієнтованих систем. Крім того, Node.js побудований на асинхронній, подієво-орієнтованій моделі виконання, яка забезпечує ефективну обробку численних запитів без створення великої кількості потоків. Це дозволяє досягти високої продуктивності та масштабованості серверної частини системи, що є особливо актуальним для

програмного забезпечення управління задачами, де користувачі часто змінюють стан завдань, отримують оновлення та працюють з даними в режимі реального часу.

2.3 Висновки за розділом 2

Для реалізації програмного забезпечення для управління задачами користувача обрано мову програмування JavaScript, що обумовлено її універсальністю, можливістю повноцінного використання як на стороні клієнта, так і на стороні сервера, а також тісною інтеграцією з вебтехнологіями. Крім того, JavaScript має розвинену екосистему бібліотек і фреймворків, які значно прискорюють розробку програмного забезпечення.

В якості серверного середовища виконання для програмного забезпечення для управління задачами обрано Node.js, що обумовлено його архітектурними особливостями та відповідністю вимогам сучасних клієнт-серверних і веборієнтованих систем. Крім того, Node.js побудований на асинхронній, подієво-орієнтованій моделі виконання, яка забезпечує ефективну обробку численних запитів без створення великої кількості потоків. Це дозволяє досягти високої продуктивності та масштабованості серверної частини системи, що є особливо актуальним для програмного забезпечення управління задачами, де користувачі часто змінюють стан завдань, отримують оновлення та працюють з даними в режимі реального часу.

3 ОПИС ПРОГРАМИ

3.1 Структура програмного забезпечення для управління задачами користувача

Структуру програмного забезпечення для управління задачами користувача наведено на рис. 3.1.



Рисунок 3.1 – Структура програмного забезпечення
для управління задачами користувача

Структура програмного забезпечення для управління задачами користувача побудована за модульним принципом, що забезпечує логічний поділ функціональності, спрощує розробку, тестування та подальший супровід системи. У межах такого підходу програмне забезпечення для управління задачами користувача складається з шести взаємопов'язаних модулів (модуль автентифікації та авторизації користувачів, модуль управління базою даних, модуль управління дошками, модуль управління колонками та картками, модуль взаємодії та динамічних змін, модуль користувацького інтерфейсу та представлення даних), кожен з яких виконує чітко визначені завдання та відповідає за окремий аспект функціонування вебзастосунку.

Ключовим елементом структури є модуль автентифікації та авторизації користувачів. Його призначення полягає в забезпеченні безпечного доступу до системи шляхом реєстрації нових користувачів, перевірки облікових даних під час входу та контролю прав доступу до ресурсів застосунку. У межах цього модуля реалізуються функції створення облікового запису, входу та виходу із системи, збереження сесій, а також базові механізми захисту персональних даних.

Наступним важливим компонентом програмного забезпечення для управління задачами користувача є модуль управління базою даних, який відповідає за збереження, оновлення та отримання інформації. Його основним призначенням є організація взаємодії між серверною логікою та сховищем даних, що містить інформацію про користувачів, дошки, колонки, картки та замітки. У цьому модулі реалізуються операції створення, читання, оновлення та видалення даних, а також забезпечується цілісність і узгодженість інформації.

Центральне місце у функціональній структурі займає модуль управління дошками, який забезпечує створення та редагування робочих просторів користувача. Його призначення полягає у наданні можливості додавати нові дошки, змінювати їх назви та параметри, а також здійснювати сортування за заданими критеріями, зокрема за назвою або часом створення.

Функції цього модуля спрямовані на організацію зручної навігації та структурування задач.

Тісно пов'язаним із попереднім є модуль управління колонками та картками, який реалізує основну логіку роботи із задачами. У межах цього модуля забезпечується додавання, редагування та видалення колонок і карток, а також підтримується можливість наповнення карток замітками. Саме цей компонент відповідає за відображення поточного стану виконання задач і формує основу для візуального представлення робочого процесу.

Окремим структурним елементом програмного забезпечення для управління задачами користувача є модуль взаємодії та динамічних змін, призначений для обробки дій користувача в реальному часі. Основною його функцією є забезпечення пересування карток між колонками без перезавантаження сторінки, оновлення інтерфейсу відповідно до змін стану даних та синхронізація клієнтської і серверної частин застосунку. Цей модуль відіграє важливу роль у підвищенні зручності та інтуїтивності використання системи.

Важливим компонентом структури є модуль користувацького інтерфейсу та представлення даних. Його призначення полягає у візуалізації інформації, забезпеченні зручної навігації та взаємодії користувача з програмним забезпеченням. У межах цього модуля реалізується відображення дошок, колонок і карток, форм введення даних, повідомлень про результати дій, а також адаптація інтерфейсу до різних пристроїв і розмірів екранів.

Таким чином, запропонована структура програмного забезпечення для управління задачами користувача охоплює всі основні функціональні аспекти вебзастосунку та забезпечує логічний поділ відповідальності між модулями. Такий підхід створює умови для стабільної роботи системи, зручності її використання та можливості подальшого розширення функціональних можливостей.

3.2 Функціонування програмного забезпечення для управління задачами користувача

Функціонування програмного забезпечення для управління задачами користувача подамо за допомогою схеми, зображеної на рис. 3.2.



Рисунок 3.2 – Схема функціонування програмного забезпечення для управління задачами користувача

Послідовність функціонування програмного забезпечення для управління задачами користувача доцільно розглядати як логічно

впорядкований ланцюг дій, що відображає взаємодію користувача з вебзастосунком та внутрішні процеси обробки даних. Такий опис дозволяє безпосередньо використати його для побудови структурної або функціональної схеми роботи системи.

Функціонування програмного забезпечення розпочинається з ініціалізації вебзастосунку та завантаження головної сторінки, під час чого виконується перевірка наявності активної користувацької сесії. У разі відсутності підтверджених облікових даних користувачеві пропонується виконати реєстрацію або авторизацію шляхом введення відповідної інформації. Після надсилання даних на сервер здійснюється їх перевірка, а у випадку успішної автентифікації створюється сесія та надається доступ до основного функціоналу системи.

Після входу до системи відбувається завантаження персональних даних користувача з бази даних, зокрема переліку створених дошок. На цьому етапі виконується формування користувацького інтерфейсу та відображення дошок із урахуванням обраного порядку сортування за назвою або часом створення. Користувач отримує можливість обрати існуючу дошку або створити нову, що ініціює відповідні запити до серверної частини та збереження змін у базі даних.

Після вибору конкретної дошки система здійснює запит на отримання пов'язаних із нею колонок та карток. Отримані дані обробляються та відображаються у вигляді структурованого робочого простору, що відображає поточний стан виконання задач. У разі додавання або редагування колонок, карток чи заміток відповідні дії користувача передаються на сервер, де відбувається оновлення інформації у базі даних з подальшим поверненням актуального стану до клієнтської частини.

Наступним етапом є взаємодія користувача з картками у межах дошки, зокрема їх переміщення між колонками. Під час виконання такої дії система фіксує зміну положення картки, надсилає оновлені дані на сервер та синхронізує їх з базою даних. Після підтвердження успішного виконання

операції інтерфейс автоматично оновлюється без перезавантаження сторінки, що забезпечує безперервність роботи користувача.

У процесі подальшого використання застосунку всі зміни, ініційовані користувачем, проходять через аналогічний цикл обробки, що включає фіксацію дії, передачу даних, оновлення сховища та відображення актуального результату. Завершальним етапом функціонування є вихід користувача з системи, під час якого сесія завершується, а доступ до персональних даних блокується до наступної авторизації.

Таким чином, описана послідовність функціонування відображає повний цикл роботи програмного забезпечення для управління задачами користувача.

У програмному забезпеченні для управління задачами користувача функціонування основних можливостей забезпечується сукупністю взаємопов'язаних алгоритмів, які реалізують повний цикл взаємодії користувача з вебзастосунком та базою даних. Кожен із таких алгоритмів орієнтований на обробку конкретного типу запитів і забезпечує узгоджену роботу клієнтської та серверної частин системи.

Алгоритм реєстрації та авторизації призначений для організації контрольованого доступу до програмного забезпечення. Його виконання ініціюється введенням користувачем облікових даних, які передаються на сервер для перевірки коректності та відповідності вимогам безпеки. У разі створення нового облікового запису здійснюється збереження інформації про користувача в базі даних, після чого формується відповідь серверної частини з результатом операції. За умови успішної перевірки даних створюється активна сесія, що надає доступ до функціональних можливостей системи.

Алгоритм отримання дошок, колонок та карток забезпечує формування робочого простору користувача після входу до системи або під час переходу між розділами застосунку. Під час його виконання клієнтська частина надсилає запит до серверної логіки, яка звертається до відповідних таблиць бази даних для вибірки структурованої інформації. Отримані дані

обробляються та повертаються у вигляді впорядкованого набору, що відображається в інтерфейсі користувача.

Алгоритм отримання заміток реалізується у момент взаємодії з окремою картою. Після відкриття діалогового вікна формується запит до серверної частини із зазначенням ідентифікатора картки, на основі якого здійснюється пошук пов'язаних записів у базі даних. Результат виконання алгоритму передається на інтерфейсну частину, де відбувається відображення текстової інформації без необхідності перезавантаження сторінки.

Алгоритм створення дошок, колонок та карток відповідає за динамічне розширення структури даних користувача. Після ініціації дії створення відповідні дані передаються на сервер, де відбувається формування нових записів у базі даних із прив'язкою до конкретного користувача або батьківського елемента. Після успішного збереження оновлений стан структури повертається на клієнтську частину та миттєво відображається в інтерфейсі.

Алгоритм редагування карток реалізує можливість зміни параметрів задач без порушення цілісності системи. У процесі його виконання сервер отримує оновлені дані для конкретної картки, здійснює їх перевірку та збереження у відповідній таблиці бази даних. Після завершення операції клієнтська частина отримує актуалізовану інформацію, що дозволяє відобразити зміни в реальному часі.

Алгоритм видалення дошок, колонок та карток ґрунтується на принципі каскадної обробки даних. Після надсилання запиту на видалення серверна частина визначає пов'язані елементи та забезпечує їх послідовне вилучення з бази даних. Такий підхід гарантує збереження логічної цілісності даних та запобігає появі неузгоджених записів. Результат виконання операції передається клієнтській частині у вигляді підтвердження успішного завершення.

Алгоритм сортування дошок призначений для впорядкування робочих просторів відповідно до обраного критерію. Після вибору користувачем

параметра сортування формується запит до серверної частини, яка здійснює відповідну обробку даних у базі. Відсортований набір дошок повертається до інтерфейсу, що забезпечує зручну навігацію та підвищує ефективність роботи з системою.

Таким чином, сукупність описаних алгоритмів формує логічну та функціонально завершену основу програмного забезпечення для управління задачами користувача, забезпечуючи стабільну роботу, зручність використання та узгоджену взаємодію всіх компонентів системи.

3.3 Розробка бази даних програмного забезпечення для управління задачами користувача

База даних програмного забезпечення для управління задачами користувача розроблена з урахуванням необхідності зберігання структурованої інформації про користувачів, їхні робочі простори та елементи задач, а також забезпечення цілісності й узгодженості даних у процесі взаємодії з вебзастосунком. Її логічна модель (рис. 3.3) орієнтована на підтримку основних функціональних можливостей системи та передбачає чіткі зв'язки між сутностями, що відображають ієрархічну організацію задач.

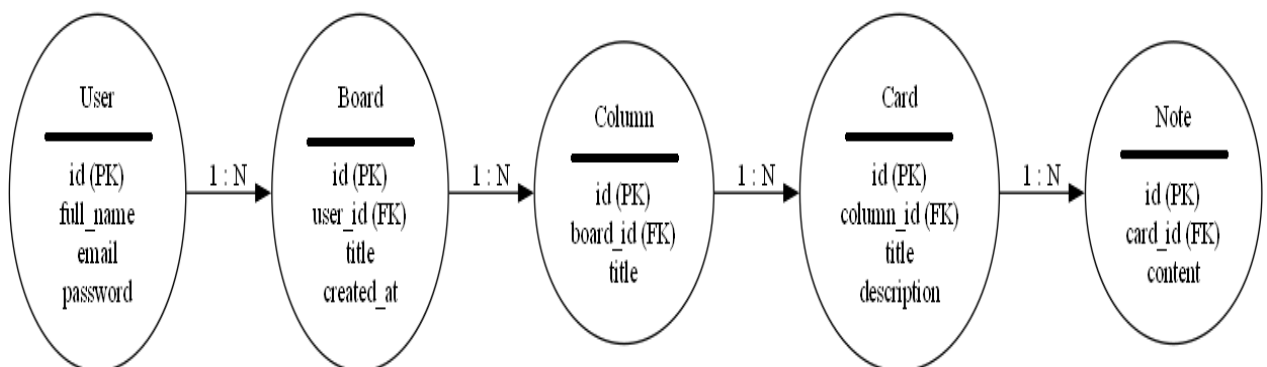


Рисунок 3.3 – Модель бази даних програмного забезпечення для управління задачами користувача

Центральною складовою бази даних є сутність, що містить інформацію про користувачів системи. У межах цієї сутності зберігаються ідентифікаційні та облікові дані, необхідні для реалізації процесів реєстрації та авторизації. Збереження інформації здійснюється таким чином, щоб забезпечити безпеку доступу та унеможливити несанкціоноване використання персональних даних, при цьому кожному користувачеві відповідає унікальний ідентифікатор, який використовується для встановлення зв'язків з іншими таблицями.

Для організації робочого простору в базі даних передбачена сутність, що відповідає за збереження інформації про дошки. Вона пов'язується з користувачами за принципом належності, що дозволяє кожному користувачеві мати власний набір дошок. У межах цієї структури фіксуються основні параметри, необхідні для ідентифікації та сортування, зокрема назва та час створення, що забезпечує коректну реалізацію функцій впорядкування даних.

Наступний рівень ієрархії представлений сутністю, яка зберігає інформацію про колонки. Кожна колонка прив'язується до конкретної дошки, що дозволяє формувати логічну структуру робочого процесу. Такий підхід забезпечує можливість гнучкої організації задач за етапами виконання та підтримує операції додавання, редагування і видалення елементів без порушення загальної цілісності даних.

Задачі в системі представлені у вигляді карток, дані про які зберігаються в окремій сутності бази даних. Картки пов'язуються з відповідними колонками та містять інформацію, необхідну для відображення стану та змісту задачі. Завдяки такій структурі забезпечується можливість переміщення карток між колонками шляхом оновлення зв'язків у базі даних, що відображає зміну етапу виконання задачі.

Для зберігання додаткової текстової інформації, пов'язаної з картками, використовується сутність заміток. Вона забезпечує збереження розширених описів або коментарів, які асоціюються з конкретними задачами. Зв'язок між картками та замітками дозволяє розширювати функціональні можливості системи без ускладнення основної структури даних.

Таким чином, база даних програмного забезпечення для управління задачами користувача має ієрархічну, логічно впорядковану структуру, що відображає взаємозв'язки між користувачами, дошками, колонками, картками та замітками. Така організація забезпечує надійне зберігання інформації, підтримку всіх основних операцій над даними та можливість масштабування системи в процесі її подальшого розвитку.

3.4 Проєктування інтерфейсу програмного забезпечення для управління задачами користувача

Проєктування інтерфейсу взаємодії користувача з програмним забезпеченням для управління задачами користувача виконано з урахуванням вимог технічного завдання. При розробленні програмного забезпечення для управління задачами користувача враховані функціональні вимоги до програми:

- підтримка можливості створення, перегляду, редагування та видалення дошок для організації задач користувача;
- підтримка можливості реєстрації та авторизації користувачів для забезпечення контролю доступу до персональних даних;
- можливість збереження інформації про користувачів, дошки, колонки, картки та замітки у централізованій базі даних;
- можливість можливості створення, редагування, переміщення та видалення карток завдань користувача;
- підтримка можливості додавання та перегляду заміток, пов'язаних із окремими картками;
- підтримка можливості сортування дошок відповідно до обраних параметрів відображення.

Інтерфейс головної сторінки програмного забезпечення для управління задачами користувача наведено на рис. 3.4.

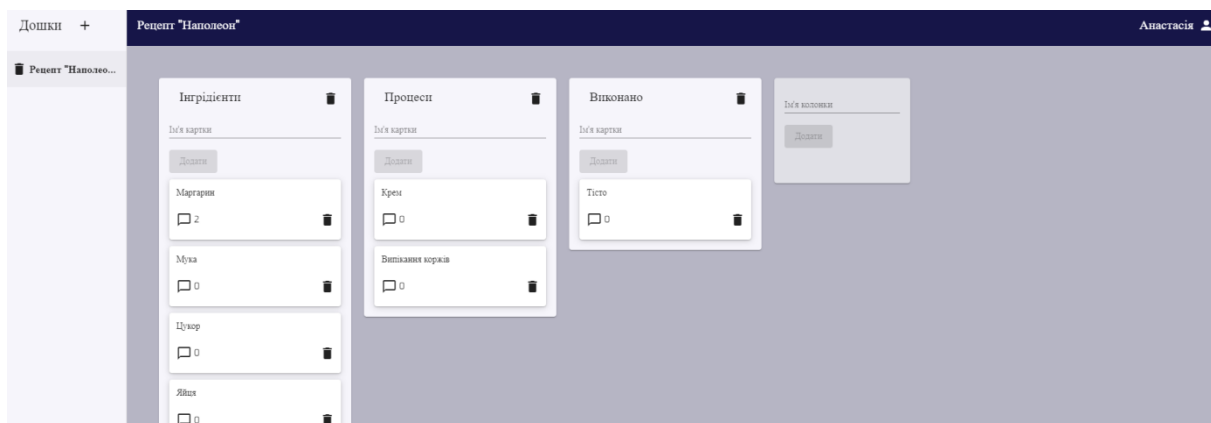


Рисунок 3.4 – Інтерфейс головної форми програмного забезпечення для управління задачами користувача

Інтерфейс форми картки завдання користувача наведено на рис. 3.5.

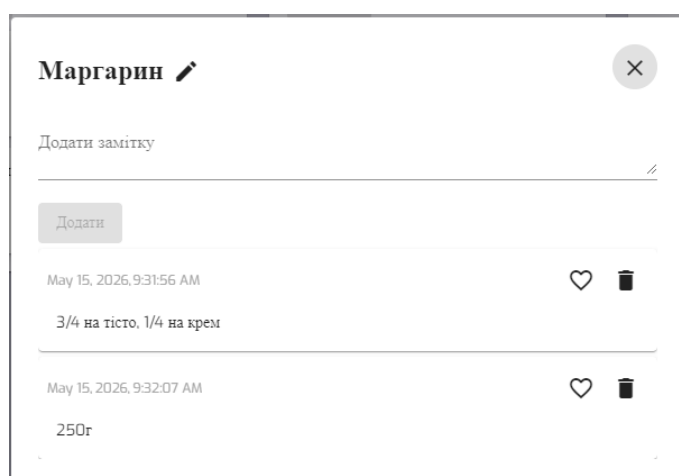


Рисунок 3.5 – Інтерфейс форми картки завдання користувача

Форму створення нової дошки наведено на рис. 3.6.

Ім'я дошки

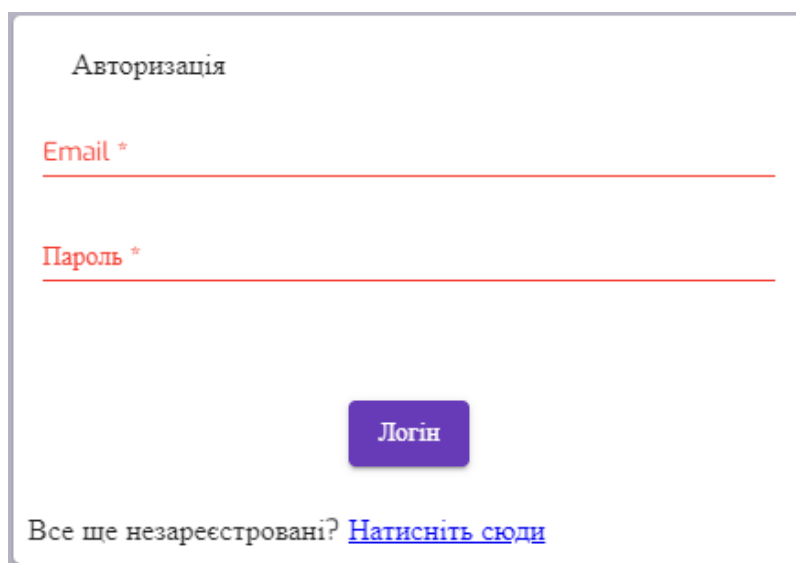
Максимум 40 символів

0/40

Створити дошку

Рисунок 3.6 – Форма створення нової дошки

Форму авторизації наведено на рис. 3.7.



Авторизація

Email *

Пароль *

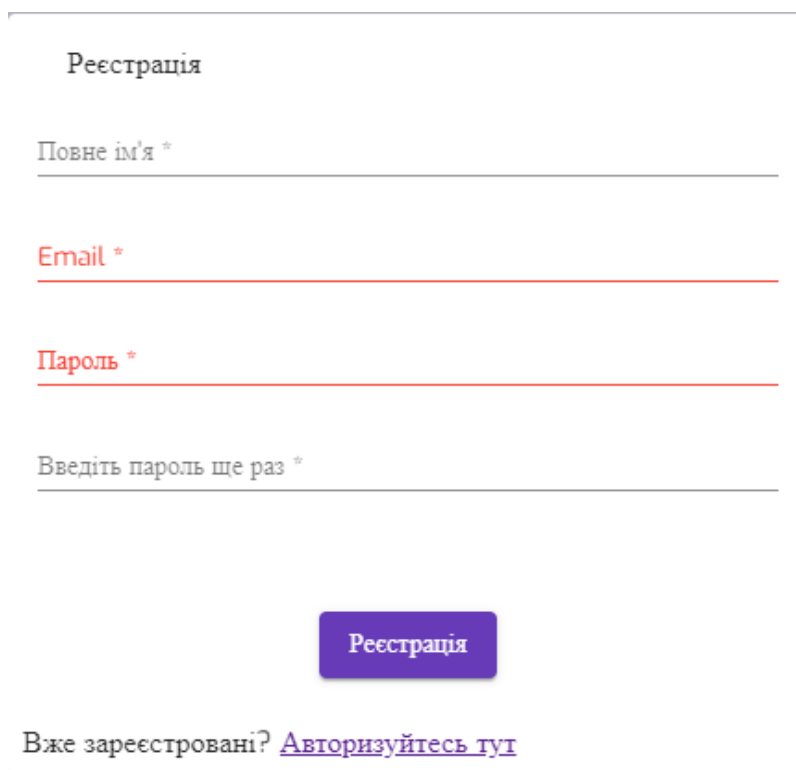
Логін

Все ще незареєстровані? [Натисніть сюди](#)

The image shows a web form for authorization. It has a title 'Авторизація' (Authorization). Below the title are two input fields: 'Email *' and 'Пароль *' (Password *). Below these fields is a purple button labeled 'Логін' (Login). At the bottom of the form, there is a link that says 'Все ще незареєстровані? Натисніть сюди' (Still not registered? Click here).

Рисунок 3.7 – Форма авторизації

Форму реєстрації наведено на рис. 3.8.



Реєстрація

Повне ім'я *

Email *

Пароль *

Введіть пароль ще раз *

Реєстрація

Вже зареєстровані? [Авторизуйтеся тут](#)

The image shows a web form for registration. It has a title 'Реєстрація' (Registration). Below the title are four input fields: 'Повне ім'я *' (Full name *), 'Email *', 'Пароль *' (Password *), and 'Введіть пароль ще раз *' (Enter password again *). Below these fields is a purple button labeled 'Реєстрація' (Registration). At the bottom of the form, there is a link that says 'Вже зареєстровані? Авторизуйтеся тут' (Already registered? Log in here).

Рисунок 3.8 – Форма реєстрації

3.5 Висновки за розділом 3

Розроблено програмне забезпечення для управління задачами користувача.

Запропоновано структуру програмного забезпечення для управління задачами користувача. Визначено, що структура програми побудована за модульним принципом, що забезпечує логічний поділ функціональності, спрощує розробку, тестування та подальший супровід системи. У межах такого підходу програмне забезпечення для управління задачами користувача складається з шести взаємопов'язаних модулів (модуль автентифікації та авторизації користувачів, модуль управління базою даних, модуль управління дошками, модуль управління колонками та картками, модуль взаємодії та динамічних змін, модуль користувацького інтерфейсу та представлення даних), кожен з яких виконує чітко визначені завдання та відповідає за окремий аспект функціонування вебзастосунку. Такий підхід створює умови для стабільної роботи системи, зручності її використання та можливості подальшого розширення функціональних можливостей.

Описано функціонування програмного забезпечення для управління задачами користувача. Послідовність функціонування програмного забезпечення для управління задачами користувача подано як логічно впорядкований ланцюг дій, що відображає взаємодію користувача з вебзастосунком та внутрішні процеси обробки даних.

Розроблено базу даних програмного забезпечення для управління задачами користувача, яка дозволяє виконувати зберігання та опрацювання структурованої інформації про користувачів, їхні робочі простори та елементи задач, а також забезпечує цілісність й узгодженість даних у процесі взаємодії з вебзастосунком. База даних програмного забезпечення для управління задачами користувача має ієрархічну, логічно впорядковану структуру, що відображає взаємозв'язки між користувачами, дошками, колонками, картками та замітками, що забезпечує надійне зберігання інформації, підтримку всіх

основних операцій над даними та можливість масштабування системи в процесі її подальшого розвитку.

Описано основні алгоритми програмного забезпечення для управління задачами користувача. Виконано проєктування інтерфейсу взаємодії користувача з програмним забезпеченням для управління задачами користувача.

4 ЕКСПЛУАТАЦІЯ, ТЕСТУВАННЯ ТА ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ ПРОГРАМИ

4.1 Призначення й умови застосування програми

Програма призначена для управління задачами користувача. Програма написана на мові JavaScript, що обумовлено її універсальністю, можливістю повноцінного використання як на стороні клієнта, так і на стороні сервера, а також тісною інтеграцією з вебтехнологіями. В якості серверного середовища виконання для програмного забезпечення для управління задачами обрано Node.js, що обумовлено його архітектурними особливостями та відповідністю вимогам сучасних клієнт-серверних і веборієнтованих систем.

Програмне забезпечення для управління задачами користувача забезпечує виконання таких функцій:

- підтримка можливості створення, перегляду, редагування та видалення дошок для організації задач користувача;
- підтримка можливості реєстрації та авторизації користувачів для забезпечення контролю доступу до персональних даних;
- можливість збереження інформації про користувачів, дошки, колонки, картки та замітки у централізованій базі даних;
- можливість можливості створення, редагування, переміщення та видалення карток завдань користувача;
- підтримка можливості додавання та перегляду заміток, пов'язаних із окремими картками;
- підтримка можливості сортування дошок відповідно до обраних параметрів відображення.

Для роботи програмного забезпечення для управління задачами користувача потрібні такі технічні та програмні ресурси: апаратна частина клієнтського пристрою повинна містити процесор з тактовою частотою не нижче 1,8 ГГц, що забезпечує коректну обробку інтерфейсних елементів та асинхронних запитів, а також оперативну пам'ять обсягом не менше 4 ГБ, що

є достатнім для стабільної роботи веббраузера разом з іншими фоновими процесами. Для відображення інтерфейсу необхідний дисплей з роздільною здатністю не нижче 1366×768 пікселів, а для взаємодії з серверною частиною обов'язковою є наявність стабільного підключення до мережі Інтернет зі швидкістю не менше 5 Мбіт/с.

Серверна частина програмного забезпечення повинна функціонувати на апаратному середовищі з процесором не нижче 2,0 ГГц, оперативною пам'яттю від 4 ГБ та вільним дисковим простором не менше 10 ГБ для зберігання бази даних, журналів подій та резервних копій. Як операційна система може використовуватися Linux або Windows Server, що підтримують серверні середовища виконання. Для коректної роботи серверної логіки необхідна наявність платформи Node.js актуальної LTS-версії, яка забезпечує виконання JavaScript-коду на сервері, а також система керування базами даних, наприклад PostgreSQL або MongoDB, що відповідає за надійне зберігання інформації про користувачів, дошки, колонки, картки та замітки.

З програмного боку клієнтський доступ до системи реалізується через веббраузер, такий як Google Chrome, Mozilla Firefox або Microsoft Edge, з увімкненою підтримкою JavaScript, HTML5 та CSS3. Для обміну даними між клієнтською та серверною частинами необхідна підтримка протоколу HTTP/HTTPS, що забезпечує безпечну передачу інформації.

Функціональні характеристики розробленого програмного забезпечення для управління задачами користувача наведено у технічному завданні (додаток А). Фрагмент тексту програмного забезпечення для управління задачами користувача наведено у додатку Б.

4.2 Характеристики програми для управління задачами користувача

Програмне забезпечення для управління задачами користувача характеризується орієнтацією на систематизацію робочих процесів і

підвищення керованості персональної або командної діяльності. Його функціонування спрямоване на впорядкування інформації шляхом логічного поділу задач за рівнями абстракції, що дозволяє користувачу формувати цілісне уявлення про поточний стан виконання робіт. Архітектура системи передбачає чітке розмежування відповідальності між компонентами, завдяки чому забезпечується узгоджена взаємодія інтерфейсної та серверної частин.

Властивістю такого програмного забезпечення є інтерактивність, що проявляється у миттєвому реагуванні системи на дії користувача та візуальному відображенні змін у структурі задач. Значна увага приділяється зручності сприйняття даних, оскільки інтерфейс проєктується з урахуванням принципів наочності та мінімізації когнітивного навантаження. Функціональна гнучкість дозволяє адаптувати систему до різних сценаріїв використання без потреби суттєвих змін у внутрішній логіці роботи.

Важливою характеристикою є надійність збереження інформації, яка досягається завдяки централізованому управлінню даними та контролю цілісності взаємопов'язаних сутностей. Система підтримує безперервність роботи навіть за умов зростання обсягу даних, що свідчить про її масштабованість та здатність до подальшого розвитку. Окрему роль відіграє безпека, яка реалізується через обмеження доступу до персональної інформації та розмежування прав користувачів.

Узагальнюючи, програмне забезпечення для управління задачами користувача вирізняється цілісністю, адаптивністю та орієнтацією на практичне застосування, що робить його ефективним інструментом організації діяльності в цифровому середовищі.

4.3 Інструкція по експлуатації програми

4.3.1 Звернення до програми

Для запуску програмного забезпечення для управління задачами користувача на серверній частині необхідно запустити відповідний файл. Для

звертання до клієнтської сторони необхідно перейти на відповідну вебсторінку.

4.3.2 Вхідні й вихідні дані

Вхідними даними до програмного забезпечення для управління задачами користувача є інформація, що вводиться під час взаємодії з інтерфейсом системи та використовується для формування і зміни внутрішнього стану застосунку. До таких даних належать персональні відомості, що подаються в процесі реєстрації та авторизації, а також текстові й структурні параметри, пов'язані зі створенням робочих просторів.

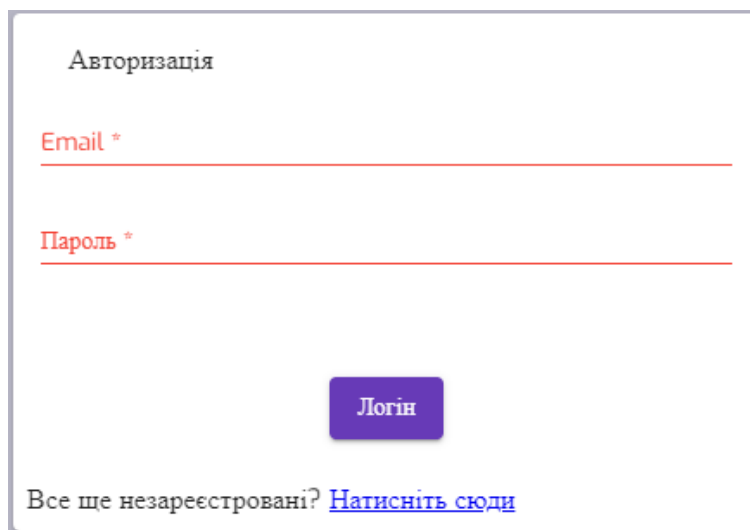
Вихідними даними програмного забезпечення для управління задачами користувача є впорядковане представлення збереженої та обробленої інформації, яке відображається у графічному інтерфейсі. Результатом роботи системи виступають актуальні дані про створені дошки, колонки, картки та пов'язані з ними замітки, що подаються у вигляді структурованих елементів.

4.3.3 Повідомлення

Користувач може отримати такі повідомлення: інформаційні повідомлення про успішне виконання дій, пов'язаних зі створенням, редагуванням або видаленням елементів задач, попередження щодо некоректно введених даних або відсутності необхідних прав доступу, а також системні сповіщення про помилки взаємодії з серверною частиною.

4.4 Виконання програмного забезпечення для управління задачами користувача

Після запуску програми користувачу відображається форма авторизації (рис. 4.1), де необхідно ввести свій пароль та електронну адресу.



Авторизація

Email *

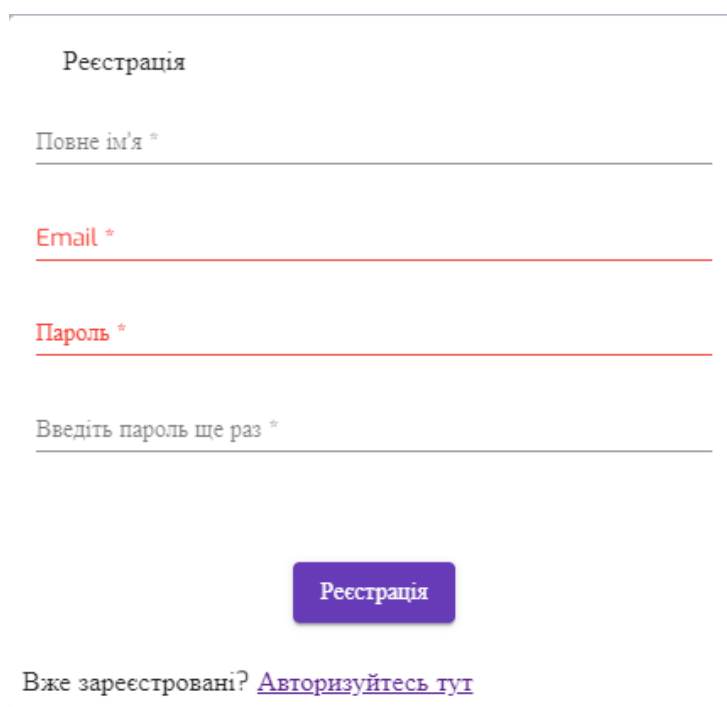
Пароль *

Логін

Все ще незареєстровані? [Натисніть сюди](#)

Рисунок 4.1 – Інтерфейс форми авторизації у програмному забезпеченні для управління задачами користувача

Незареєстрований користувач у формі авторизації повинен перейти за відповідним посиланням та зареєструватися (рис. 4.2).



Реєстрація

Повне ім'я *

Email *

Пароль *

Введіть пароль ще раз *

Реєстрація

Вже зареєстровані? [Авторизуйтеся тут](#)

Рисунок 4.2 – Інтерфейс форми реєстрації у програмному забезпеченні для управління задачами користувача

Після авторизації відображається головна сторінка (рис. 4.3) з дошками користувача, які відповідають його задачам. Користувач має можливість додавання, редагування та видалення дошок задач.

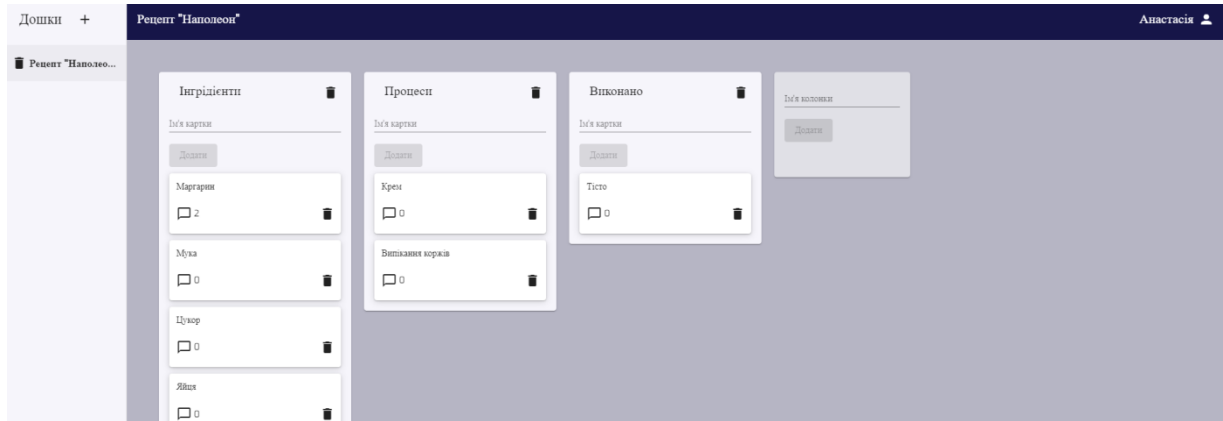


Рисунок 4.3 – Головна сторінка програмного забезпечення для управління задачами користувача

При натисканні на картку задачі відкривається форма (рис. 4.4), що відповідає обраній задачі. Користувач має можливість редагування задачі, або видалення окремих її елементів, поданих у вигляді заміток.

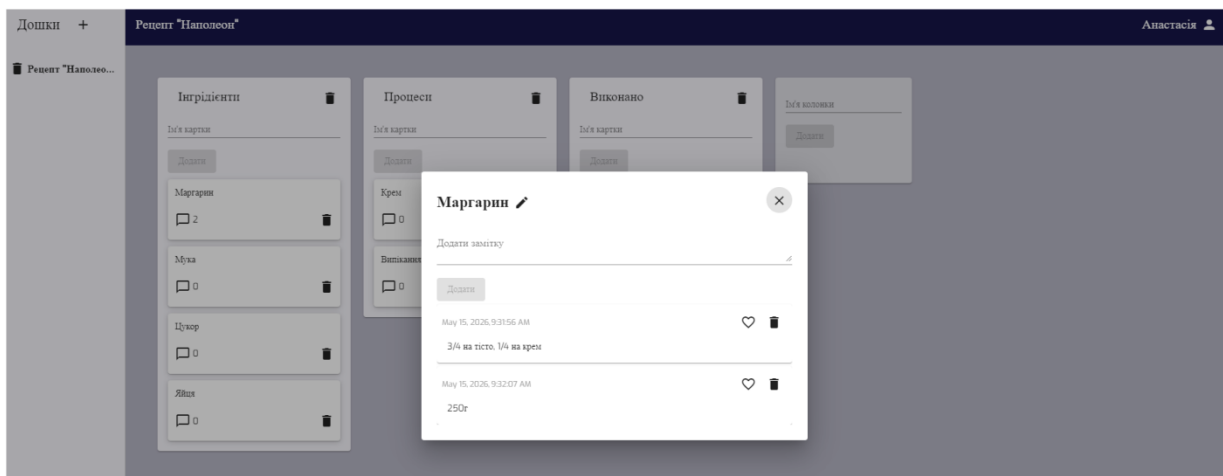


Рисунок 4.4 – Форма картки завдання користувача

Для додавання нової дошки задач необхідно натиснути на символ + у меню Дошки та ввести назву дошки (рис. 4.5).

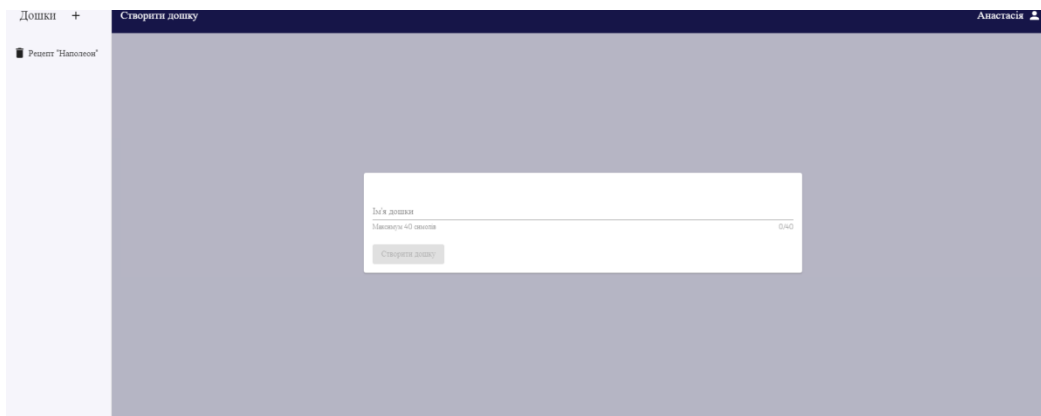


Рисунок 4.5 – Форма створення нової дошки

Для редагування профілю користувача (рис. 4.6) необхідно натиснути на його імені у правому верхньому куті сторінки та ввести, за необхідності, нове ім'я та електронну пошту.

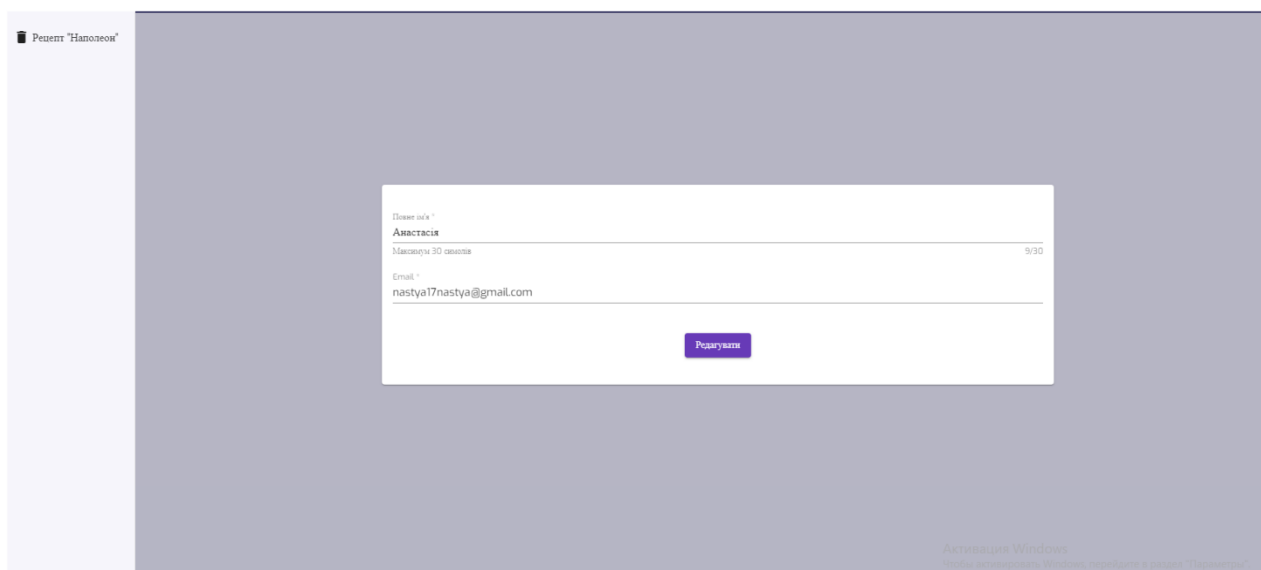


Рисунок 4.6 – Редагування профілю користувача

4.5 Тестування програмного забезпечення для управління задачами користувача

Виконано тестування програмного забезпечення для управління задачами користувача.

Виявлено, що програма для управління задачами користувача функціонує правильно та злагоджено. Вона реалізує всі функціональні вимоги та успішно виконує своє основне призначення управління задачами користувача.

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

4.6 Висновки за розділом 4

Описано програмне забезпечення для управління задачами користувача. Виконано тестування розробленого програмного забезпечення для управління задачами користувача. Результати тестування програмного забезпечення показали, що розроблена програма дозволяє забезпечити автоматизацію процесів, пов'язаних з підтримкою процесу управління задачами користувача.

ВИСНОВКИ

В ході виконання дипломної кваліфікаційної роботи бакалавра було проаналізовано та досліджено процес розробки програмного забезпечення для управління задачами користувача.

Визначено, що програмне забезпечення для управління задачами користувача представляє собою комплекс цифрових інструментів, призначених для організації, контролю та оптимізації процесів виконання завдань у рамках індивідуальної чи колективної діяльності. Такі програмні засоби забезпечують централізоване відстеження робочих процесів, дозволяють структурувати інформацію про завдання, терміни їх виконання та відповідальних осіб, а також надають можливість координувати діяльність між різними учасниками проєктів.

За результатами проведеного аналізу зроблено висновок, що у наш час існує досить багато програмних засобів для управління задачами користувача. Проте незважаючи на значні переваги, використання таких програмних систем не завжди забезпечує очікуваний результат та в окремих випадках створює додаткові ускладнення в процесі діяльності користувача. Одним із суттєвих недоліків є складність освоєння окремих програмних рішень. Надмірна функціональність, складний інтерфейс або недостатньо зрозуміла логіка роботи можуть ускладнювати процес адаптації користувачів, особливо за відсутності спеціальної підготовки. У таких випадках час, витрачений на навчання та налаштування системи, може перевищувати вигоди від її використання. Ще одним недоліком є ризик інформаційного перевантаження. Постійні сповіщення, нагадування та оновлення статусів здатні знижувати концентрацію уваги та спричиняти фрагментацію робочого процесу. Замість підвищення продуктивності це може призводити до зростання рівня стресу та відчуття постійної зайнятості без реального прогресу у виконанні завдань. Крім того, питання захисту персональних даних і конфіденційності інформації залишаються актуальними, особливо при використанні хмарних сервісів. Тому

актуальною є розробка програмного забезпечення для управління задачами користувача.

Сформульовано функціональні вимоги до програмного забезпечення для управління задачами користувача.

Для реалізації програмного забезпечення для управління задачами користувача обрано мову програмування JavaScript, що обумовлено її універсальністю, можливістю повноцінного використання як на стороні клієнта, так і на стороні сервера, а також тісною інтеграцією з вебтехнологіями. Крім того, JavaScript має розвинену екосистему бібліотек і фреймворків, які значно прискорюють розробку програмного забезпечення.

В якості серверного середовища виконання для програмного забезпечення для управління задачами обрано Node.js, що обумовлено його архітектурними особливостями та відповідністю вимогам клієнт-серверних і веборієнтованих систем. Крім того, Node.js побудований на асинхронній, подієво-орієнтованій моделі виконання, яка забезпечує ефективну обробку численних запитів без створення великої кількості потоків. Це дозволяє досягти високої продуктивності та масштабованості серверної частини системи, що є особливо актуальним для програмного забезпечення управління задачами, де користувачі часто змінюють стан завдань, отримують оновлення та працюють з даними в режимі реального часу.

Розроблено програмне забезпечення для управління задачами користувача.

Запропоновано структуру програмного забезпечення для управління задачами користувача. Визначено, що структура програми побудована за модульним принципом, що забезпечує логічний поділ функціональності, спрощує розробку, тестування та подальший супровід системи. У межах такого підходу програмне забезпечення для управління задачами користувача складається з шести взаємопов'язаних модулів (модуль автентифікації та авторизації користувачів, модуль управління базою даних, модуль управління дошками, модуль управління колонками та картками, модуль взаємодії та

динамічних змін, модуль користувацького інтерфейсу та представлення даних), кожен з яких виконує чітко визначені завдання та відповідає за окремий аспект функціонування вебзастосунку. Такий підхід створює умови для стабільної роботи системи, зручності її використання та можливості подальшого розширення функціональних можливостей.

Описано функціонування програмного забезпечення для управління задачами користувача. Послідовність функціонування програмного забезпечення для управління задачами користувача подано як логічно впорядкований ланцюг дій, що відображає взаємодію користувача з вебзастосунком та внутрішні процеси обробки даних.

Розроблено базу даних програмного забезпечення для управління задачами користувача, яка дозволяє виконувати зберігання та опрацювання структурованої інформації про користувачів, їхні робочі простори та елементи задач, а також забезпечує цілісність й узгодженість даних у процесі взаємодії з вебзастосунком. База даних програмного забезпечення для управління задачами користувача має ієрархічну, логічно впорядковану структуру, що відображає взаємозв'язки між користувачами, дошками, колонками, картками та замітками, що забезпечує надійне зберігання інформації, підтримку всіх основних операцій над даними та можливість масштабування системи в процесі її подальшого розвитку.

Описано основні алгоритми програмного забезпечення для управління задачами користувача. Виконано проєктування інтерфейсу взаємодії користувача з програмним забезпеченням для управління задачами користувача.

Виконано тестування розробленого програмного забезпечення для управління задачами користувача. Результати тестування програмного забезпечення показали, що розроблена програма дозволяє забезпечити автоматизацію процесів, пов'язаних з підтримкою процесу управління задачами користувача.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. What Is Task Management Software And How Can It Help You. URL: <https://peoplemanagingpeople.com/hr-operations/what-is-task-management-software/> (date of access: 17.04.2026).
2. How a Task Manager Helps Prioritize and Track Tasks Amid Deadline Chaos. URL: <https://intranet.smart-it.com/en/blog-post/what-is-task-manager/> (date of access: 17.04.2026).
3. What is Task Management Software and How Does It Differ from a Project Management Tool. URL: <https://www.techradar.com/pro/what-is-task-management-software-and-how-does-it-differ-from-a-project-management-tool> (date of access: 17.04.2026).
4. How to Manage Team Tasks: The Complete Guide. URL: <https://project.co/team-task-management/> (date of access: 17.04.2026).
5. What Is Task Management? Key Tools and More. URL: <https://www.coursera.org/articles/task-management> (date of access: 17.04.2026).
6. Best Task Management Software. URL: <https://thedigitalprojectmanager.com/tools/best-task-management-software/> (date of access: 17.04.2026).
7. Top 10 best task management software. URL: <https://www.retable.io/blog/top-10-best-task-management-software-in-2025-free-and-paid-tools> (date of access: 17.04.2026).
8. The Best Task Management Apps. URL: <https://www.pcmag.com/picks/the-best-task-management-apps> (date of access: 17.04.2026).
9. Best Free Task Management Software. URL: <https://www.softwaresuggest.com/task-management-software/free> (date of access: 17.04.2026).

10. Best Task Management Software To Keep You on Track. URL: <https://www.websiteplanet.com/blog/best-task-management-software/> (date of access: 17.04.2026).
11. Best Task Management Software for Teams. URL: <https://www.founderjar.com/best-task-management-software/> (date of access: 17.04.2026).
12. Nifty. URL: <https://niftypm.com/> (date of access: 17.04.2026).
13. MeisterTask. URL: <https://www.meistertask.com/> (date of access: 17.04.2026).
14. FunctionFox. URL: <https://functionfox.com/> (date of access: 17.04.2026).
15. JavaScript Syntax. URL: https://www.w3schools.com/Js/js_syntax.asp (date of access: 17.04.2026).
16. JavaScript Guide. URL: <https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide> (date of access: 17.04.2026).
17. How Node.js Works. URL: <https://www.geeksforgeeks.org/node-js/how-node-js-works/> (date of access: 17.04.2026).
18. Introduction to Node.js. URL: <https://nodejs.org/en/learn/getting-started/introduction-to-nodejs> (date of access: 17.04.2026).

ДОДАТОК А
Технічне завдання

Вступ

Програмне забезпечення може використовуватися для управління задачами користувача.

А.1 Підстава для розробки

Підставою для розробки є завдання на дипломну кваліфікаційну роботу на тему «Програмне забезпечення для управління задачами користувача», затверджене наказом Національного університету «Запорізька політехніка» № 139 від 7 квітня 2026 р.

А.2 Призначення розробки

Програмний продукт призначений для керування задачами користувача.

А.3 Основні вимоги до програми, що розробляється

А.3.1 Вимоги до функціональних характеристик

Програмне забезпечення для управління задачами користувача забезпечує виконання таких функцій:

- підтримка можливості створення, перегляду, редагування та видалення дошок для організації задач користувача;
- підтримка можливості реєстрації та авторизації користувачів для забезпечення контролю доступу до персональних даних;
- можливість збереження інформації про користувачів, дошки, колонки, картки та замітки у централізованій базі даних;
- можливість можливості створення, редагування, переміщення та видалення карток завдань користувача;

- підтримка можливості додавання та перегляду заміток, пов'язаних із окремими картками;
- підтримка можливості сортування дошок відповідно до обраних параметрів відображення.

А.3.2 Вимоги до інтерфейсу програми

Інтерфейс програмного забезпечення для управління задачами користувача повинен бути зручним для користувачів. Повинна бути забезпечена можливість роботи в візуальному режимі.

А.3.3 Вимоги до надійності

Програмне забезпечення для управління задачами користувача повинно забезпечити надійне функціонування.

А.3.4 Умови експлуатації

Для експлуатації програмного забезпечення для управління задачами користувача необхідна наявність персонального комп'ютера.

А.3.5 Вимоги до складу та параметрів технічний засобів

Для роботи програмного забезпечення для управління задачами користувача потрібні такі технічні та програмні ресурси: апаратна частина клієнтського пристрою повинна містити процесор з тактовою частотою не нижче 1,8 ГГц, що забезпечує коректну обробку інтерфейсних елементів та асинхронних запитів, а також оперативну пам'ять обсягом не менше 4 ГБ, що

є достатнім для стабільної роботи сучасного веббраузера разом з іншими фоновими процесами. Для відображення інтерфейсу необхідний дисплей з роздільною здатністю не нижче 1366×768 пікселів, а для взаємодії з серверною частиною обов'язковою є наявність стабільного підключення до мережі Інтернет зі швидкістю не менше 5 Мбіт/с.

Серверна частина програмного забезпечення повинна функціонувати на апаратному середовищі з процесором не нижче 2,0 ГГц, оперативною пам'яттю від 4 ГБ та вільним дисковим простором не менше 10 ГБ для зберігання бази даних, журналів подій та резервних копій. Як операційна система може використовуватися Linux або Windows Server, що підтримують сучасні серверні середовища виконання. Для коректної роботи серверної логіки необхідна наявність платформи Node.js актуальної LTS-версії, яка забезпечує виконання JavaScript-коду на сервері, а також система керування базами даних, наприклад PostgreSQL або MongoDB, що відповідає за надійне зберігання інформації про користувачів, дошки, колонки, картки та замітки.

З програмного боку клієнтський доступ до системи реалізується через веббраузер, такий як Google Chrome, Mozilla Firefox або Microsoft Edge, з увімкненою підтримкою JavaScript, HTML5 та CSS3. Для обміну даними між клієнтською та серверною частинами необхідна підтримка протоколу HTTP/HTTPS, що забезпечує безпечну передачу інформації.

А.3.6 Вимоги до маркування і пакування

Програма для управління задачами користувача може бути записана на будь-якому носії інформації.

На пакуванні повинна бути назва програми – «Програмне забезпечення для управління задачами користувача».

A.4 Вхідні дані до роботи

Вхідними даними до програмного забезпечення для управління задачами користувача є інформація, що вводиться під час взаємодії з інтерфейсом системи та використовується для формування і зміни внутрішнього стану застосунку. До таких даних належать персональні відомості, що подаються в процесі реєстрації та авторизації, а також текстові й структурні параметри, пов'язані зі створенням робочих просторів.

Вихідними даними програмного забезпечення для управління задачами користувача є впорядковане представлення збереженої та обробленої інформації, яке відображається у графічному інтерфейсі. Результатом роботи системи виступають актуальні дані про створені дошки, колонки, картки та пов'язані з ними замітки, що подаються у вигляді структурованих елементів.

ДОДАТОК Б
Фрагмент тексту програми

```

export function crtStt() {
  let usr = null
  let brds = []
  let actvBrd = null
  let actvClmn = null
  let actvCrd = null

  function rndrLgn() {
    const frm = document.createElement("div")
    const nptNm = document.createElement("input")
    const nptMl = document.createElement("input")
    const btn = document.createElement("button")
    btn.textContent = "Lgn"
    btn.onclick = () => {
      str.crtUsr(nptNm.value, nptMl.value)
      rndr()
    }
    frm.appendChild(nptNm)
    frm.appendChild(nptMl)
    frm.appendChild(btn)
    rt.appendChild(frm)
  }

  function rndrBrds() {
    const btn = document.createElement("button")
    btn.textContent = "Nw Brd"
    btn.onclick = () => {
      str.crtBrd("Brd")
      rndr()
    }
    rt.appendChild(btn)
    stt.gtBrds().forEach(b => {
      const d = document.createElement("div")
      d.textContent = b.ttl
      d.onclick = () => {
        stt.stActvBrd(b.d)
        rndrClmns(b)
      }
      rt.appendChild(d)
    })
  }

  function rndrClmns(brd) {
    rt.innerHTML = ""
    const btn = document.createElement("button")
    btn.textContent = "Nw Clmn"
    btn.onclick = () => {
      str.crtClmn(brd.d, "Clmn")
    }
  }

```

```

rndrClmns (brd)
}
rt.appendChild(btn)
brd.clmns.forEach(c => {
  const d = document.createElement("div")
  d.textContent = c.ttl
  d.onclick = () => {
    stt.stActvClmn(c.d)
    rndrCrds(brd, c)
  }
  rt.appendChild(d)
})
}

function rndrCrds(brd, clmn) {
  rt.innerHTML = ""
  const btn = document.createElement("button")
  btn.textContent = "Nw Crd"
  btn.onclick = () => {
    str.crtCrd(brd.d, clmn.d, "Crd")
    rndrCrds(brd, clmn)
  }
  rt.appendChild(btn)
  clmn.crds.forEach(c => {
    const d = document.createElement("div")
    d.textContent = c.ttl
    d.onclick = () => {
      stt.stActvCrd(c.d)
      rndrNts(brd, clmn, c)
    }
    rt.appendChild(d)
  })
}

function rndrNts(brd, clmn, crd) {
  rt.innerHTML = ""
  const btn = document.createElement("button")
  btn.textContent = "Nw Nt"
  btn.onclick = () => {
    str.addNt(brd.d, clmn.d, crd.d, "Nt")
    rndrNts(brd, clmn, crd)
  }
  rt.appendChild(btn)
  crd.nts.forEach(n => {
    const d = document.createElement("div")
    d.textContent = n.txt
    rt.appendChild(d)
  })
}

```

```

}

function stUsr(vl) {
  usr = vl
}

function gtUsr() {
  return usr
}

function addBrd(brd) {
  brds = brds.concat([brd])
}

function rmvBrd(d) {
  brds = brds.filter(b => b.d !== d)
}

function gtBrds() {
  return brds
}

function stActvBrd(d) {
  actvBrd = d
}

function gtActvBrd() {
  return actvBrd
}

function stActvClmn(d) {
  actvClmn = d
}

function gtActvClmn() {
  return actvClmn
}

function stActvCrd(d) {
  actvCrd = d
}

function gtActvCrd() {
  return actvCrd
}

return {
  stUsr,

```

```

gtUsr,
addBrd,
rmvBrd,
gtBrds,
stActvBrd,
gtActvBrd,
stActvClmn,
gtActvClmn,
stActvCrd,
gtActvCrd
}
}

export function crtStr(stt) {
function crtUsr(nm, ml) {
stt.stUsr({ nm, ml })
}

function crtBrd(ttl) {
const brd = { d: Math.random().toString(), ttl, clmns: [] }
stt.addBrd(brd)
}

function crtClmn(brdD, ttl) {
const brds = stt.gtBrds()
const brd = brds.find(b => b.d === brdD)
if (brd) {
brd.clmns.push({ d: Math.random().toString(), ttl, crds: [] })
}
}

function crtCrd(brdD, clmnD, ttl) {
const brds = stt.gtBrds()
const brd = brds.find(b => b.d === brdD)
if (brd) {
const clmn = brd.clmns.find(c => c.d === clmnD)
if (clmn) {
clmn.crds.push({ d: Math.random().toString(), ttl, nts: [] })
}
}
}

function mvCrd(brdD, frmClmnD, tClmnD, crdD) {
const brds = stt.gtBrds()
const brd = brds.find(b => b.d === brdD)
if (brd) {
const frm = brd.clmns.find(c => c.d === frmClmnD)
const t = brd.clmns.find(c => c.d === tClmnD)
if (frm && t) {

```

```

const crd = frm.crds.find(c => c.d === crdD)
frm.crds = frm.crds.filter(c => c.d !== crdD)
t.crds.push(crd)
}
}
}

function addNt(brdD, clmnD, crdD, txt) {
const brds = stt.gtBrds()
const brd = brds.find(b => b.d === brdD)
if (brd) {
const clmn = brd.clmns.find(c => c.d === clmnD)
if (clmn) {
const crd = clmn.crds.find(c => c.d === crdD)
if (crd) {
crd.nts.push({ d: Math.random().toString(), txt })
}
}
}
}

export function crtTkn(usrD) {
let hdr = btoa(JSON.stringify({ tp: "JWT", alg: "HS" }))
let pld = btoa(JSON.stringify({ d: usrD, tm: Date.now() }))
let sgn = btoa(hdr + "." + pld)
return hdr + "." + pld + "." + sgn
}

export function stTkn(tkn) {
document.cookie = "tkn=" + tkn + "; path=/"
}

export function gtTkn() {
let mtc = document.cookie.match(/tkn=([^\;]+)/)
return mtc ? mtc[1] : null
}

export function clrTkn() {
document.cookie = "tkn=; path=/; expires=Thu, 01 Jan 1970 00:00:00 UTC"
}

export async function sndLgn(d) {
let rsp = await fetch("/api/lgn", {
mthd: "POST",
hdrs: { "Cntnt-Tp": "ppplctn/jsn" },
bdy: JSON.stringify(d)
})
let rs = await rsp.json()

```

```

if (rs.tkn) {
  stTkn(rs.tkn)
}
return rs
}

export async function sndRgs(d) {
  let rsp = await fetch("/api/rgs", {
    mthd: "POST",
    hdrs: { "Cntnt-Tp": "ppplctn/jsn" },
    bdy: JSON.stringify(d)
  })
  let rs = await rsp.json()
  return rs
}

export async function gtUsrFrmTkn() {
  let tkn = gtTkn()
  if (!tkn) {
    return null
  }
  let rsp = await fetch("/api/usr", {
    hdrs: { "thrzt": "Brr " + tkn }
  })
  let rs = await rsp.json()
  return rs
}

export async function dbGtBrds() {
  let tkn = gtTkn()
  let rsp = await fetch("/api/brds", {
    hdrs: { "thrzt": "Brr " + tkn }
  })
  let rs = await rsp.json()
  return rs
}

export async function dbCrtBrd(d) {
  let tkn = gtTkn()
  let rsp = await fetch("/api/brds", {
    mthd: "POST",
    hdrs: { "Cntnt-Tp": "ppplctn/jsn", "thrzt": "Brr " + tkn },
    bdy: JSON.stringify(d)
  })
  let rs = await rsp.json()
  return rs
}

export async function dbCrtClmn(brdD, d) {

```

```

let tkn = gtTkn()
let rsp = await fetch("/api/brds/" + brdD + "/clmns", {
mthd: "POST",
hdrs: { "Cntnt-Tp": "ppplctn/jsn", "thrzt": "Brr " + tkn },
bdy: JSON.stringify(d)
})
let rs = await rsp.json()
return rs
}

```

```

export async function dbCrtCrd(clmnD, d) {
let tkn = gtTkn()
let rsp = await fetch("/api/clmns/" + clmnD + "/crds", {
mthd: "POST",
hdrs: { "Cntnt-Tp": "ppplctn/jsn", "thrzt": "Brr " + tkn },
bdy: JSON.stringify(d)
})
let rs = await rsp.json()
return rs
}

```

```

export async function dbMvCrd(crdD, d) {
let tkn = gtTkn()
let rsp = await fetch("/api/crds/" + crdD + "/mv", {
mthd: "PUT",
hdrs: { "Cntnt-Tp": "ppplctn/jsn", "thrzt": "Brr " + tkn },
bdy: JSON.stringify(d)
})
let rs = await rsp.json()
return rs
}

```

```

export async function dbAddNt(crdD, d) {
let tkn = gtTkn()
let rsp = await fetch("/api/crds/" + crdD + "/nts", {
mthd: "POST",
hdrs: { "Cntnt-Tp": "ppplctn/jsn", "thrzt": "Brr " + tkn },
bdy: JSON.stringify(d)
})
let rs = await rsp.json()
return rs
}

```

```

export async function dbDltBrd(brdD) {
let tkn = gtTkn()
let rsp = await fetch("/api/brds/" + brdD, {
mthd: "DLT",
hdrs: { "thrzt": "Brr " + tkn }
})
}

```

```
let rs = await rsp.json()
return rs
}

export function vrfTkn() {
let tkn = gtTkn()
if (!tkn) {
return false
}
let prts = tkn.split(".")
if (prts.length !== 3) {
clrTkn()
return false
}
return true
}

export function atRfrsh(fn) {
window.addEventListener("focus", () => {
if (vrfTkn()) {
fn()
}
})
}
```

ДОДАТОК В
Слайди презентації

Міністерство освіти і науки України
Національний університет “Запорізька політехніка”
Кафедра програмних засобів

Програмне забезпечення для управління задачами користувача

Виконала: Анастасія НОСКОВА

ст. групи КНТ-113сп

Керівник: Михайло КОЦУР

к.т.н., доцент НУ «Запорізька політехніка»

Рисунок В.1 – Слайд 1

2

Об’єкт, предмет та мета роботи

Об’єкт дослідження – процес розробки програмного забезпечення для управління задачами користувача.

Предмет дослідження – програмні засоби для управління задачами користувача.

Метою роботи є розробка програмного забезпечення для управління задачами користувача, що забезпечує зменшення часу на планування та виконання завдань.

Рисунок В.2 – Слайд 2

Завдання роботи

- Для досягнення поставленої мети у кваліфікаційній роботі бакалавра необхідно розв'язати такі задачі:
- виконати аналіз предметної області та програмних засобів для управління задачами користувача;
 - здійснити проєктування програмного забезпечення для управління задачами користувача;
 - створити програмне забезпечення для управління задачами користувача;
 - виконати тестування розробленого програмного забезпечення для управління задачами користувача.

Рисунок В.3 – Слайд 3

Порівняння існуючих аналогів

Критерій порівняння	Nifty	MeisterTask	FunctionFox
Інтуїтивний інтерфейс	+	+	+—
Візуальне планування	+	+—	+—
Відстеження часу	+—	+	+
Аналіз продуктивності та звітність	+—	+	+—
Персоналізація робочого простору	+	+—	+—
Інтеграції із зовнішніми сервісами	+	+	—

Рисунок В.4 – Слайд 4

Вибір мови програмування

Критерій порівняння мов програмування	Мова програмування		
	JavaScript	C#	PHP
Підтримка інтерактивних вебінтерфейсів	+	+-	-
Можливість використання на клієнті і сервері	+	-	-
Кросплатформність	+	+	+-
Швидкість розробки вебзастосунків	+	+-	+-
Простота інтеграції з зовнішніми сервісами	+	+-	+-

Рисунок В.5 – Слайд 5

Вибір середовища виконання

Критерій порівняння середовищ виконання	Середовища виконання		
	Node.js	Deno	Cloudflare Workers
Зрілість та стабільність платформи	+	+-	+-
Поширеність у промисловій розробці	+	+-	+-
Екосистема пакетів і бібліотек	+	+-	-
Масштабованість для багатокористувацьких систем	+	+-	+
Простота розгортання та хостингу	+	+-	+-

Рисунок В.6 – Слайд 6

Структура програми

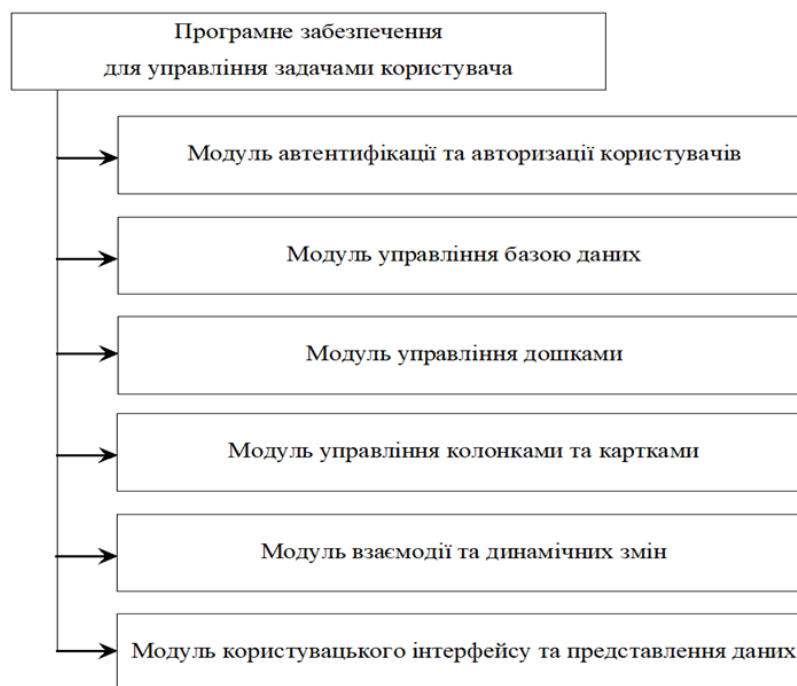


Рисунок В.7 – Слайд 7

Схема функціонування

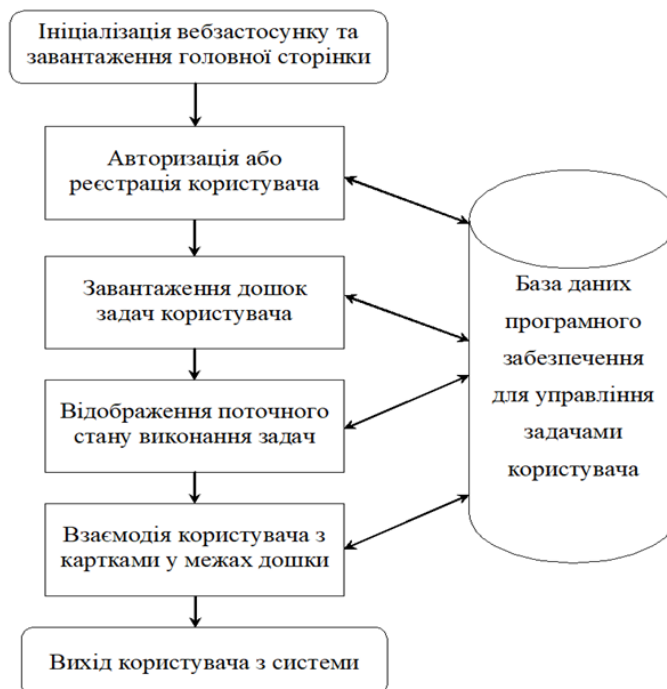


Рисунок В.8 – Слайд 8

Модель бази даних ПЗ для управління задачами користувача

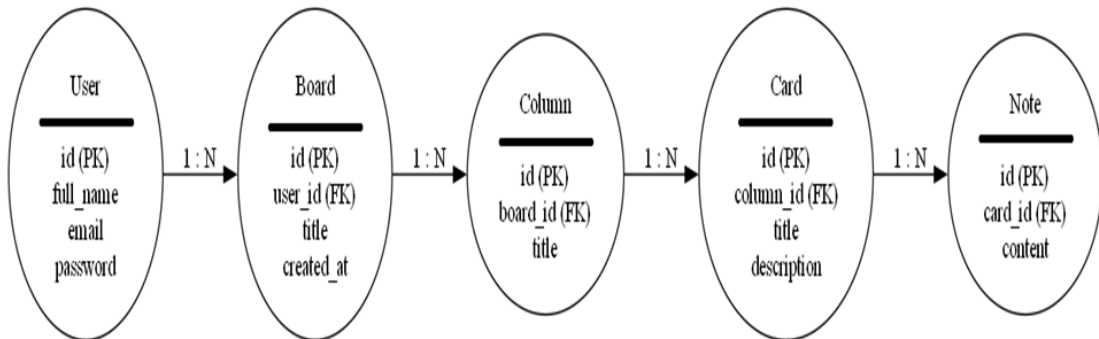
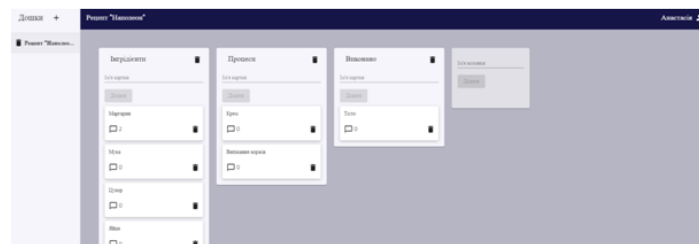


Рисунок В.9 – Слайд 9

Робота з програмою. Головна сторінка. Авторизація та реєстрація



Авторизація

Email *

Пароль *

Логін

Все ще незареєстровані?

Натисніть сюди

Реєстрація

Повне ім'я *

Email *

Пароль *

Введіть пароль ще раз *

Реєстрація

Вже зареєстровані?

Авторизуйтеся тут

Рисунок В.10 – Слайд 10

Робота з програмою. Робота з завданнями та профілем користувача

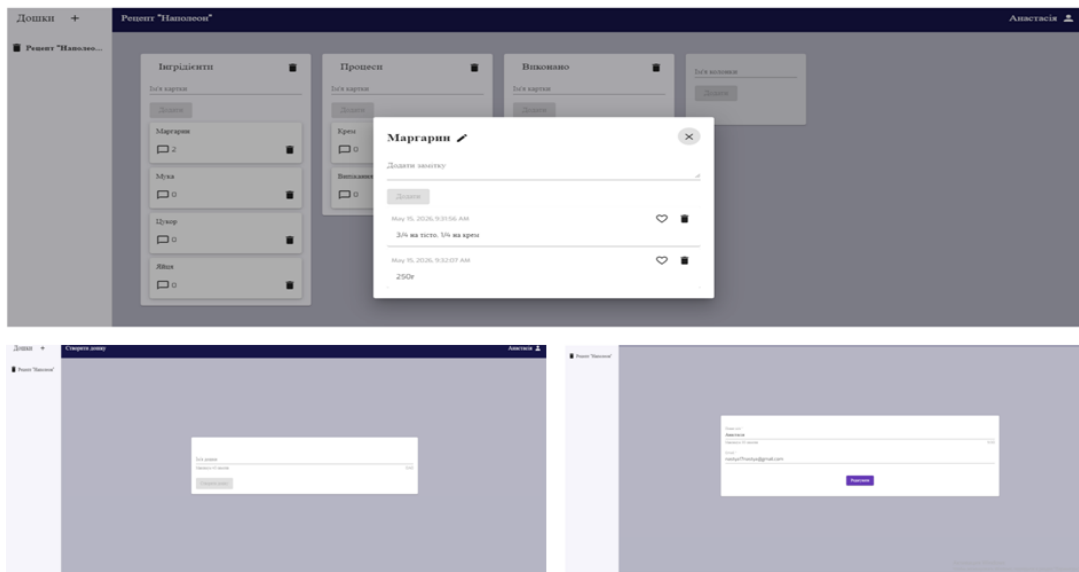


Рисунок В.11 – Слайд 11

Висновки

В ході виконання дипломної кваліфікаційної роботи бакалавра було проаналізовано та досліджено процес розробки програмного забезпечення для управління задачами користувача.

Для реалізації програмного забезпечення для управління задачами користувача обрано мову програмування JavaScript та серверне середовище виконання Node.js, що обумовлено його архітектурними особливостями та відповідністю вимогам клієнт-серверних і веборієнтованих систем.

Розроблено програмне забезпечення для управління задачами користувача. Запропоновано структуру програмного забезпечення для управління задачами користувача. Визначено, що структура програми побудована за модульним принципом, що забезпечує логічний поділ функціональності, спрощує розробку, тестування та подальший супровід системи. Розроблено базу даних програмного забезпечення для управління задачами користувача.

В результаті виконання завдання було отримано програмну систему, що відповідає всім вимогам представленим в технічному завданні, зручна для використання та виконує всі необхідні функції.

Усі завдання випускної дипломної роботи бакалавра повністю виконано.

Рисунок В.12 – Слайд 12