МІНІСТЕРСТВО ОСВІТИ І НАУКИ

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ВОДНОГО ГОСПОДАРСТВА ТА ПРИРОДОКОРИСТУВАННЯ

«До захисту допущена»
Зав. кафедри комп'ютерних наук
та прикладної математики
д.т.н., професор Турбал Ю.В.
«____» _____ 20__ р.

КВАЛІФІКАЦІЙНА РОБОТА

Розробка вебдодатку для управління командними проектами

Виконав:Прокопюк Олександр Сергійович,
студент навчально-наукового інституту автоматики, кібернетики та обчислювальної техніки
група КН-41

_____
(підпис)

Керівник: к.т.н., доцент кафедри комп'ютерних наук та прикладної математики Демчук О.С.

_____
(підпис)

Рівне 2023

Національний університет водного господарства та природокористування

НН інститут**автоматики, кібернетики та обчислювальної техніки**

Кафедра **комп'ютерних наук таприкладної математики**

Освітньо-кваліфікаційний рівень **бакалавр**

Галузь знань 12 «Інформаційні технології»

Спеціальність 122 «Комп'ютерні науки»

Спеціалізація _____

ЗАТВЕРДЖУЮ

Завідувач кафедри

_____

_____

"____" _____20___року

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ
Прокопюку Олександру Сергійовичу

1. Тема роботи РОЗРОБКА ВЕБДОДАТКУ ДЛЯ УПРАВЛІННЯ КОМАНДНИМИ ПРОЕКТАМИ
керівник роботи к.т.н., доцент кафедрикомп'ютерних наук та прикладної математики Демчук О.С.
затверджені наказом вищого навчального закладу від «19» квітня 2023 року С №-449
2. Термін подання роботи студентом 31.05.2023
3. Вихідні дані до роботи технології, що необхідні для розробки веб-додатку для управління командними проєктами
4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) РОЗДІЛ 1. ТЕОРЕТИЧНІ ВІДОМОСТІ ПРО УПРАВЛІННЯ КОМАНДНИМИ ПРОЄКТАМИ. ОГЛЯД АНАЛОГІВ. РОЗДІЛ 2. ПРОЄКТУВАННЯ ТА РОЗРОБКА ВЕБ-ДОДАТКУ ДЛЯ ОРГАНІЗАЦІЇ КОМАНДНОЇ РОБОТИ. РОЗДІЛ 3. ІНТЕРФЕЙС КОРИСТУВАЧА ВЕБ-ДОДАТКУ. ІНСТРУКЦІЯ ДЛЯ ВИКОРИСТАННЯ

5. Перелік графічного матеріалу (з точним зазначення обов'язкових креслень) мультимедійна презентація.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Розділ 1	к.т.н., доцент кафедри прикладної математики Демчук О.С.	20.12.2022	20.12.2022
Розділ 2	к.т.н., доцент кафедри прикладної математики Демчук О.С.	05.01.2023	05.01.2023
Розділ 3	к.т.н., доцент кафедри прикладної математики Демчук О.С.	24.04.2023	24.04.2023

7. Дата видачі завдання _____

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Вивчення літератури за тематикою роботи	01.10.2022 – 09.01.2023	виконав
2	Формулювання задачі	10.01.2023 – 15.01.2023	виконав
3	Аналіз аналогів програмної розробки.	16.01.2023 – 19.01.2023	виконав
4	Вибір стеку технологій	20.01.2023 – 22.01.2023	виконав
5	Збір та аналіз даних	23.01.2023 – 10.02.2023	виконав
6	Розробка структури веб-додатку	11.02.2023 – 04.04.2023	виконав
7	Розробка користувацького інтерфейсу	05.04.2023 – 20.04.2023	виконав
8	Відлагодження програмного продукту	21.04.2023 – 13.05.2023	виконав
9	Підготовка звіту	14.05.2023 – 31.05.2023	виконав
10	Підготовка мультимедійної презентації, виступу	05.06.2023 – 24.06.2023	виконав

Студент_____Прокопюк О.С.

(підпис) (прізвище та ініціали)

Керівник роботи_____Демчук О.С._____

(підпис) (прізвище та ініціали)

РЕФЕРАТ

Кваліфікаційна робота: 75 с., 23 малюнки, 1 таблиця, 20 джерел.

Метою кваліфікаційної роботи є аналіз сучасних програмних рішень для командної співпраці та розробка власного веб-додатку спільної роботи над проєктами.

Об’єкт дослідження – процес організації та візуалізації завдань на дошках Kanban за допомогою веб-платформи.

Предмет дослідження – сучасні методи та інформаційні технології організації робочого процесу команди для досягнення безперебійної синхронізації даних.

Методи дослідження –методологія Kanban для організації командної роботи над проєктами, JSON, Node.js, CouchDB.

Проведено аналіз сучасних програмних рішень для організації командної роботи та описано основні методології, які використовуються у сучасних розробках. Розроблено власний додаток, що реалізує централізовану платформу дляорганізації роботи команди. Розроблений в рамках даної кваліфікаційної роботи програмний продукт дозволяєпідвищити продуктивність роботи команди та покращити їх результати за допомогою наочного підходу до розподілення задач та комунікації між учасниками процесу.

Ключові слова: КОМАНДНА РОБОТА, ВЕБ-САЙТ, REACT, COUCHDB.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

ПЗ – програмне забезпечення

БД – база даних

СКБД – система керування базами даних

JSON – JavaScript Object Notation

DOM – Document Object Model

API – Application Programming Interface

ЗМІСТ

ВСТУП

Розробка веб-сайтів для управління командними проектами є дуже актуальною в сучасному світі, де віддалена робота та розподілені команди стають все більш поширеними. З розвитком глобалізації та зростанням популярності віддаленої роботи все більше компаній звертаються до онлайн-інструментів управління проєктами, щоб допомогти своїм командам ефективно співпрацювати.

Веб-сайт для командного управління проєктами може забезпечити централізовану платформу для спілкування членів команди, відстеження прогресу та спільної роботи над проектами. Це також може допомогти впорядкувати робочі процеси, зменшити кількість непорозумінь і підвищити продуктивність.

Більше того, після пандемії COVID-19 багато компаній були змушені впровадити віддалену роботу та інструменти онлайн співпраці, щоб підтримувати безперервність бізнесу. Тому веб-сайт для управління командними проектами важливий як ніколи, щоб забезпечити ефективну та результативну роботу віддалених команд в компаніях та організаціях будь-якого розміру.

Метою дослідження є вивчення необхідних інформаційних технологій та розробка веб-платформи, яка полегшить командне управління проєктами..

Конкретні цілі дослідження можуть включати:

- Визначення ключових функцій, необхідних для ефективного управління командними проектами, та включення їх у веб-платформу.
- Вивчення існуючих інструментів управління командними проєктами та виявлення недоліків, які може усунути запропонована платформа.
- Розробка зручного інтерфейсу, зрозумілого для всіх членів команди і з можливістю легкої навігації.

- Проведення тестування зручності використання, щоб переконатися, що платформа є інтуїтивно зрозумілою та відповідає потребам користувачів.
- Впровадження заходів безпеки для забезпечення захисту конфіденційної інформації проекту.
- Оцінка ефективності платформи для покращення командної комунікації, співпраці та продуктивності.
- Надання рекомендацій щодо подальшого розвитку та вдосконалення платформи.

Об'єктом дослідження є процес розробки веб-сайту для командного управління проєктами. Це включає планування, проєктування, реалізацію, тестування та розгортання веб-сайту, а також його постійне обслуговування та підтримку.

Предметом дослідження є сам веб-сайт, який слугує платформою для управління командними проектами. Сюди входять функції і можливості веб-сайту, такі як відстеження завдань, командна комунікація та інструменти для спільної роботи, а також користувацький інтерфейс і дизайн.В дослідженні також необхідно оцінити ефективність вебсайту в підвищенні продуктивності команди та покращенні результатів проєкту, а також будь-які виклики чи обмеження, що виникають під час процесу розробки.

РОЗДІЛ 1. ТЕОРЕТИЧНІ ВІДОМОСТІ ПРО УПРАВЛІННЯ КОМАНДНИМИ ПРОЄКТАМИ. ОГЛЯД АНАЛОГІВ

1.1. Поняття команди та командної роботи

Поняття "команда" і "командна робота" означає колективні зусилля групи людей, які працюють разом для досягнення спільної мети або завдання. Команда – це група людей, які поділяють спільну ціль і прагнуть працювати разом для її досягнення. Командна робота – це процес спільної роботи з іншими для досягнення спільної мети, що часто передбачає координацію індивідуальних навичок і зусиль.

Ефективні колективи характеризуються тісною комунікацією, довірою та спільним відчуттям мети. Члени команди повинні вміти ефективно спілкуватися один з одним, обмінюватися ідеями та надавати зворотний зв'язок. Довіра також має важливе значення для ефективної роботи, оскільки члени команди повинні мати можливість покладатися один на одного у виконанні своїх зобов'язань і підтримувати один одного, коли це необхідно.

На додаток до міцної комунікації та довіри, ефективна спільна робота також вимагає спільного відчуття мети. Члени команди повинні мати чітке розуміння своїх цілей і завдань, а також відчуття того, як їхні індивідуальні внески вписуються в загальну картину. Таке спільне відчуття мети може допомогти спрямувати індивідуальні зусилля на досягнення спільної цілі, а також сприяти формуванню почуття товариськості та згуртованості серед товариства.

Загалом, концепція команди та командної роботи важлива в багатьох сферах життя, зокрема на роботі, у спорті та соціальному середовищі. Ефективні команди можуть досягти більшого, ніж окремі люди, що працюють поодинці, а процес спільної роботи з іншими для досягнення спільної мети може бути корисним і захопливим[1].

1.2. Основні етапи управління командними проєктами

Основні етапи управління проєктом зазвичай включають в себе наступні пункти:

1. Ініціалізація. На цьому етапі визначаються цілі, масштаби та обмеження проєкту. Сюди входить створення плану, який окреслює цілі, результати, терміни, бюджет і ресурси, необхідні для завершення проєкту.
2. Планування. Цей етап передбачає розробку детального плану проєкту, який окреслює завдання, терміни і ресурси, необхідні для його завершення. Сюди входить створення графіка, визначення етапів проєкту, а також визначення потенційних ризиків і способів їх зниження.
3. Виконання. Цей етап передбачає реалізацію плану проєкту, включаючи призначення завдань членам команди, моніторинг прогресу та управління змінами до плану за необхідності. Сюди також входить управління бюджетом проєкту, ресурсами та очікуваннями зацікавлених сторін.
4. Моніторинг і контроль. Цей етап передбачає відстеження прогресу проєкту відповідно до плану та внесення будь-яких необхідних коректив, щоб проєкт не відставав від графіка. Він включає моніторинг результатів, виявлення потенційних проблем і вжиття коригувальних заходів у разі потреби.
5. Закриття. Цей етап передбачає завершення проєкту, включаючи надання кінцевого продукту або послуги замовнику, оцінку успіху та документування всіх отриманих уроків для майбутнього.

Ці етапи можуть відрізнятися залежно від конкретного проєкту та методології управління, що використовується. Однак, загалом, вони забезпечують основу для планування, виконання та закриття проєкту, щоб

забезпечити його успішне завершення з дотриманням обмежень щодо обсягу, бюджету та часових рамок[2].

1.3. Методики управління командними проєктами

Ефективне управління проєктами має важливе значення для забезпечення успіху будь-якої ідеї, незалежно від сфери чи галузі. Управління командним проєктом передбачає координацію різних завдань, управління ресурсами та забезпечення завершення роботи вчасно і в рамках бюджету. Однією з ключових проблем управління проєктами є збалансування конкуруючих вимог, таких як час, вартість, якість та обсяг.

Для подолання цих викликів протягом багатьох років були розроблені та використовуються різні методи управління проєктами. Ці заходи покликані допомогти командам організовувати та виконувати великі обсяги завдань ефективно, результативно та з мінімальним стресом. У цьому контексті розглянемо деякі з найпопулярніших методів командного управління проєктами.

Водоспадна модель

Метод водоспаду – це підхід до управління проєктами, який передбачає послідовний, лінійний процес. У цьому методі проєкт ділиться на окремі фази або етапи, і кожна фаза повинна бути завершена, перш ніж команда зможе перейти до наступної. П'ять фаз методу водоспаду такі:

- Збір вимог. Це перша фаза, на якій визначаються і збираються вимоги до проєкту. Команда працює з клієнтом або зацікавленими сторонами, щоб визначити обсяг, цілі, результати і терміни.
- Проєктування. На цьому етапі команда створює детальний дизайн і план проєкту. Це включає визначення технічних специфікацій, створення графіка та визначення необхідних ресурсів.

- Реалізація. На цьому етапі відбувається фактична розробка проєкту. Команда слідує плану розробки, створеному на попередньому етапі, і починає створювати проєкт. Ця фаза зазвичай включає кодування, тестування та інтеграцію.
- Тестування. Після завершення фази реалізації команда переходить до фази тестування. Тут вона виконує ряд тестів, щоб переконатися, що проєкт функціонує належним чином. Це включає модульне тестування, інтеграційне тестування та системне тестування.
- Розгортання. Заключним етапом методу водоспаду є розгортання. Це коли проєкт передається клієнту або зацікавленим сторонам. Ця фаза включає в себе встановлення продукту в системі клієнта, навчання та підтримку, а також забезпечення безперебійної роботи.

Однією з ключових переваг методу водоспаду є те, що він забезпечує чітку структуру та графік проєкту. Він також гарантує, що кожна фаза буде завершена до того, як команда перейде до наступної, що може допомогти зменшити кількість помилок і гарантувати, що кінцевий продукт відповідає вимогам. Однак метод водоспаду також може бути негнучким, оскільки може бути складно вносити зміни після завершення фази. Крім того, лінійна природа методу водоспаду може ускладнити реагування на зміни у вимогах або непередбачувані проблеми, що виникають під час проєкту [4].

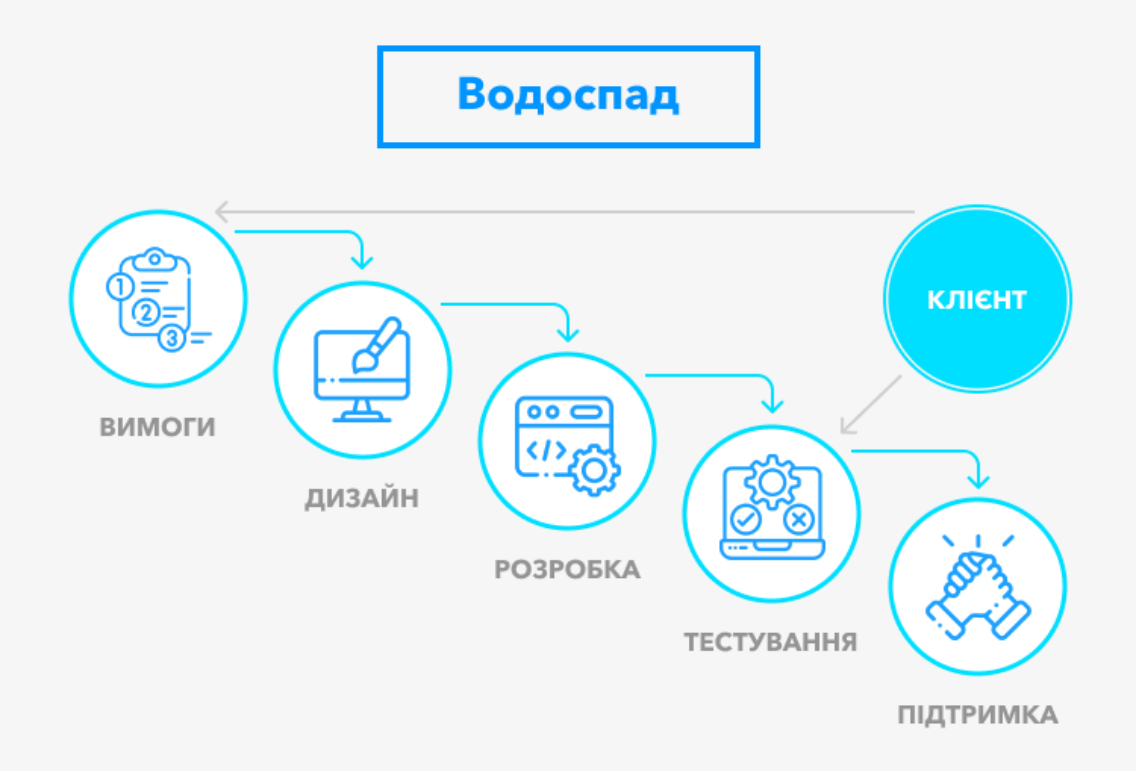

Рис. 1.1. Структура процесу “Водоспад”

Згідно рисунку 1.1, цей метод має свої переваги, такі як простота та логічність процесу, низькі витрати на управління проєктом, а також можливість точної оцінки вартості та термінів виконання. Однак, існують певні недоліки, такі як відсутність гнучкості та неможливість внесення змін на пізніших етапах, що може обмежувати отримання кращих результатів. Крім того, продукт тестується лише наприкінці, а не на кожному етапі та їхній взаємодії [3].

Agile модель

Метод Agile – це підхід до управління проєктами, який підкреслює гнучкість, співпрацю та швидке реагування на зміни. Цей метод є ітеративним, тобто проєкт розбивається на менші, керовані компоненти, які називаються спринтами, і кожен спринт призначений для поступового створення робочого продукту. Метод Agile слідує принципам, викладеним в Маніфесті Agile, який ставить на перше місце задоволення потреб клієнтів,

безперервну доставку робочого програмного забезпечення та адаптивність до мінливих вимог.

Метод Agile покликаний допомогти командам швидко реагувати на зміни у вимогах або ринкових умовах, а також постійно вдосконалювати проєкт по мірі його реалізації. Метод Agile включає в себе наступні ключові компоненти:

- Беклог продукту. Це пріоритетний список функцій та вимог до проєкту. Беклог продукту постійно оновлюється протягом усього проєкту, коли додаються або змінюються нові вимоги.
- Спринт-планування. На початку кожного спринту команда проводить зустріч з планування, щоб визначити цілі спринту та завдання, які необхідно виконати. Колектив працює спільно, щоб оцінити час і зусилля, необхідні для виконання кожного завдання.
- Щоденні наради. Команда проводить короткі щоденні зустрічі, щоб проаналізувати прогрес, обговорити будь-які виклики та перешкоди, а також спланувати наступний день.
- Підбиття підсумків спринту. Наприкінці кожного спринту команда проводить оглядову зустріч, щоб продемонструвати робочий продукт і отримати зворотній зв'язок від зацікавлених сторін.

Однією з ключових переваг Agile-методу є те, що він дозволяє часто отримувати зворотній зв'язок і співпрацювати зі всіма зацікавленими сторонами, що може допомогти гарантувати, що проєкт рухається за графіком і відповідає очікуванням. Метод Agile також дозволяє швидко реагувати на зміни, оскільки проєкт розбивається на менші компоненти, які можуть бути скориговані за потреби. Однак, метод Agile може бути складним для деяких команд, оскільки вимагає високого рівня співпраці, комунікації та адаптивності. Він також вимагає високого рівня дисципліни, щоб

гарантувати, що команда не відстає від графіка і регулярно надає робоче програмне забезпечення [5].

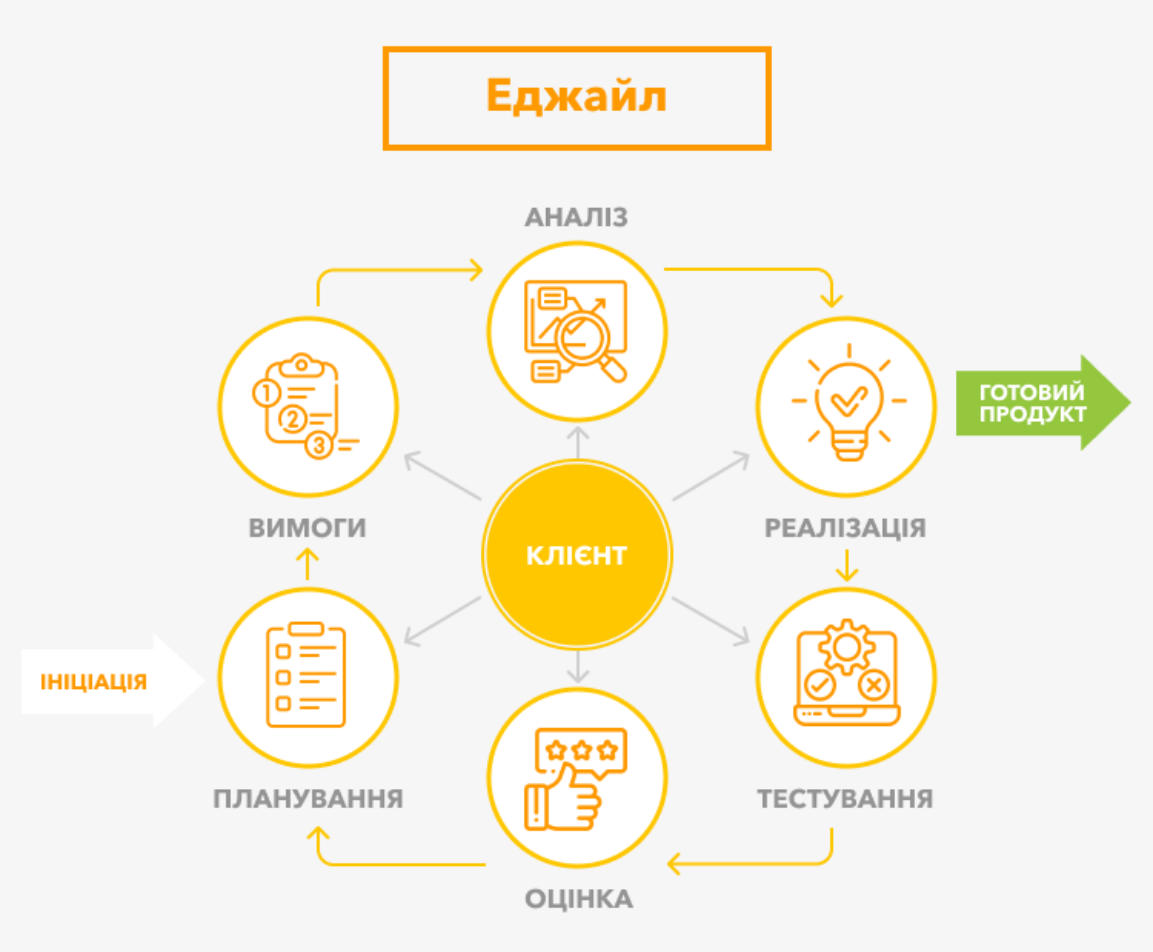

Рис. 1.2. Структура процесу Agile

Згідно рисунку 1.2, цей підхід дозволяє етапам проєкту працювати паралельно чи в будь-якому порядку, а не обов'язково по послідовності. Проте, на кожному етапі продукт має відповідати вимогам та працювати на певному рівні якості. Агіль має переваги у забезпеченні якісної взаємодії між учасниками проєкту та гнучкості. Цей підхід підходить для компаній, які операціонують в умовах швидких та непередбачуваних змін, у високотехнологічній сфері або при роботі з клієнтами, які постійно змінюють вимоги. Однак серед недоліків можна виділити той факт, що постійна адаптація до змін може призвести до того, що фінальний продукт ніколи не буде готовий [3].

Scrum модель

Метод Scrum – це особливий підхід в рамках гнучкого управління проєктами Agile. Скрам покликаний допомогти командам керувати складними проєктами, розбиваючи їх на менші, більш керовані завдання. Підхід наголошує на співпраці, гнучкості та безперервному циклі зворотного зв'язку. Метод Scrum включає в себе наступні ключові компоненти:

- Скрам фреймворк. Структура Scrum складається з набору ролей, аргументів і подій, які визначають процес управління проєктом. Фреймворк включає такі ролі: власник продукту, скрам-майстер і команда розробників.
- Беклог продукту.Це пріоритетний список функцій і вимог до проєкту. Беклог продукту постійно оновлюється впродовж роботи, коли додаються або змінюються нові вимоги.
- Планування спринту. На початку кожного спринту команда проводить зустріч з планування, щоб визначити цілі спринту та завдання, які необхідно виконати. Колектив працює спільно, щоб оцінити час і зусилля, необхідні для виконання кожного завдання.
- Щоденний скрам. Команда проводить короткі щоденні зустрічі, щоб переглянути прогрес, обговорити будь-які виклики або перешкоди та спланувати наступний день.
- Підбиття підсумків спринту. Наприкінці кожного спринту команда проводить оглядову зустріч, щоб продемонструвати робочий продукт і отримати зворотній зв'язок від зацікавлених сторін.
- Ретроспектива спринту. Після кожного спринту команда проводить ретроспективну зустріч, щоб проаналізувати його та визначити сфери для вдосконалення.

Однією з ключових переваг методу Scrum є те, що він забезпечує структурований процес управління складними проєктами. Цей підхід робить

акцент на співпраці та комунікації, що може допомогти гарантувати, що план не відстає від графіка і відповідає потребам зацікавлених сторін. Метод Scrum також дозволяє швидко реагувати на зміни, оскільки робота розбивається на менші компоненти, які можуть бути скориговані за потреби. Однак метод Scrum може бути складним для деяких команд, оскільки вимагає високого рівня дисципліни та відданості процесу. Він також вимагає сильного акценту на командній роботі та співпраці, оскільки успіх проєкту залежить від здатності ефективно працювати разом [6].

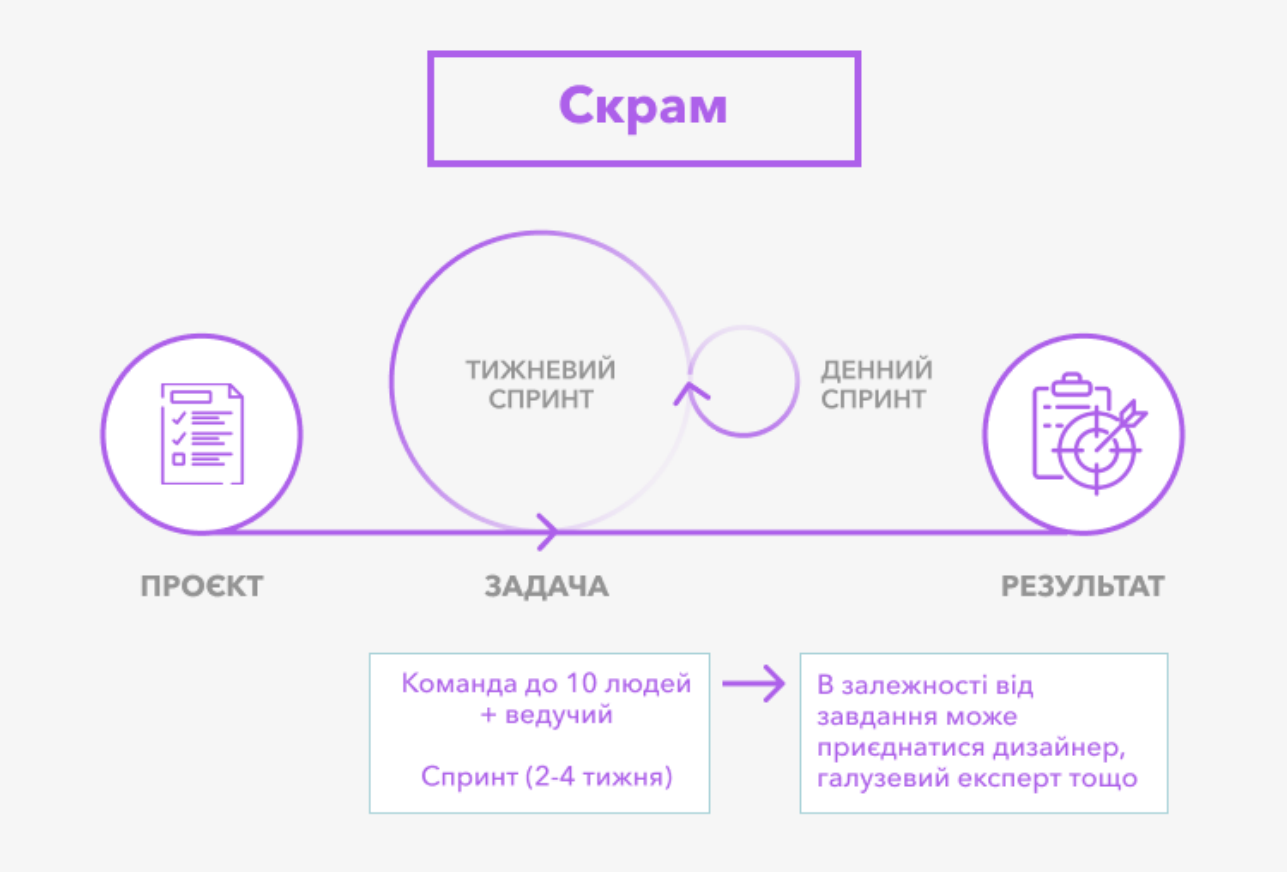

Рис. 1.3. Структура процесу Scrum

Згідно рисунку 1.3, процес розпочинається зі збору та аналізу вимог, які допомагають сформулювати задачі для відставання продукту. Для пріоритезації елементів відставання продукту враховуються залежності, а елементи з вищим пріоритетом реалізуються першими. Розробка системи включає ітерації, які зазвичай займають від 2 до 4 тижнів. Початок кожного ітераційного циклу (спринту) передбачає розподіл задач між розробниками

та їх подальше виконання. По завершенні спринту клієнт проводить огляд результатів, оцінює їх відповідність поставленим цілям та проводить навчальну сесію з метою поліпшення практики роботи. У цей час програмне забезпечення повинно бути протестоване та готове до випуску [3].

Kanban модель

Метод Kanban – це підхід до управління проєктами, який фокусується на візуалізації робочого процесу, обмеженні незавершеного виробництва та максимізації ефективності. Kanban виник у виробничій галузі, але згодом був адаптований для використання в розробці програмного забезпечення та інших галузях, що базуються на знаннях. В основі підходу лежить набір принципів і практик, покликаних допомогти командам більш ефективно управляти своєю роботою.

Метод Kanban включає в себе наступні ключові компоненти:

- Канбан-дошка. Дошка Канбан – це візуальне представлення робочого процесу, що зазвичай складається зі стовпчиків, які відображають різні етапи процесу. Кожне завдання представлене карткою, яка переміщується по стовпчиках у міру просування по робочому процесу.
- Кількість незавершеного продукту (WIP). Канбан наголошує на обмеженні кількості незавершеної роботи в будь-який момент часу. Це допомагає запобігти виникненню проблем і гарантує, що завдання будуть виконані вчасно.
- Постійне вдосконалення. Канбан заохочує команди до постійного вдосконалення своїх процесів і робочих операцій. Це може включати в себе регулярний перегляд дошки і визначення областей для покращення.
- Система "витягування". Канбан базується на системі "витягування", що означає, що завдання витягуються з робочого процесу в міру того, як

стають доступними ресурси. Це допомагає забезпечити ефективне використання ресурсів і своєчасне завершення роботи.

– Візуальні сигнали. Канбан використовує візуальні сигнали, такі як картки або наклейки, для представлення завдань і позначення їхнього статусу. Це допомагає зробити робочий процес більш наочним і зрозумілим.

Однією з ключових переваг методу Канбан є те, що він забезпечує простий і гнучкий спосіб управління роботою. Підхід легко зрозуміти і його можна адаптувати до різних робочих процесів і типів проєктів. Канбан також наголошує на постійному вдосконаленні, що може допомогти командам виявляти та вирішувати проблеми в міру їх виникнення. Однак, для деяких команд Kanban може бути складним, оскільки вимагає високого рівня дисципліни та сильної зосередженості на робочому процесі. Для його ефективного впровадження також можуть знадобитися значні зміни в існуючих процесах і робочих схемах [7].

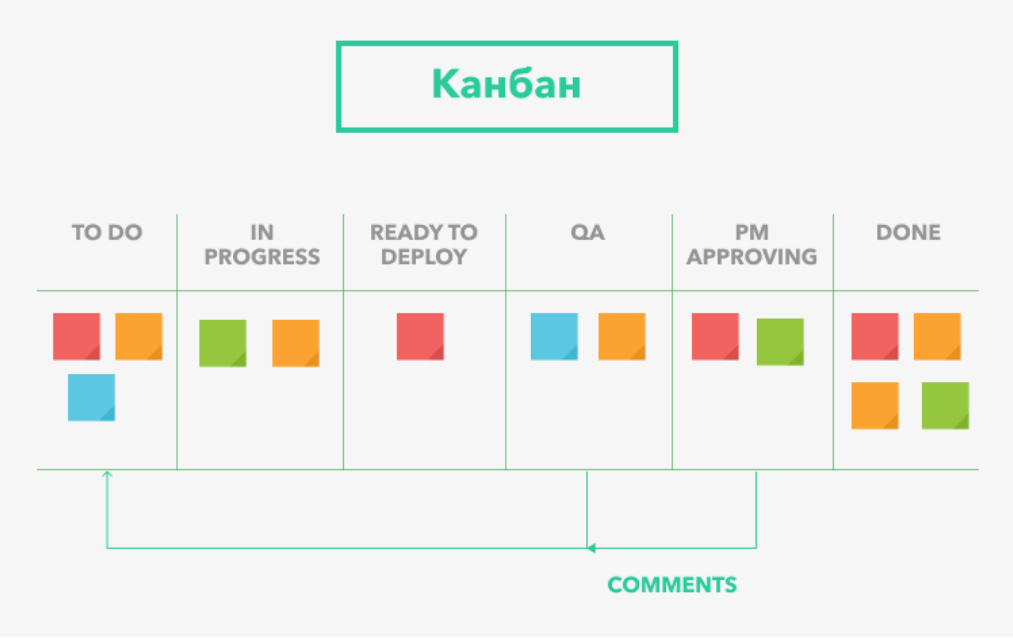

Рис. 1.4. Структура процесу Канбан

Рисунок 1.4 відображає суть Канбан-моделі. На ньому видно, що завдання, які потрібно виконати, відображаються на дошці. Кожне завдання

може мати різний статус: "To Do" (ще не розпочато), "In Progress" (виконується), або "Done" (виконано). Завдання можуть бути розбиті на менші частини (наприклад, "To Do", "In Progress", "QA", "Done"), які можуть бути відображені на окремих стовпчиках дошки. Канбан-модель передбачає постійне оновлення інформації на дошці, щоб відображати актуальний статус завдань, та постійну увагу до незавершених завдань, які потребують додаткових ресурсів для завершення. Це дозволяє здійснювати контроль над термінами виконання та переміщенням завдань, що сприяє постійному удосконаленню процесів [3].

Lean модель

Метод Lean – це підхід до управління проєктами, який фокусується на максимізації цінності при мінімізації втрат. В основі підходу лежить набір принципів і практик, які виникли в обробній промисловості, але згодом були адаптовані для використання в інших галузях, включаючи розробку програмного забезпечення та охорону здоров'я.

Метод ощадливого виробництва включає в себе наступні ключові компоненти:

- Цінність. Lean фокусується на створенні цінності для клієнтів. Це вимагає глибокого розуміння потреб клієнтів і готовності адаптувати проєкт для задоволення цих потреб.
- Витрати. Lean наголошує на необхідності мінімізації відхилень у всіх аспектах проекту. Це може включати втрату часу, ресурсів або матеріалів.
- Постійне вдосконалення. Lean заохочує команди постійно вдосконалювати свої процеси та робочі процеси. Це може включати регулярний перегляд проєкту та визначення сфер для вдосконалення.
- Система "витягування". Як і Канбан, Lean базується на системі витягування, що означає, що робота витягується з робочого процесу в

міру того, як стають доступними ресурси. Це допомагає забезпечити ефективне використання ресурсів і своєчасне завершення роботи.

– Кайдзен. Кайдзен – це ключовий принцип ощадливого виробництва, який наголошує на постійному вдосконаленні. Це може включати в себе невеликі, поступові поліпшення проєкту або процесу.

Однією з ключових переваг методу ощадливого виробництва є те, що він забезпечує системний підхід до зменшення відходів і максимізації цінності. Цей принцип наголошує на глибокому розумінні потреб замовника та готовності адаптувати проєкт для задоволення цих потреб. Lean також сприяє постійному вдосконаленню, що може допомогти командам виявляти та вирішувати проблеми в міру їх виникнення. Однак для деяких команд Lean може бути складним викликом, оскільки вимагає високого рівня дисципліни та сильного фокусу на зменшенні браку. Крім того, для його ефективного впровадження можуть знадобитися значні зміни в існуючих процесах і робочих циклах [8].

Рис. 1.5. Принципи Lean-методу

1.4. Огляд популярних платформ для управління командними проєктами

Успіх будь-якого проєкту значною мірою залежить від ефективної командної співпраці та управління. У сучасному динамічному середовищі платформи для командного управління проєктами стали незамінними інструментами для впорядкування робочих процесів, полегшення комунікації та забезпечення успіху проєкту. У цьому розділі буде представлено поглиблений аналіз популярних платформ, таких як Asana, Trello, Microsoft Teams, Notion та інших, з акцентом на їхніх особливостях, функціональних можливостях, перевагах та недоліках. Аналіз має на меті забезпечити комплексне розуміння цих платформ, їх придатність для різних розмірів команд і типів проєктів, а також їх потенційний вплив на продуктивність команди та результати проєкту.

1.4.1. Asana

Asana – це популярнеПЗ для управління командними проєктами, яка пропонує широкий спектр можливостей і функцій, призначених для оптимізації робочих процесів і покращення командної співпраці. Деякі з ключових функцій Asana включають:

1. Управління завданнями та проектами. Asana дозволяє користувачам створювати завдання, призначати їх членам команди, встановлювати терміни виконання та відстежувати прогрес. Користувачі також можуть створювати проєкти, додавати до них завдання та організовувати завдання за різними категоріями, що полегшує управління складними проєктами.
2. Взаємодія та комунікація. Asana надає платформу для спільної роботи членів команди над завданнями та проєктами в режимі реального часу. Користувачі можуть залишати коментарі до завдань, прикріплювати файли та позначати членів команди для конкретних обговорень. Це

сприяє прозорості та забезпечує ефективну комунікацію між членами команди.

3. Перегляд календаря та часової шкали. Asana пропонує календар та часові шкали, які забезпечують візуальне представлення завдань та проєктів. Це допомагає членам команди краще розуміти часові рамки, залежності та дедлайни проєктів, що сприяє ефективному плануванню та складанню графіків.

Переваги використання Asana для командного управління проєктами наступні:

- Зручний інтерфейс. Asana має інтуїтивно зрозумілий і зручний інтерфейс, що дозволяє членам команди легко орієнтуватися і використовувати платформу, навіть тим, хто має обмежені технічні навички.
- Масштабованість. Asana можна використовувати для управління проєктами будь-якого розміру, від невеликих команд до великих підприємств, що робить її універсальною для різних організаційних потреб.
- Співпраця та прозорість. Asana сприяє командній співпраці та прозорості, надаючи центральний модуль для управління завданнями, проєктами та командною комунікацією, що може підвищити продуктивність та прозорість.
- Кастомізація. Asana пропонує можливості кастомізації, такі як створення власних полів, шаблонів і правил, що дозволяє командам адаптувати платформу до своїх конкретних процесів управління проєктами та вимог.

Недоліки використання Asana для командного управління проєктами наступні:

- Ціна. Хоча Asana пропонує безкоштовну базову версію, розширені функції, такі як перегляд часової шкали, кастомні поля та інтеграції вимагають платної підписки, що може бути обмеженням для команд з невеликим бюджетом.
- Процес навчання. Хоча Asana має зручний інтерфейс, членам команди може знадобитися деякий час, щоб ознайомитися з усіма можливостями та функціями платформи, що може вплинути на початкову продуктивність.
- Складність для великих проектів. Інтерфейс Asana може стати надто складним для великих, комплексних проєктів з багатьма завданнями та залежностями, що вимагатиме додаткових зусиль для налаштування та управління.
- Обмежені можливості звітності. Функції звітності Asana є відносно базовими, що може не відповідати потребам деяких команд або організацій у розширених звітах.

Отже, Asana пропонує надійну платформу для командного управління проєктами з такими функціями, як керування завданнями та проєктами, співпраця та комунікація, перегляд календаря та часових шкал, а також інтеграція з іншими інструментами. Вона має переваги з точки зору зручного інтерфейсу, масштабованості, співпраці та можливостей кастомізації. Однак, можуть бути обмеження, пов'язані з ціноутворенням, тривалістю навчання, складністю для великих проєктів та обмеженими можливостями звітування. Ретельна оцінка можливостей та обмежень Asana необхідна, щоб визначити її придатність для конкретних потреб управління командними проєктами[9].

1.4.2. Trello

Trello – це популярна платформа для управління командними проєктами, яка використовує дошки, списки та картки для візуальної

організації завдань і проєктів. Проведемо детальний аналіз Trello, включаючи його можливості, функціонал, переваги та недоліки.

Можливості та функціонал:

1. Дошки, списки та картки. Trello використовує візуальний інтерфейс з дошками, що представляють проєкти, списками, що представляють етапи або категорії, і картками, що представляють завдання або елементи. Користувачі можуть створювати, переміщати і впорядковувати картки в списках і дошках, забезпечуючи гнучкий і наочний спосіб управління проєктами.
2. Співпраця та взаємодія. Trello дозволяє членам команди співпрацювати над завданнями та проєктами в режимі реального часу. Користувачі можуть додавати коментарі, прикріплювати файли та позначати членів команди для обговорення, що сприяє комунікації та прозорості між членами команди.
3. Персоналізація. Trello надає можливості кастомізації, такі як додавання міток, термінів виконання, контрольних списків та кастомних полів до карток. Це дозволяє командам адаптувати платформу до своїх конкретних потреб і процесів управління проєктами.
4. Автоматизація. Trello пропонує функції автоматизації за допомогою функції "Butler", яка дозволяє користувачам автоматизувати повторювані завдання, встановлювати правила і створювати індивідуальні робочі процеси для оптимізації процесів управління.

Переваги використання Trello для командного управління проєктами полягають у наступному:

– Візуальний та інтуїтивно зрозумілий інтерфейс. Візуальний підхід Trello з використанням дошок, списків та карток полегшує розуміння та використання платформи членами команди, сприяючи адаптації користувачів та підвищенню продуктивності.

- Гнучкість та адаптивність. Гнучка структура дошок, списків і карток Trello дозволяє командам налаштовувати платформу під свої робочі процеси і способи управління, що робить її зручною для різних типів проєктів і розмірів команд.
- Співпраця та ефективність. Trello сприяє командній співпраці та підвищенню взаємодії, надаючи членам команди наочну та доступну платформу для обміну інформацією про проєкти в режимі реального часу, покращуючи комунікацію та підзвітність.
- Інтеграція з іншими інструментами: Trello інтегрується з різними сторонніми інструментами, такими як Google Drive, Slack та календарі, що дозволяє безперешкодно обмінюватися даними та керувати робочими процесами на різних платформах.

До недоліків використання Trello для командного управління проєктами можна віднести наступні:

- Обмежений функціонал. Функції Trello можуть бути відносно базовими у порівнянні з іншими платформами, що може не відповідати сучасним потребам деяких команд або організацій.
- Масштабованість. Trello може стати менш ефективним для управління великими та складними проєктами з багатьма завданнями та залежностями, оскільки візуальний підхід може стати перевантажуючим та складним в управлінні.
- Ціноутворення. Хоча Trello пропонує безкоштовну версію, розширені функції та інтеграції вимагають платної підписки, що може вплинути на команди та організації з обмеженим бюджетом.

Загалом, Trello пропонує гнучкий і наочний підхід до управління командними проєктами з такими функціями, як дошки, списки і картки, співпраця і комунікація, кастомізація та можливості автоматизації. Його перевагами є інтуїтивно зрозумілий інтерфейс, гнучкість, можливість

співпраці та інтеграції. Однак можуть бути й недоліки, пов'язані з обмеженим функціоналом, відсутністю розширених звітів, масштабуванням для великих проектів і цінами. Ретельне вивчення можливостей та обмежень Trello необхідне для того, щоб визначити його придатність для конкретних потреб командного управління проектами [10].

1.4.3. Microsoft Teams

Microsoft Teams – це комплекснеПЗ для командної співпраці та комунікації, яка пропонує широкий спектр можливостей і функцій для командного управління проєктами. Ключові особливості, функціональні можливості, переваги та недоліки:

Можливості та функції:

1. Чат і обмін повідомленнями. Microsoft Teams надає надійну систему чату та обміну повідомленнями, яка дозволяє членам команди спілкуватися в режимі реального часу за допомогою індивідуальних і групових каналів. Вона також підтримує голосові та відеодзвінки, спільний доступ до екрана та спільний доступ до файлів, що сприяє ефективному спілкуванню та співпраці між членами команди.
2. Команди та канали. Microsoft Teams дозволяє користувачам створювати команди для конкретних проєктів або відділів, а в межах кожної команди можна створювати канали для різних тем або завдань. Це забезпечує структурований спосіб організації та управління обговореннями, файлами та завданнями, пов'язаними з конкретними проєктами або командами.
3. Інтеграція з Office 365. Microsoft Teams легко інтегрується з іншими програмами Office 365, такими як Word, Excel, PowerPoint і SharePoint, що дозволяє командам співпрацювати над документами, електронними таблицями та презентаціями безпосередньо в межах платформи.

4. Керування завданнями. Microsoft Teams пропонує вбудовані функції керування завданнями, зокрема можливість створювати та призначати завдання, встановлювати терміни виконання та відстежувати прогрес. Завдання можна організовувати в межах каналів або інтегрувати з іншими інструментами керування завданнями, як-от Microsoft Planner чи сторонніми програмами.

Переваги використання Microsoft Teams для командного управління проєктами полягають у наступному:

- Інтегрована співпраця. Microsoft Teams пропонує бездоганну інтеграцію з іншими програмами Office 365, забезпечуючи комплексну екосистему для командної співпраці та управління проектами в межах єдиної платформи.
- Надійні засоби комунікації. Microsoft Teams пропонує широкий спектр комунікаційних функцій, включаючи чат, голосові та відеодзвінки, що дозволяє членам команди спілкуватися в режимі реального часу та залишатися на зв'язку незалежно від їхнього місцезнаходження або пристрою.
- Можливості керування завданнями. Microsoft Teams надає вбудовані функції управління завданнями, що дозволяють командам створювати, призначати та відстежувати завдання безпосередньо в платформі, усуваючи необхідність у зовнішніх інструментах управління завданнями.

Недоліки використання Microsoft Teams для управління командними проектами можуть бути такими:

- Довший процес навчання. Microsoft Teams пропонує широкий спектр функцій і можливостей, які можуть потребувати певного часу та зусиль від членів команди, особливо для користувачів, які не знайомі з іншими програмами Office 365, щоб їх вивчити та освоїти.

– Складний інтерфейс. Microsoft Teams може бути складним для користувачів, які не звикли до його інтерфейсу, оскільки він пропонує безліч вкладок, каналів і функцій, які можуть вимагати ретельної організації та управління, щоб уникнути безладу.
– Ліцензування та вартість. Хоча Microsoft Teams пропонує безкоштовну версію, розширені функції та інтеграції вимагають платної підписки на Office 365, що може вплинути на бюджет невеликих команд або організацій з обмеженим бюджетом.

Отже, Microsoft Teams – це комплексна платформа для командної співпраці та комунікації, яка пропонує широкий спектр можливостей і функцій для управління командними проєктами, включаючи чат і обмін повідомленнями, команди і канали, інтеграцію з Office 365, управління завданнями та інструменти для спільної роботи. Її перевагами є інтегрована співпраця, надійні комунікаційні функції, можливості управління завданнями, а також безпека та відповідність вимогам. Однак можуть бути й недоліки, пов'язані з більш складним процесом навчання, перевантаженим інтерфейсом, а також з питаннями ліцензування та вартості. Ретельна оцінка функцій та обмежень Microsoft Teams необхідна, щоб визначити її придатність для конкретних потреб в управлінні командними проектами [11].

1.4.4. Notion

Notion – це універсальна платформа "все-в-одному", яка поєднує в собі функції ведення нотаток, управління проєктами, організації баз даних та спільної роботи. Нижче наведено докладний аналіз Notion, включаючи його особливості, функціональні можливості, переваги та недоліки.

Особливості та функціональні можливості:

1. Налаштовуваний робочий простір. Notion дозволяє користувачам створювати індивідуальний робочий простір зі сторінками, базами даних, таблицями, календарями тощо, забезпечуючи гнучку платформу

для організації та управління командними проєктами у візуальний та інтуїтивно зрозумілий спосіб.

2. Управління завданнями та проєктами. Notion надає вбудовані функції управління завданнями та проектами, включаючи можливість створювати завдання, встановлювати терміни виконання, додавати мітки та нагадування, а також відстежувати прогрес. Ці функції допомагають командам залишатися організованими та ефективно керувати своїми проєктами.
3. Співпраця та обмін даними. Notion пропонує надійні функції для спільної роботи, включаючи редагування, коментування та сповіщення в режимі реального часу, що дозволяє членам команди співпрацювати над сторінками, базами даних і документами в режимі реального часу. Він також надає можливості спільного доступу для зовнішніх співавторів, що полегшує співпрацю з клієнтами, партнерами або зацікавленими сторонами.
4. Управління базами даних та знаннями. Notion дозволяє користувачам створювати бази даних і таблиці для організації та управління інформацією, такою як деталі проєкту, контакти, ресурси тощо. Він також пропонує потужні можливості фільтрації, сортування та запитів, що полегшує пошук та швидке отримання інформації.

Переваги використання Notion наступні:

– Гнучкість і кастомізація. Notion пропонує робоче середовище, яке легко налаштовується і може бути адаптоване до конкретних потреб команди або проєкту. Користувачі можуть створювати сторінки, бази даних і таблиці відповідно до своїх унікальних вимог, що робить його універсальною платформою для командної співпраці.
– Платформа "все в одному". Notion поєднує в собі безліч функцій, таких як ведення нотаток, управління завданнями, організація баз даних і спільна робота, в одній платформі, зменшуючи потребу в декількох

інструментах і спрощуючи робочий процес для командного управління проектами.

- Співпраця та обмін даними. Notion надає надійні функції для спільної роботи, дозволяючи членам команди співпрацювати в режимі реального часу та легко обмінюватися інформацією з внутрішніми та зовнішніми співробітниками, сприяючи ефективній командній комунікації та співпраці.

Недоліки використання Notion для командного управління проєктами можуть включати в себе наступні моменти:

- Час навчання. Гнучкі та персоналізовані функції Notion можуть потребувати певного часу для вивчення та адаптації членами команди, особливо для користувачів, які не знайомі з подібними платформами або базами даних.
- Обмежені розширені функції. Хоча Notion пропонує широкий спектр можливостей, йому може не вистачати деяких розширених функцій у порівнянні зі спеціалізованими інструментами управління проєктами, такими як діаграми Ганта, відстеження часу та розподіл ресурсів, які можуть знадобитися для складних або масштабних проєктів.

Підсумовуючи, можна сказати, що Notion – це універсальна платформа "все в одному", яка пропонує гнучкий і настроюваний робочий простір для командного управління проєктами, включаючи такі функції, як управління завданнями і проектами, співпраця і спільне використання, управління базами даних і знаннями, а також інтеграція з іншими інструментами [12].

1.4.5. BaseCamp

Basecamp – це інструмент для управління проєктами та командної співпраці, який пропонує широкий спектр можливостей та функцій. Нижче наведено поглиблений аналіз Basecamp, включаючи його особливості, функціональні можливості, переваги та недоліки:

Можливості та функції:

1. Управління проектами. Basecamp надає інструменти для створення та управління проєктами, включаючи можливість створювати списки справ, встановлювати терміни виконання, призначати завдання та відстежувати прогрес. Він також пропонує шаблони проєктів для повторюваних проєктів, що полегшує створення та управління ними.
2. Комунікація та співпраця. Basecamp пропонує вбудовані функції для спілкування та співпраці, включаючи дошки оголошень, групові чати, спільний доступ до файлів та сповіщення в режимі реального часу. Ці функції полегшують командну комунікацію, співпрацю та обмін файлами на централізованій платформі.
3. Управління документами. Basecamp надає функції зберігання та управління документами, дозволяючи командам завантажувати, організовувати та співпрацювати над документами в рамках проєктів. Він також пропонує контроль версій і коментування документів, що полегшує спільну роботу над документами в режимі реального часу.

Переваги використання Basecamp наступні:

- Простота і зручність використання. Basecamp має простий та інтуїтивно зрозумілий користувацький інтерфейс, що дозволяє членам команди швидко почати користуватися платформою без тривалого навчання чи технічних знань.
- Співпраця з клієнтами. Basecamp пропонує функції для співпраці з клієнтами, що дозволяють командам співпрацювати в безпечному та контрольованому середовищі, покращуючи комунікацію з клієнтами та залучення їх до проєктів.
- Шаблони проектів. Basecamp надає шаблони проєктів, які можна використовувати для повторюваних проєктів, заощаджуючи час і зусилля при створенні та управлінні ними.

Недоліки використання Basecamp для командного управління проєктами полягають в наступному:

– Обмежені розширені функції. Basecamp може не мати деяких розширених функцій, таких як відстеження часу, діаграми Ганта та розподіл ресурсів, які можуть знадобитися для складних або масштабних проєктів.
– Обмежена кастомізація. Basecamp має обмежені можливості кастомізації порівняно з іншими інструментами, що може заважати командам пристосовувати платформу до своїх конкретних потреб.

Отже, Basecamp – це простий та інтуїтивно зрозумілий інструмент для командної співпраці, який пропонує функції для управління проєктами, комунікації, співпраці, документообігу та взаємодії з клієнтами. Серед його переваг – простота, універсальність платформи, функції співпраці з клієнтами та шаблони проєктів. Однак він може мати обмеження щодо розширених функцій та можливостей кастомізації. Оцінка функцій та обмежень Basecamp важлива для визначення його придатності для конкретних потреб[13].

1.5. Використання платформ в реальних проєктах

Реальне застосування платформ для управління проєктами, таких як Asana, Trello, Microsoft Teams та Notion, широко висвітлено в тематичних дослідженнях та історіях успіху. Ці платформи використовуються командами різних розмірів та галузей, від стартапів до транснаціональних корпорацій.

Одним із прикладів успішного впровадження є використання Airbnb Trello. Команда інженерів Airbnb використовує Trello для управління своїми складними проєктами з розробки програмного забезпечення, від відстеження помилок до планування запуску продукту. Налаштовувані функції

платформи дозволяють їм створювати дошки, пристосовані до їхніх робочих процесів, та ефективно співпрацювати між різними командами та відділами.

Інший приклад – використання Asana в The New York Times. Команда з цифрової стратегії газети використовує Asana для управління проєктами, пов'язаними з редизайном веб-сайтів, запуском продуктів і кампаніями в соціальних мережах. Функції управління завданнями та проєктами в Asana дозволяють координувати міжфункціональні команди та ефективно відстежувати прогрес.

Notion також використовується такими компаніями, як Fiverr, для управління своїми проєктами. Команда розробників Fiverr використовує Notion для управління дорожніми картами продуктів, дослідженнями користувачів та запитами на функції. Можливості кастомізації Notion та функції управління документами дозволяють їм оптимізувати робочі процеси та співпрацювати над проєктами в режимі реального часу.

Microsoft Teams використовується такими компаніями, як Ernst & Young (EY). EY використовує Teams для спілкування та співпраці зі своїми клієнтами та колегами над різними проєктами, від аудиту до консультаційних послуг. Інтеграція Teams з іншими додатками Microsoft, такими як OneNote та SharePoint, дозволяє безперешкодно управляти документами та обмінюватися ними, а функції співпраці та комунікації сприяють ефективній координації роботи команди.

У цих випадках використання даних платформ призвело до підвищення ефективності, продуктивності та співпраці між командами. Можливості та функції платформ дозволили командам оптимізувати свої робочі процеси та ефективно відстежувати прогрес, що призвело до успішного завершення проєктів.

Однак важливо зазначити, що успішне впровадження цих платформ також залежить від таких факторів, як культура організації, розмір команди,

складність проєкту та підтримка керівництва. Впровадження цих платформ вимагає змін у процесах і робочих циклах, а для успішного впровадження може знадобитися навчання та адаптація персоналу.

Статистичний аналіз використання платформ для управління проєктами в реальних розробках може надати цінну інформацію про їхню ефективність та вплив на результати проекту. Було проведено кілька досліджень для аналізу використання таких платформ, як Asana, Trello, Microsoft Teams та Notion у різних галузях та сферах застосування.

Одне дослідження, проведене компанією Capterra, проаналізувало відгуки користувачів про платформи і виявило, що Asana і Trello є двома найпопулярнішими платформами, на які припадає 66% від загальної кількості проаналізованих відгуків. Дослідження також показало, що користувачі високо оцінили ці платформи за простоту використання, функції управління завданнями та можливості співпраці.

В іншому дослідженні Wrike проаналізували вплив програмного забезпечення на результати проектів і виявили, що команди, які його використовують, завершили свої проекти на 16,5% швидше, ніж ті, які не використовували його. Дослідження також показало, що програмне забезпечення допомагає командам краще управляти своїми ресурсами та покращує командну комунікацію [14].

У тематичному дослідженні Notion проаналізували вплив використання їхньої платформи на продуктивність віддаленої команди. Дослідження показало, що команда змогла скоротити час щоденних зустрічей з 90 хвилин до 30 хвилин, що призвело до збільшення продуктивності на 50%. Команда також змогла впорядкувати свої робочі процеси та покращити результати проекту завдяки ефективній співпраці та управлінню завданнями за допомогою платформи.

Аналогічно, в тематичному дослідженні Microsoft Teams було проаналізовано вплив використання платформи на продуктивність організації охорони здоров'я. Дослідження показало, що використання Teams дозволило організації скоротити час, витрачений на адміністративні завдання, на 35%, що призвело до значної економії коштів. Платформа також покращила командну співпрацю та комунікацію, що призвело до швидшого прийняття рішень та кращих результатів для пацієнтів.[15]

Загалом, ці дослідження свідчать про те, що платформи управління проєктами можуть мати значний позитивний вплив на результати роботи, включаючи підвищення продуктивності, покращення співпраці та краще управління ресурсами. Статистичний аналіз відгуків користувачів і тематичних досліджень дає цінну інформацію про ефективність різних платформ, що дозволяє організаціям приймати обґрунтовані рішення при виборі платформи для своїх потреб.

РОЗДІЛ 2. ПРОЕКТУВАННЯ ТА РОЗРОБКА ВЕБ-ДОДАТКУ ДЛЯ ОРГАНІЗАЦІЇ КОМАНДНОЇ РОБОТИ

2.1. Вибір технологій та фреймворків для розробки

Вибір правильних технологій та фреймворків для розробки є вирішальним кроком на шляху до створення успішного додатку. Зважаючи на постійне зростання кількості доступних мов програмування, бібліотек та інструментів, визначити, які з них найкраще підходять для роботи, може бути непростим завданням. Правильний вибір може суттєво вплинути на результат проєкту, впливаючи на його масштабованість, продуктивність та ремонтопридатність. У цьому підпункті ми обговоримо фактори, які впливають на вибір відповідних технологій та фреймворків для розробки.

2.1.1. React

React – це популярна бібліотека JavaScript, яка широко використовувалася в проєкті. Однією з головних переваг React є можливість створювати багаторазові компоненти, які можна використовувати у всьому додатку. Такий підхід робить код більш модульним, легшим для розуміння і більш зручним для підтримки. React також використовує віртуальний DOM, що допомагає підвищити продуктивність за рахунок мінімізації кількості оновлень, які потрібно зробити в реальному DOM.

Однією з ключових особливостей React є його здатність обробляти зміни стану у передбачуваний спосіб. Станом додатку можна легко керувати та оновлювати за допомогою вбудованої в React системи управління станами, що допомагає запобігати помилкам та гарантувати, що користувацький інтерфейс залишається адаптивним. React також використовувався разом з іншими бібліотеками та фреймворками, такими як Redux, для управління станом додатку та забезпечення узгодженості даних між усіма компонентами. Такий підхід допоміг зменшити складність додатку та спростити його налагодження і підтримку.

Загалом, React виявився потужним інструментом, який допоміг створити сучасний та адаптивний користувацький інтерфейс для додатку. Його модульний підхід, система управління станами та здатність безперешкодно працювати з іншими бібліотеками та фреймворками зробили його ідеальним вибором для проекту.[16]

2.1.2. TypeScript

TypeScript – це надбудова JavaScript, яка широко використовувалася в проєкті. TypeScript має багато переваг над JavaScript, включаючи статичну типізацію, кращий інструментарій та покращену підтримку коду. Додаючи статичні типи до JavaScript, TypeScript дозволяє розробникам виявляти помилки під час компіляції, а не під час виконання, що допомагає запобігти помилкам і підвищити загальну якість коду.

Однією з головних переваг TypeScript є можливість визначати інтерфейси, які допомагають забезпечити узгодженість структур даних у всьому додатку. Такий підхід полегшує керування складними потоками даних і підтримує їхню узгодженість. Крім того, інструментальна підтримка TypeScript, включаючи завершення коду та перевірку типів, допомагає розробникам писати більш зручний для супроводу програм. TypeScript використовувався разом з React для створення багаторазових компонентів та покращення загальної якості коду. Визначаючи інтерфейси та типи, TypeScript допоміг гарантувати, що дані, які передаються між компонентами, є узгодженими та безпечними за типом. Такий підхід допоміг зменшити кількість помилок під час виконання та підвищити стабільність роботи додатку.

Крім того, TypeScript був використаний для створення типізованого шару API, який взаємодіє з базою даних CouchDB/PouchDB. Такий підхід допоміг гарантувати, що дані, які отримуються та зберігаються в базі даних, є

послідовними та безпечними за типом, що покращило загальну якість додатку.

Загалом, TypeScript був потужним інструментом, який допоміг покращити зручність супроводу, стабільність та загальну якість програми. Його здатність визначати інтерфейси та типи, а також інтеграція з React та базою даних CouchDB/PouchDB зробили його ідеальним вибором для проекту.[17]

2.1.3. Redux

Redux – це бібліотека управління станом, яка відіграла вирішальну роль у проекті. Вона широко використовувалася для управління станом додатку, включаючи дані з бази даних CouchDB/PouchDB та елементи користувацького інтерфейсу. Основна мета Redux – полегшити керування станом додатку та забезпечити узгодженість даних між усіма компонентами.

Redux було використано для створення глобального сховища, доступного для всіх компонентів додатку, що полегшило керування станом додатку та забезпечило узгодженість даних між усіма компонентами. Сховище дозволило централізовано керувати статусом додатку та спростило передачу інформації між компонентами.

Однією з головних переваг Redux була його здатність працювати зі складними структурами даних та керувати станом у передбачуваний та ефективний спосіб. Він надавав набір інструментів для налагодження та моніторингу стану додатку, що полегшувало діагностику та виправлення помилок. Відокремивши управління станами від логіки користувацького інтерфейсу, він зменшив складність додатку і зробив код більш модульним та зрозумілим. Redux також легко інтегрувався з іншими бібліотеками та фреймворками, такими як React, надаючи набір API, які дозволяли React-компонентам взаємодіяти зі сховищем та оновлювати стан додатку. Це

спростило створення складних користувацьких інтерфейсів, які потребують оновлення в реальному часі.

Таким чином, Redux став потужним інструментом, який допоміг керувати станом додатку та забезпечувати узгодженість даних між усіма компонентами. Його здатність обробляти складні структури даних, інтегруватися з іншими бібліотеками та фреймворками, а також надавати інструменти для налагодження та моніторингу зробили його ідеальним вибором для проекту.[18]

2.1.4. Material-UI

Material-UI – популярний фреймворк React UI, який широко використовувався в проєкті. Він надає набір попередньо розроблених компонентів та стилів, які можна використовувати для створення сучасних та адаптивних користувацьких інтерфейсів. Однією з ключових переваг Material-UI є його здатність створювати узгоджені та візуально привабливі інтерфейси на різних платформах та пристроях.

У проєкті Material-UI використовувався для створення більшості компонентів користувацького інтерфейсу, включаючи форми, діалогові вікна та навігаційні меню. Він був обраний за простоту використання та можливість створювати складні макети з мінімальними зусиллями. Попередньо розроблені компоненти Material-UI заощадили час і зусилля, зменшивши необхідність створювати кастомні компоненти з нуля.

Material-UI також надає систему тем, яка дозволяє розробникам налаштовувати зовнішній вигляд програми, змінюючи кольори, шрифти та інші елементи дизайну. Ця функція була широко використана в проекті для створення послідовного та візуально привабливого вигляду всього додатку.

Ще однією перевагою Material-UI є його документація та підтримка спільноти. Фреймворк має велику бібліотеку документації та велику й активну спільноту, яка надає підтримку та рекомендації щодо ефективного

використання фреймворку. Material-UI не тільки надає багатий набір попередньо розроблених компонентів, але й дозволяє легко налаштовувати їх відповідно до конкретних вимог проекту. Це дозволило швидко змінювати дизайн та користувацький інтерфейс, що полегшило впровадження оновлень. На додаток до попередньо розроблених компонентів, Material-UI також надавав потужний набір утиліт та інструментів, які допомогли оптимізувати процес розробки. Наприклад, система Grid фреймворку дозволяє легко створювати складні макети за допомогою простого та інтуїтивно зрозумілого API.

Можливості адаптивного дизайну Material-UI також були ключовою особливістю, яка допомогла гарантувати, що додаток чудово виглядатиме на різних пристроях і розмірах екранів. Це було особливо важливо, оскільки додаток призначався для використання широким колом користувачів, від стаціонарних комп'ютерів до мобільних пристроїв. Ще однією перевагою Material-UI була його орієнтованість на доступність. Фреймворк забезпечував вбудовану підтримку стандартів доступності, таких як WCAG 2.0, що полегшувало створення інклюзивного користувацького інтерфейсу, яким могли б користуватися всі користувачі, незалежно від їхніх здібностей.

Загалом, Material-UI був важливим компонентом успіху проекту. Багатий набір попередньо розроблених елементів, можливості кастомізації, адаптивний дизайн та функції доступності зробили його ідеальним вибором для створення сучасного та зручного для користувачів додатку. Активна спільнота фреймворку та обширна документація також допомогли забезпечити ресурсами та підтримкою, необхідними для ефективного використання фреймворку.[19]

2.1.5. CouchDB/PouchDB

CouchDB і PouchDB – це NoSQL-бази даних з відкритим вихідним кодом, які використовувалися в проекті для зберігання та управління даними.

Вони розроблені для безперебійної роботи з веб- та мобільними додатками і забезпечують гнучкі та масштабовані рішення для управління інформацією.

CouchDB – це документно-орієнтована база даних, яка використовує JSON для зберігання даних, що полегшує роботу з ними у веб- та мобільних додатках. PouchDB – це бібліотека JavaScript, яка надає простий у використанні інтерфейс для роботи з базами даних CouchDB в автономному режимі та синхронізації даних з сервером. У проєкті CouchDB і PouchDB використовувалися для зберігання та управління різними типами даних, включаючи облікові записи користувачів, інформацію про проект і списки завдань. Бази даних забезпечили надійне і гнучке рішення для управління інформацією, яке дозволило легко синхронізувати дані між клієнтом і сервером.

Підхід PouchDB "спочатку офлайн, потім онлайн" був особливо корисним у проекті, оскільки він дозволив додатку продовжувати функціонувати навіть тоді, коли інтернет-з'єднання було втрачено. Дані автоматично синхронізувалися, як тільки інтернет-з'єднання відновлювалося. CouchDB та PouchDB також надають набір API, які полегшують роботу з даними у веб- та мобільних додатках. Універсальна модель даних та можливості реплікації баз даних дозволили створити масштабовану та відмовостійку систему.

Загалом, CouchDB та PouchDB стали потужними інструментами, які допомогли керувати даними в додатку та забезпечити їхню доступність і узгодженість на всіх пристроях і платформах. Їх гнучкість, надійність та офлайн-підхід зробили їх ідеальним вибором для проекту [20].

2.2. Розроблення функціональних вимог до проєкту

У цьому розділі представлено функціональні вимоги до веб-додатку, розробленого для командного управління проектами. Мета додаткузабезпечити комплексну платформу, яка полегшує ефективну

співпрацю, управління завданнями та відстеження прогресу між членами команди.

Таблиця 1. Функціональні вимоги до програмної системи

Функціональна вимога	Опис
Створення дошок	Користувачі повинні мати можливість створювати дошки для організації та категоризації завдань проекту
Редагування дошок	Користувачі повинні мати можливість редагувати дані дошки, такі як назва та опис, щоб відобразити зміни у вимогах або цілях проекту
Створення та редагування завдань	Користувачі повинні мати можливість створювати завдання на кожній дошці і надавати відповідні деталі, такі як назва завдання, опис і терміни виконання
Відстеження прогресу виконання завдань	Користувачі можуть відстежувати хід виконання завдань за допомогою попередньо визначених статусів: "Done", "In progress" і "To do"
Календарний перегляд завдань	Користувачі повинні мати доступ до перегляду календаря, який відображає завдання і відповідні терміни їх виконання, забезпечуючи огляд проекту
Повністю налаштовуваний робочий простір	Додаток повинен пропонувати робочу область, що легко налаштовується, дозволяючи користувачам персоналізувати макет, види та налаштування відповідно до їхніх уподобань
Підтримка синтаксису Markdown	Додаток повинен підтримувати синтаксис Markdown в описах завдань, дозволяючи користувачам форматувати текст, створювати контрольні списки і додавати гіперпосилання для ефективного управління завданнями
Інтеграція з сервером CouchDB	Веб-додаток повинен бути

	інтегрований з сервером CouchDB для зберігання даних, синхронізації та можливості офлайн-доступу

Ці функціональні вимоги окреслюють необхідні функції та можливості веб-додатку. Вони забезпечують ефективну співпрацю, управління завданнями та відстеження прогресу, а також надають настроюваний і зручний робочий простір. Інтеграція з сервером CouchDB гарантує надійне зберігання та синхронізацію даних, дозволяючи користувачам отримувати доступ до своїх проектів і працювати над ними як онлайн, так і офлайн. Підтримка синтаксису Markdown підвищує чіткість і організованість завдань, полегшуючи членам команди спілкування та ефективну співпрацю.

2.3. Бекенд веб-додатку

Бекенд слугує основою програми, забезпечуючи основні функції, такі як зберігання, пошук та обробка даних. Він забезпечує безперебійний зв'язок між фронтендом і базою даних, дозволяючи безперешкодно інтегрувати функції та ефективно керувати даними. У цьому розділі ми обговоримо архітектурний дизайн, технології, що використовуються, та ключові компоненти розробки бекенду.

Бекенд побудований за архітектурою клієнт-сервер, де фронтенд взаємодіє з сервером для виконання різних операцій. Така архітектура забезпечує масштабований та ефективний зв'язок, коли сервер отримує запити від фронтенду, обробляє їх та взаємодіє з базою даних для отримання або оновлення необхідних даних. Такий розподіл завдань між фронтендом і бекендом підвищує загальну продуктивність і ремонтопридатність додатку.

Використані технології:

- Node.js – серверне середовище виконання JavaScript, забезпечує надійну і масштабовану платформу для розробки бекенду. Його

здатність використовувати JavaScript та TypeScript, у нашому випадку, як на фронтенді, так і на бекенді забезпечує узгодженість у всьому додатку.

- Express.js – гнучкий фреймворк веб-додатків для Node.js, пропонує ряд інструментів і функцій для створення надійних API, управління проміжним програмним забезпеченням і маршрутизації.
- CouchDB – база даних NoSQL, забезпечує гнучку модель даних, безперебійну реплікацію даних та можливості офлайн-доступу. Вона особливо добре підходить для додатків, які потребують ефективної синхронізації даних та обробки великих обсягів неструктурованих або напівструктурованих даних.

Внутрішні компоненти:

- Кінцеві точки API. Бекенд надає набір кінцевих точок API, які полегшують взаємодію з фронтендом. Ці кінцеві точки обробляють такі операції, як створення дошки, управління завданнями та отримання даних з бази даних CouchDB.
- Проміжне програмне забезпечення для автентифікації. Для забезпечення безпеки реалізовано проміжне програмне забезпечення для аутентифікації. Воно аутентифікує вхідні запити до захищених кінцевих точок.
- Моделі даних. Бекенд визначає моделі даних для представлення дошок, завдань, користувачів та інших об'єктів у додатку. Ці моделі визначають структуру даних, що зберігаються в базі даних CouchDB, і надають методи для пошуку, маніпулювання та перевірки даних.
- Інтеграція з базою даних. Бекенд встановлює з'єднання з сервером CouchDB, забезпечуючи безперешкодну інтеграцію для зберігання та пошуку даних. Він використовує API CouchDB та пов'язані з ним бібліотеки для виконання операцій з базою даних, таких як створення документів, оновлення та запити.

Діаграма роботи бекенду виглядає так:

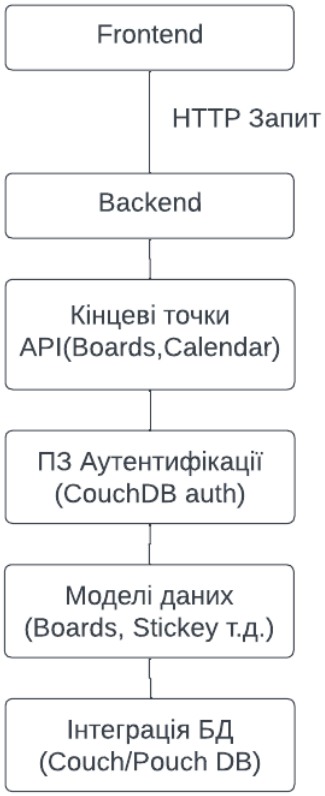

Рис. 2.1. Діаграма бекенду проєкту

Загалом, ця архітектура бекенду забезпечує безперебійний зв'язок між фронтендом і базою даних. Кінцеві точки API дозволяють взаємодіяти з функціями програми, проміжне програмне забезпечення для автентифікації забезпечує безпечний доступ до захищених кінцевих точок, моделі даних визначають структуру та логіку зберігання даних, а інтеграція з CouchDB сприяє ефективному зберіганню та пошуку даних.

Ключові можливості для зберігання та синхронізації даних в NoSQL базі даних полягають в наступному.

Додаток використовує документо-орієнтоване середовище, що не передбачає класичних схем бази даних та діаграм. Усі потрібні дані для роботи із проектом зберігаються хешовано у браузері у форматі JSON-файлів, що дозволяє гнучко та динамічно змінювати моделі даних, наприклад

для реалізації налаштувань кожної дошки. У програмі використовуються методи по типу startAddStickey та startUpdateAppconfig для вибору та маніпулюванню певними властивостями документа.

Метод розподіленої архітектури, який широко використовується в створеному додатку дозволяє використовувати реплікації даних і забезпечує високу доступність до усіх елементів програми. Організація унарного сховища для усіх модулів дозволяє легко і безперешкодно контролювати стан додатку окремо для усіх розділів. Програма використовує методи startSync() та setRemoteDb() для синхронізації локальної бази даних з віддаленою базою даних, що дозволяє взаємодіяти та перерозприділяти дані серед обох. Контроль даного процесу знаходиться на стороні бекенду та контролюється методами запуску та рестарту.

CouchDB надає механізми виявлення та вирішення конфліктів при зміні структури файлів бази даних та їх переміщенні. У розподіленому середовищі конфлікти можуть виникати при одночасній модифікації декількох копій документа. Хоча додаток не містить явних функцій вирішення конфліктів, CouchDB надає такі можливості на базі API.

Для пошуку та аналізу даних використовується представлення MapReduce. Хоча програма безпосередньо не включає дані функції мапування, представлення CouchDB використовуються для ефективного запиту та перетворення даних.

Так як усі дані зберігаютьсяу вигляді ключ:значення у форматі JSON-файлів, на прикладі таблиці “Завдання” ми можемо побачити дані котрі зберігаються у базі даних:

```
export type KanbanRecordDbRecordUserData = {
    type: 'kanban',
    dueDate: string;
    description: string;
    barcode: string;
    memo: string;
    flags: string[];
    tags: string[];
    boardId: string;
    taskStatus: string;
    teamOrStory: string;
};
```

Рис.2.2. Структура даних таблиці “Завдання”

На рисунку 2.2 ми можемо побачити структуру однієї із таблиць бази даних у котрій зберігаються усі записи для відображення на дошці, тобто поставлені завдання. Перш за все встановлюється тип елементу, після чого перечислюються усі потрібні значення, такі як : дата здачі завдання, опис, можливість додати qr-код, нотатки до завдання, мітки, Id дошки до котрої буде додано завдання, його статус та категорію.

Використання бібліотеки PouchDB

В проєкті широко було використано даний потужний інструмент для досягнення безперебійної синхронізації між локальною та віддаленою базою даних. Використовуючи базу даних CouchDB було створено зручний спосіб збереження даних у додатку.

PouchDB дозволяє виконувати різні операції з базами даних локально, такі я створення баз даних, збереження документів, оновлення записів, видалення даних і виконання запитів. Ці операції забезпечують ефективний спосіб зберігання та управління.

Використавши слухачі подій, які довзоляють відстежувати та реагувати на зміни в локальних та віддалених базах даних ви можете підписатись на такі подіх, як зміни в документах чи оновлення бази даних. Ця можливість дозволяє взаємодіяти в режимі реального часу та реагувати на зміни у вашому додатку.

Однією із найбільш визначних причин використання саме цього типу бази даних для проекту є підтримка роботи у оффлайн режимі. Це дозволяє додатку рпи автономній роботі продовжувати безперебійно працювати. Зміни даних, внесені під час тако сесії, автоматично синхронізуються після відновлення мережевого з'єднання, забезпечуючи безперебійну роботу користувача незалежно від доступності мережі.

Використовуючи даний тип бази даних ми можемо переглядати усі зміни та редагувати їх за потреби на стороні додатку, у кабінеті сервера якщо ви використовуєте віддалену базу та прямо із браузерної сторінки. Приклад відображення даних у базі можна спостерігати на рисунку 2.3:

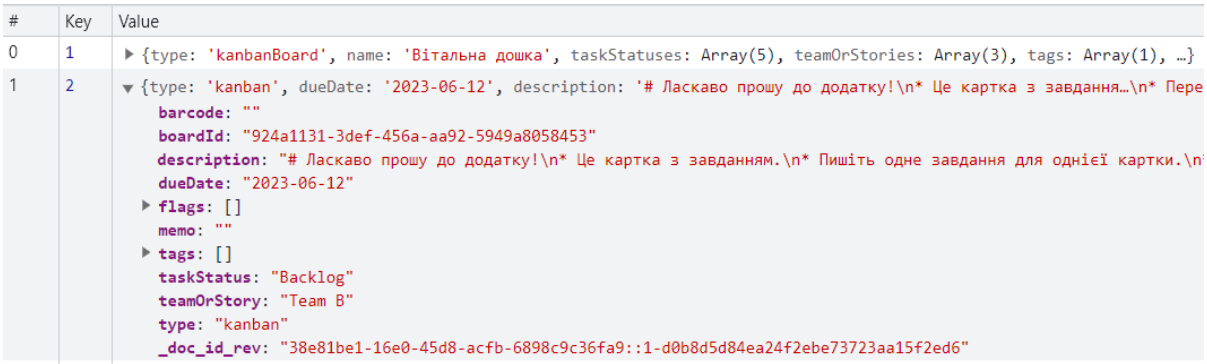

Рис. 2.3. Приклад збереження даних у базі

Як ми можемо побачити, дані зберігаються документами, це спрощує роботу із великими об'ємами інформації, в даному випадку із цілими дошками та картами завдань у яких зберігаються достатньо значні масштаби інформації.

2.4. Фронтенд веб-додатку

Фронтенд відповідає за створення користувацького інтерфейсу та забезпечення безперебійної роботи користувачів. Він надає користувачам можливість взаємодіяти з функціями програми, переглядати інформацію, пов'язану з проектом, і виконувати різні операції. У цьому розділі ми розглянемо архітектурний дизайн, технології, що використовуються, та ключові компоненти, що беруть участь у розробці інтерфейсу.

Фронтенд використовує клієнтську архітектуру, де користувацький інтерфейс рендериться і виконується на пристрої клієнта. Така архітектура забезпечує адаптивність і масштабованість, дозволяючи отримати доступ до додатку з різних пристроїв і платформ. Вона також забезпечує ефективний зв'язок з бекендом через кінцеві точки API для отримання даних та маніпуляцій з ними.

Розглянемо наступні фронтенд-компоненти:

Інтерфейс користувача. Фронтенд створює інтуїтивно зрозумілий та зручний інтерфейс, який дозволяє користувачам взаємодіяти з функціями та можливостями додатку. Він включає такі компоненти, як навігаційні меню, форми, списки завдань, дошки проектів та інші елементи інтерфейсу.

Управління станом. Redux використовується для управління станом додатку. Він зберігає такі дані, як інформація про користувача, деталі проекту, статуси завдань та іншу важливу інформацію. Redux забезпечує послідовний потік даних, спрощує оновлення станів і полегшує обмін даними між різними компонентами.

Зв'язок через API. Фронтенд взаємодіє з бекендом через кінцеві точки API для отримання та оновлення даних. Couch DB API використовується для створення HTTP-запитів до бекенду та обробки відповідей, що забезпечує безперешкодну інтеграцію з бекенд-сервісами.

Стилістика та тематика. Material-UI використовується для створення візуально привабливого та цілісного дизайну. Він надає заздалегідь створені та настроювані компоненти, які відповідають керівним принципам Material Design, забезпечуючи послідовний і професійний вигляд і відчуття в усьому додатку.

Модель UI

Рис. 2.4. Модель користувацького інтерфейсу

На рисунку 2.4 компонент "Інтерфейс користувача" представляє інтерфейс веб-сайту. Він складається з різних вкладок, які дозволяють користувачам переміщатися і взаємодіяти з різними розділами програми.

Вкладка "Дошки" дозволяє користувачам переглядати і керувати своїми існуючими дошками, які можуть містити завдання і інформацію, пов'язану з проектом. Кнопка "Нова дошка" надає користувачам можливість створити нову дошку. Вони можуть вказати назву дошки, її опис та інші необхідні деталі.

Вкладка "Робоча зона" представляє візуальне представлення завдань на дошці, використовуючи підхід до дошки Kanban. Користувачі можуть

переглядати і керувати завданнями в різних колонках, таких як "To-Do", "In Progress" і "Done".

Вкладка "Календар" пропонує перегляд календаря, що дозволяє користувачам бачити пов'язані з проектом терміни, етапи або події у форматі, заснованому на даті. Вони можуть додавати, редагувати і видаляти події в календарі.

Вкладка "Редактор" містить текстовий редактор для повної кастомізації дошок користувача. Вона містить великий набір функціоналу, починаючи додаванням нових стовпців, рядків та зміною їх назви, закінчуючи відображенням лише потрібних даних для конкретних завдань.

Вкладка "Налаштування" дозволяє користувачам налаштовувати різні параметри програми, такі як інтеграція з віддаленим сервером CouchDB.

Ці вкладки надають користувачам повний набір функціональних можливостей для ефективного управління командними проектами та налаштування їхньої роботи в додатку.

2.5. Програмна реалізація

Перш за все під час створення програмного продукту було ініціалізовано пустий React проєкт та встановлено потрібні навантаження такі як ‘react-dom’, ‘redux’, ‘react-redux’, ‘@material-ui/core’, ‘@material-ui/icons’, ‘pouchdb’, ‘pouch-find’ та інші. Після чого було організовано сховище, тобто Redux Store для синхронізації усіх майбутніх елементів проєкту. Файл сховища виглядає наступним чином:

```
//Ініціалізація
let store: Store<AppState, AnyAction> = null as any;

//Функція для отримання сховища додатку
export function getConstructedAppStore() {
    return store;
}

// Отримання асинхронно
export default async function getAppStore() {
    //Створення нового сховища
    if (!store) {
        store = createStore(
            //Підключення усіх змінних
            combineReducers<AppState>({
                router: connectRouter(history),
                appEvents: await getAppEventsReducer(),
                kanbanBoard: await getKanbanBoardReducer(),
                calendar: await getCalendarReducer(),
            }),
            compose(
                applyMiddleware(
                    routerMiddleware(history), //Надсилання дій маршрутизатора
                ),
            ),
        );
    }
    return store;
}
```

Рис. 2.5. Redux Store

На рисунку 2.5ми можемо спостергіти створення нового сховища, підключення роутера та редукторів усіх подій додатку, таких як зміни стану додатку, зміни самої дошки та календаря, до них повернемось пізніше.

Після установки основного транспортного хабу додатку під час розробки я перейшов до створення дій для додатку за допомогою бібліотеки ‘typescript-fsa’. Ця частина коду встановлює базову інфраструктуру для створення та відправки дій у додатку. За допомогою використаної бібліотеки було створено безпечні за типом дії та корисні за навантаженням генератори подій. Потім ці дії будут використовуватись, щоб викликати оновлення стану додатку. Було створено три файли в окремому каталозі для координації дій окремо для стану додатку, стану дошки та її записів та стану календаря. На прикладі скріншоту із діями стану додатку розглянемо як це працює.

```
// інтерфейс дій
export interface AppEventsActions {
    showAlertDialog: (v: ConfirmDialogProps) =>
        Action<ConfirmDialogProps>;
    closeAlertDialog: () =>
        Action<void>;

    updateAppConfig: (v: AppConfig) =>
        Action<AppConfig>;
    resetApplication: () =>
        Action<void>;
}

//методи для створення дій додатку
const actionCreator = actionCreatorFactory();


const updateAppConfig =
    actionCreator.async<AppConfig, AppEventsState, Error>('ACTIONS_UPDATE_APP_CONFIG');
const resetApplication =
    actionCreator.async<void, AppEventsState, Error>('ACTIONS_RESET_APPLICATION');

//опис функцій додатку та всі можливі дії
export const appEventsActions = {
    showAlertDialog: actionCreator<ConfirmDialogProps>('ACTIONS_SHOW_ALERT_DIALOG'),
    closeAlertDialog: actionCreator<void>('ACTIONS_CLOSE_ALERT_DIALOG'),

    startUpdateAppConfig: updateAppConfig.started,
    doneUpdateAppConfig: updateAppConfig.done,
    failedUpdateAppConfig: updateAppConfig.failed,

    startResetApplication: resetApplication.started,
    doneResetApplication: resetApplication.done,
    failedResetApplication: resetApplication.failed,
};
```

Рис. 2.6. AppEvent Actions

На рисунку 2.6 ми можемо спостерігати інтерфейс з назвою AppEventActions, який визначено для представлення дій, які можуть бути виконані у додатку. Він включає такі функції як showAlertDialog, closeAlertDialog, updateAlertDialog та resetApplication. Функція actionCreatorFactory викликається для створення екземпляра творця дій, який буде буде використовуватися для їх створення. Дві асинхронні дії, updateAppConfig та resetApplication, визначаються за допомогою методу actionCreator.async. Ці дії мають різні стадії (started, done, failed) і приймають певні навантаження та стани. Об’єкт appEventActions експортується і містить

створювачі дій для різних операцій у додатку. Він включає функції showAlertDialog, closeAlertDialog, а також різні етапи оновлення AppConfig і перезавантаження програми. Файли із реалізацію функцій календаря та дошки створені за аналогічним принципом.

Після реалізації усіх потрібних дій для додатку перейдемо до створення компонентів користувацького інтерфейсу за допомогою React та Material UI, після чого підключимо ці компоненти методом connect до раніше створенного спільного сховища. Отож для реалізації інтерфейсу було використано стандартизовану бібліотеку з уже створеними рішеннями для усіх моделей котрі нам потрібні, тому додаток буде відображатись коректно на усіх платформах і браузерах в котрих підтримується дана бібліотека.

Розглянемо це на прикладі створення діалогового вікна додавання нового завдання на дошку. Перш за все нам потрібну ініціалізувати потрібні функції котрі буде реалізовано на даній сторінці, приклад на скріншоті нижче:

```
//архівувати/видалити
function handleArchiveOrDeleteClick() {
    setConfirmOpen(true);
}
//зміна опису
function handleDescriptionChange(event: React.ChangeEvent<HTMLTextAreaElement>) {
    setDescription(event.target.value);
}
//зміна дати
function handleDueDateChange(date: MaterialUiPickersDate) {
    setDueDate(date);
}
//зміна статусу
function handleTaskStatusChange(event: React.ChangeEvent<{name?: string, value: unknown}>) {
    if (! event.target.name) {
        return;
    }
    setTaskStatus(event.target.value as string);
}
//зміна команди
function handleTeamOrStoryChange(event: React.ChangeEvent<{name?: string, value: unknown}>) {
    if (! event.target.name) {
        return;
    }
    setTeamOrStory(event.target.value as string);
}
```

Рис. 2.7. Функції для роботи з картками

На рисунку 2.7ми можемо бачити методи для операцій над картками, що знаходяться на дошці, функції видалення, зміни опису, дати, статусу, команди і т.д. Усі ці функції в своєму тілі викликають метди які пов’язанні із станом додатку React, тому вони повністю синхронізовані із основним сховищем. Усі ці методи потрібні для коректної роботи та синхронізації користувацького інтерфейсу із подіями у базі даних.

Розглянемо структуру сторінки на прикладі діалогу зміни картки з завданням:

```
<Dialog open={open} onClose={handleCancelClick} aria-labelledby={formDialogTitleId}>
    <DialogTitle id={formDialogTitleId} style={{paddingBottom: '0'}}>
        Змінити завдання
        <Button
            className={clsx(classes.fabDelete)}
            variant="outlined"
            color="default"
            onClick={handleArchiveOrDeleteClick} >
            {props.board.preferArchive ? <ArchiveIcon /> : <DeleteIcon color="secondary" />}
            <Typography variant="body1" color={props.board.preferArchive ? 'initial' : 'secondary'}
                style={{marginLeft: theme.spacing(1)}} >
                {props.board.preferArchive ? 'Архівувати' : 'Видалити'}
            </Typography>
        </Button>
    </DialogTitle>
    <DialogContent>
```

Рис. 2.8. Верстка зміни завдання

На рисунку 2.8ми можемо побачити верстку блоку сторінки із зміною наповнення картки із завданням. Організовано кнопку зміни, після чого відкривається новий діалог у якому можна змінювати усі параметри карток. Також можна побачити окрему кнопку для архівування/видалення вибраного завдання. Організація обробки усіх змін створена наступним чином:

```
<div>
    <TextField
        margin="dense"
        label="Опис"
        multiline
        rows={4}
        rowsMax={16}
        fullWidth
        value={description}
        onChange={handleDescriptionChange}
        helperText="Можна використовувати Markdown"
        />
</div>
<div>
    <KeyboardDatePicker
        margin="dense"
        label="Кінцева дата"
        format="yyyy-MM-dd"
        KeyboardButtonProps={{
            'aria-label': 'change date',
        }}
        value={dueDate}
        onChange={handleDueDateChange}
```

Рис. 2.9. Організація обробки змін

На рисунку 2.9представлено обробник подій для кожного із параметрів, в даному прикладі обробляються можливі зміни дати та опису вибраного користувачом завдання. Абсолютно аналогічні блок прописані для кожного елементу картки та представленні в користувацькому інтерфейсі.

Таким ж чином було створено відображення усіх інших всплиаючих вікон у різних частинах додатку. Для окремих модулів таких як календар, основна робоча зона, налаштування та кастомізація створено окремі представлення, які рендеряться при кліках на відповідні пункти меню, ось як це працює:

```
    <ListItem button
            selected={currentView === 'kanban' || currentView === ''}
            onClick={ev => handleChangeView('kanban', props.activeBoardId)}>
        <ListItemIcon><TableChartIcon /></ListItemIcon>
        {open ? <ListItemText primary="Робоча зона" /> : <></>}
    </ListItem>
    <ListItem button
            selected={currentView === 'calendar'}
            onClick={ev => handleChangeView('calendar', props.activeBoardId)}>
        <ListItemIcon><CalendarTodayIcon /></ListItemIcon>
        {open ? <ListItemText primary="Календар" /> : <></>}
    </ListItem>
    <ListItem button
            selected={currentView === 'edit'}
            onClick={ev => handleChangeView('edit', props.activeBoardId)}>
        <ListItemIcon><EditIcon /></ListItemIcon>
        {open ? <ListItemText primary="Редактор" /> : <></>}
    </ListItem>
</List>
{open ?
```

Рис. 2.10. Організація усіх пунктів меню

Після верстки усіх сторінок ми підключаємо їх до єдиного Redux store для відображення у браузері і подальшій роботі з ними.

Після створення усіх основних функцій та використанню їх на стороні користувача конфігуруємо базу даних PouchDB. Для цього реалізуємо новий файл і імпортуємо одноіменну бібліотеку ‘pouchdb’. Реалізуємо локальну змінну для ініціалізації початкової бази, створюємо усі необхідні функцій та можливість підключення до віддаленої бази CouchDB. Файл із методологією достатньо обширний, тому як приклад розглянемо синхронізацію локальної бази із віддаленою.

```
async function startSync() {
    if (remoteDb) {
        const localDocs = await localDb.allDocs({});
        const remoteDocs = remoteDb ? await remoteDb.allDocs({}) : null;
        const idSet = new Set<string>();
        for (const doc of localDocs.rows) {
            idSet.add(doc.id);
        }
        if (remoteDocs) {
            for (const doc of remoteDocs.rows) {
                idSet.add(doc.id);
            }
        }

        rep = localDb.sync(remoteDb, {
            live: true,
            retry: true,
            doc_ids: Array.from(idSet.values()),
        })
```

Рис. 2.11. Запуск та синхронізація баз даних

Даний уривок коду запускає нашу базу даних та якщо користувачом введені дані його віддаленого сервера – створює репліку та синхронізує обидві бази воєдино.Усі дані для бази зберігаються в окремому файлі у форматі JSON-файлів. Після цього створимо диспетчери обробки даних.

Диспетчери в React Redux-додатку – це функції, які відповідають за запуск дій для оновлення стану додатку. Вони діють як міст між компонентами та сховищем Redux. Диспетчери зазвичай створюються за допомогою функції dispatch, що надається сховищем Redux. Коли викликається функція диспетчера, вона надсилає дію до сховища, вказуючи конкретний тип оновлення, яке потрібно виконати. На прикладі диспетчера для дошок розглянемо як це працює.

```
export function mapDispatchToProps(dispatch: Dispatch<Action<any>>) {
    return {
        addBoard: (boardName: string) =>
            dispatch(kanbanBoardActions.startAddBoard({ boardName })),
        changeActiveBoard: (boardId: string) =>
            dispatch(kanbanBoardActions.startChangeActiveBoard({ boardId })),
        updateBoardName: (v: {boardId: string, boardName: string}) =>
            dispatch(kanbanBoardActions.startUpdateBoardName(v)),
        deleteBoard: (boardId: string) =>
            dispatch(kanbanBoardActions.startDeleteBoard({ boardId })),
```

Рис. 2.12. Код реалізації диспетчера

Як ми можемо побачити, за допомогою методу dispatch було створено механізм виклику раніше створених дій, однак замість того, щоб викликати їх в пусту, ми передаємо стан додатку після їх виклику саме в Redux Store.

В останню чергу створимо редуктори обробки даних, уже використовуючи об’єкти із бази, та будемо передавати усі зміни додатку в єдиний Redux Store.

Для цього створимо окремі файли редукторів для календаря, дошки та загальних подій додатку, аналогічно до створення дій програми. Кожний із цих файлів містить у собі велику частину коду та основну логіку програми по роботі із об’єктами бази даних. Редуктори відповідають за обробку оновлень стану на основі відправлених дій, котрі ми створили раніше. Також редуктори слугують прошарком між самими діями та нашою БД, організовуючи зв’язки та змінюючи значення по обидві сторони додатку. За допомогою оператору switch було організовано усі функції логіки додатку у трьох різних редукторах, для календаря, дошки та загального стану додатку. Приклад одного з методів редуктора для дошок описаний нижче.

```
.case(kanbanBoardActions.startDeleteBoard, (state, payload) => {
    (async () => {
        try {
            if (state.boards.length <= 1) {
                return state;
            }
            const dbBoard = await db.get<KanbanBoardDbRecord>(payload.boardId);
            if (! dbBoard) {
                return state;
            }

            const records: KanbanRecord[] = (await db.find({selector: {
                type: 'kanban',
                boardId: payload.boardId,
            }})).docs as any;

            for (const record of records) {
                await db.remove(record, {});
            }

            await db.remove(dbBoard, {});
```

Рис. 2.13. Метод видалення дошки

На прикладі методу видалення дошки розглянемо роботу редуктора, в конкретному випадку ми використовуємо раніше створену дію startDeleteBoard та отримуємо поточний стан додатку із Redux Store та стан дошки із бд CouchDB, після чого реалізуємо логіку видалення дошки. Усі ці етапи аналогічно зроблені для усіх методів додатку, котрі потребують роботи із базою даних і/або із станом додатку.

Окрім реалізації основних компонентів, які описані вище, у додатку також присутні файли валідації, реалізовано створення унікальних ключів дял кожної із дошок та записів, форматування дати та можливості кастомізації кольорової теми. В даному розділі описаний повний процес реалізації веб додатку на основі технологій React, Redux, Material UI та CouchDB.

РОЗДІЛ 3. ІНТЕРФЕЙС КОРИСТУВАЧА ВЕБ-ДОДАТКУ. ІНСТРУКЦІЯ ДЛЯ ВИКОРИСТАННЯ

3.1. Детальний огляд усіх функцій додатку

Цей пункт присвячений наданню модульного опису веб-додатку, із виділенням ключових слів – вкладок або модулів, з яких складається додаток. Кожна вкладка являє собою окремий розділ з певними функціональними можливостями та особливостями. Такий модульний підхід дає чітке розуміння структури додатку і сприяє масштабованості, підтримці та повторному використанню коду. У цьому пункті буде розглянуто чотири основні вкладки: робоча зона, календар, редактор та налаштування, а також додаткові функції.

Робоча зона

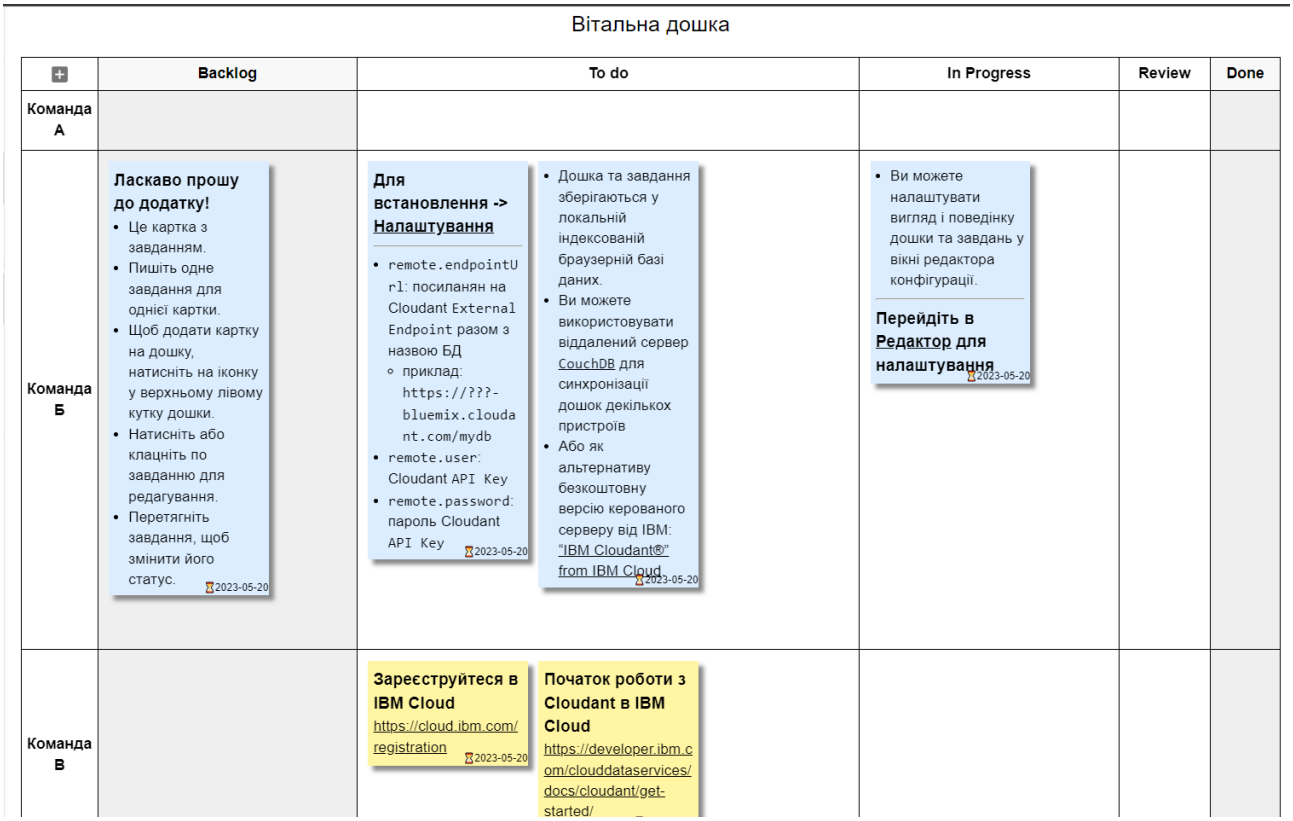

Рис. 3.1. Робоча зона додатку

На рисунку 3.1 можна спостерігати основний і найбільш важливий як по суті, так і по наповненню модуль веб-додатку. На ньому міститься

поточна дошка вибраного користувачем проєкту та відбуваються майже усі основні процеси.Вкладка надає користувачам середовище для спільної роботи над проєктами, завданнями та спілкування в команді. Скріншот демонструє основний інтерфейс робочого простору, виділяючи дошки проєктів, списки завдань і членів команди.

Основні можливості програми:

1. Додавання, видалення, редагування та архівування завдань.

Користувачі можуть легко створювати нові завдання, натиснувши кнопку "+" на вкладці Робоча зона. Ця функція дозволяє користувачам надавати важливі деталі, такі як назва завдання, опис, дата завершення, статус, категорія і спеціальні теги. Користувачі також можуть встановлювати рівні пріоритету і прикріплювати qr-коди з корисною інформацією. Можливість видаляти завдання забезпечує ефективне управління завданнями, а архівування завдань дозволяє звільнити робочий простір від безладу, зберігаючи важливі попередні дані.

2. Редагування завдань.

Вкладка "Робоча зона" пропонує широкі можливості для редагування завдань. Користувачі можуть змінювати різні атрибути завдань, включаючи назву, опис, дату завершення, статус, категорію і спеціальні мітки. Ця гнучкість дозволяє користувачам адаптувати деталі завдань відповідно до вимог проекту. Крім того, користувачі можуть додавати до завдань QR-коди, які можна відсканувати для швидкого доступу або надання додаткової інформації. Можливість писати нотатки ще більше покращує управління завданнями, дозволяючи користувачам занотовувати важливі деталі або нагадування, пов'язані з конкретними завданнями. Візуальне представлення вікна редактора можна побачити на рисунку 3.2:

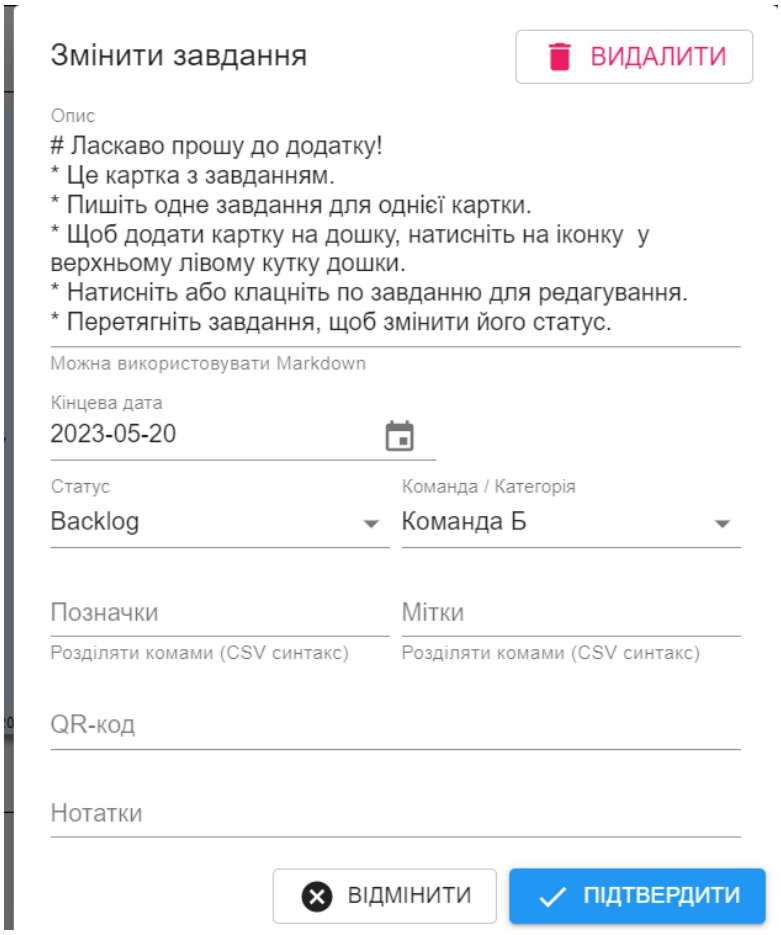

Рис. 3.2. Вікно редагування завдання

3. Редагування назви дошки.

Модуль також дозволяє користувачам редагувати назву самої дошки проекту. Ця функція гарантує, що дошки проекту можна перейменовувати, щоб відобразити зміни в обсязі проекту, цілях або будь-які інші відповідні оновлення. Користувачі можуть легко отримати доступ до налаштувань або опцій дошки і відповідно змінити назву дошки, забезпечуючи гнучкість і можливість налаштування.

4. Механізм перетягування завдань.

Для покращення організації завдань і визначення пріоритетів на вкладці "Робоча зона" реалізовано механізм перетягування. Користувачі можуть інтуїтивно впорядковувати завдання, просто перетягуючи їх в різні місця в списку завдань або між різними категоріями. Ця функція дозволяє швидко і безперешкодно змінювати порядок завдань, забезпечуючи

динамічний і ефективний процес управління завданнями. Використовуючи функцію перетягування, користувачі можуть легко адаптувати пріоритети завдань відповідно до вимог проекту, термінів або інших факторів.

Календар

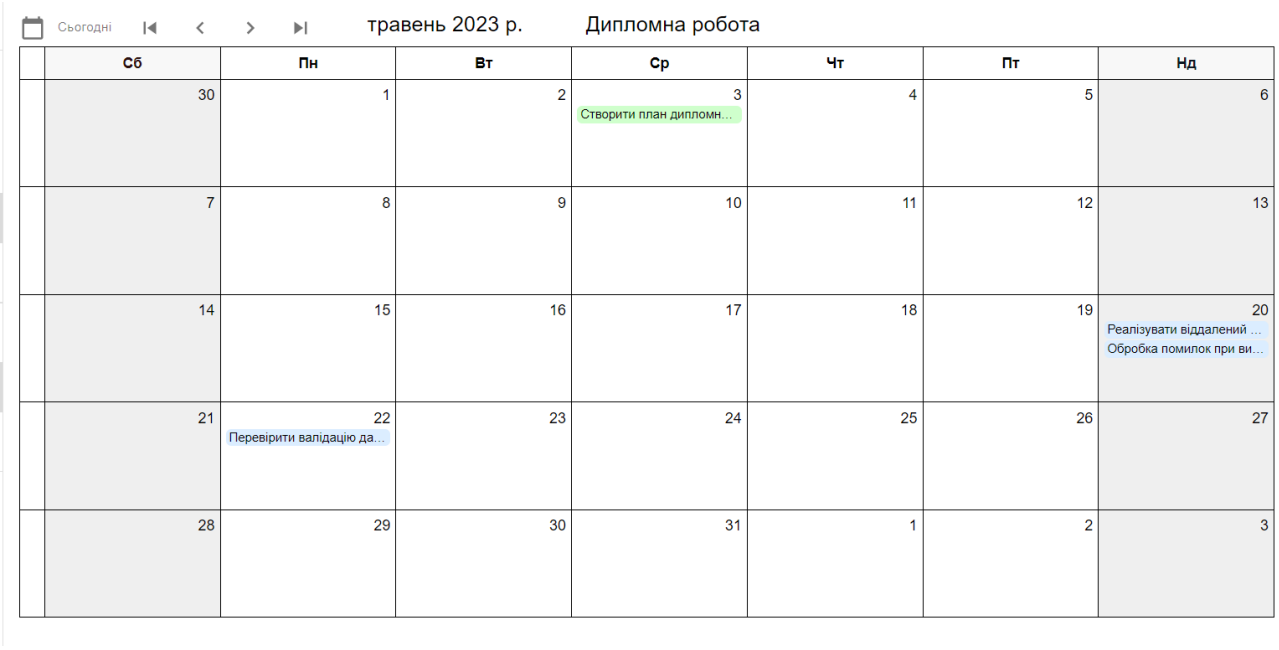

Рис. 3.3. Модуль “Календар”

Вкладка "Календар" надає візуальне представлення подій, дедлайнів та етапів, пов'язаних з проектом. На скріншоті показано місячний календар з виділенням дедлайнів завдань.

Основні можливостімодуля “Календар”:

1. Синхронізація з робочою зоною.

Вкладка "Календар" легко синхронізується з вкладкою "Робоча зона", гарантуючи, що завдання і події, створені або змінені в робочій зоні, автоматично відображаються в поданні календаря. Ця синхронізація дозволяє користувачам мати цілісне уявлення про часову шкалу проекту і легко відстежувати терміни виконання завдань, етапи та інші важливі події. Зміни, внесені на вкладці "Робоча область" або "Календар", миттєво оновлюються в обох поданнях, забезпечуючи спільну роботу в режимі реального часу і послідовний огляд проєкту.

2. Збільшена тривалість календаря.

На відміну від традиційних місячних календарів, вкладка "Календар" пропонує розширену тривалість, що дозволяє користувачам переглядати і планувати більше, ніж один місяць. Цей розширений перегляд дозволяє користувачам ефективно планувати і розподіляти ресурси, виявляти потенційні конфлікти або прогалини в розкладі і приймати обґрунтовані рішення на основі більш широкого часового горизонту.

3. Редагування завдань безпосередньо з вкладки "Календар":

Щоб спростити управління завданнями і підвищити продуктивність користувачів, вкладка "Календар" дозволяє редагувати завдання безпосередньо в межах перегляду календаря. Користувачі можуть просто натиснути на завдання або подію в календарі, щоб отримати доступ до його деталей і внести необхідні зміни. Ця функція усуває необхідність перемикатися між робочим простором і календарем, забезпечуючи безперебійний робочий процес і заощаджуючи дорогоцінний час. Завдяки редагуванню завдань на вкладці Календар, користувачі можуть ефективно оновлювати деталі завдань, коригувати терміни, призначати ресурси і підтримувати точний графік проєкту.

Поєднуючи синхронізацію з робочим простором, збільшену тривалість календаря і можливість редагувати завдання безпосередньо з календаря, вкладка "Календар" дає користувачам можливість ефективно планувати, складати графік і керувати завданнями і подіями у веб-додатку. Ця комплексна і зручна для користувача функціональність покращує координацію проєкту, вдосконалює управління часом і гарантує дотримання термінів виконання. Незалежно від того, чи це візуалізація термінів проєкту, відстеження термінів виконання завдань або внесення коригувань "на льоту", вкладка "Календар" є цінним інструментом для ефективного планування та виконання проєкту.

Редактор

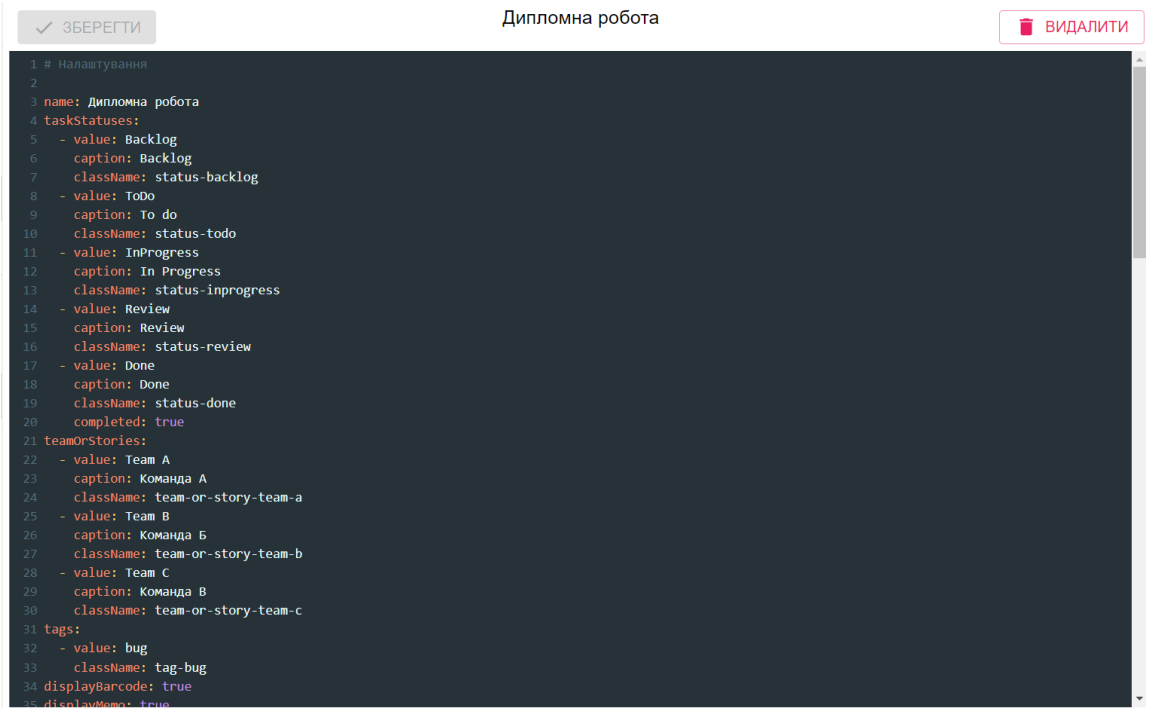

Рис. 3.4. Модуль “Редактор”

Вкладка "Редактор" пропонує багате середовище редагування для створення та співпраці з дошками і завданнями. На скріншоті показано інтерфейс текстового редактора з різними варіантами форматування.

Основні можливостімодуля “Редактор”**:**

1. Видалення дошки проєкту:

Вкладка "Редактор" надає користувачам можливість видаляти дошки проєктів, коли це необхідно. Ця функція дозволяє видаляти непотрібні або застарілі дошки, забезпечуючи вільний і організований робочий простір. Вибравши потрібну дошку проєкту та ініціювавши процес видалення, користувачі можуть ефективно керувати своїм проєктним портфоліо та підтримувати впорядковане і цілеспрямоване робоче середовище.

2. Налаштування етапів та категорій проєктів:

Щоб узгодити процес управління проєктами з індивідуальними вимогами, вкладка Редактор дозволяє користувачам налаштовувати етапи і категорії проєктів. Користувачі можуть визначати і змінювати стадії своїх

проєктів, такі як "To do", "In progress" і "Done", щоб відповідати бажаному робочому процесу. Так само користувачі можуть створювати власні категорії або мітки, щоб класифікувати завдання на основі певних критеріїв або потреб конкретного проєкту. Така кастомізація забезпечує гнучкість і адаптивність, дозволяючи користувачам пристосовувати дошку проєктів до своїх унікальних методологій управління.

3. Зміна кольорів завдань.

На вкладці "Редактор" користувачі мають можливість змінювати колір завдань, щоб посилити візуальну диференціацію і поліпшити розпізнавання завдань. Призначаючи різні кольори завданням або категоріям завдань, користувачі можуть швидко ідентифікувати і розрізняти завдання на основі їх важливості, пріоритету або будь-яких інших відповідних критеріїв. Ця функція допомагає користувачам визначати пріоритетність завдань, з першого погляду визначати критичні елементи та ефективно керувати своїм робочим навантаженням.

4. Додавання користувацьких міток.

Щоб покращити організацію та категоризацію завдань, на вкладці "Редактор" ви можете додавати власні мітки. Користувацькі мітки дозволяють користувачам призначати завданням додаткові метадані або мітки, забезпечуючи гнучкий спосіб класифікувати і фільтрувати завдання на основі певних атрибутів. Користувачі можуть створювати і застосовувати мітки, які відповідають вимогам проекту, уподобанням команди або будь-яким іншим відповідним критеріям. Ця функція полегшує ефективний пошук, фільтрацію та групування завдань, покращуючи загальні можливості управління проєктом.

Налаштування

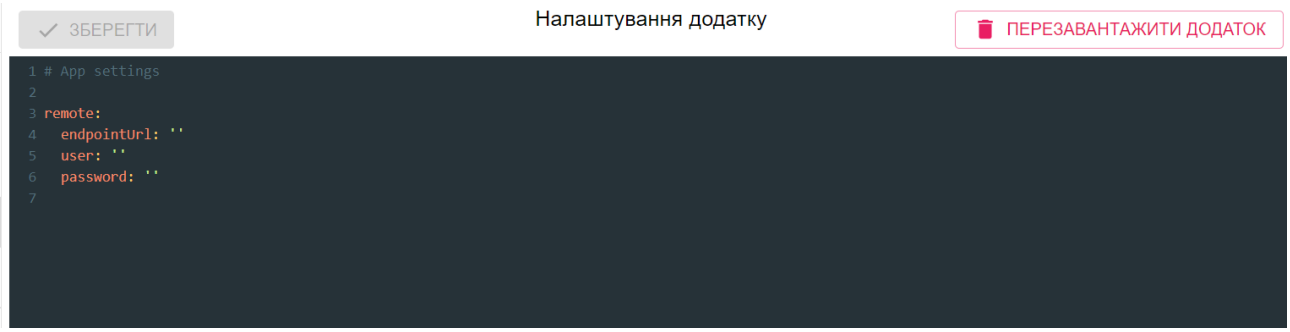

Рис. 3.5. Модуль “Налаштування”

Вкладка "Налаштування" є важливим компонентом веб-додатку, який надає користувачам можливість конфігурувати та керувати різними налаштуваннями, пов'язаними з функціональністю та підключенням додатку. Вона пропонує функції для встановлення з'єднання з сервером CouchDB та повного перезавантаження програми, якщо це необхідно.

Основні можливостімодуля “Налаштування”**:**

1. Підключення до сервера CouchDB.

У вкладці Налаштування користувачі можуть встановити з'єднання з сервером CouchDB, який слугує внутрішньою базою даних для веб-додатку. Щоб встановити з'єднання, користувачі повинні надати облікові дані для входу, включаючи ім'я користувача та пароль, а також посилання на кінцеву точку, яка вказує місце розташування сервера CouchDB. Ця функція забезпечує безпечний та автентифікований доступ до сервера, що дозволяє користувачам взаємодіяти з базою даних, отримувати дані та зберігати зміни, зроблені в додатку.

2. Повне перезавантаження програми.

У певних випадках користувачам може знадобитися повне перезавантаження програми, що призведе до стирання всіх локальних змін, зроблених у програмі. Ця функція корисна, коли користувачі стикаються з критичними проблемами або хочуть повернути програму до початкового стану. Ініціювавши повне перезавантаження програми з вкладки

"Налаштування", користувачі можуть почати з чистого аркуша, усунувши будь-які невідповідності або небажані зміни в своїх даних.

Вкладка "Налаштування" слугує центральним вузлом для керування конфігурацією програми та налаштуваннями підключення. Вона пропонує простий інтерфейс для підключення до сервера CouchDB, надаючи необхідні облікові дані для входу та посилання на кінцеву точку. Крім того, він надає засоби для перезавантаження програми, що дозволяє користувачам скинути свої дані і почати заново, якщо це необхідно. Завдяки цим функціям вкладка "Налаштування" забезпечує користувачам контроль над налаштуваннями програми, встановлює безпечні з'єднання, а також пропонує спосіб вирішення проблем і збереження цілісності даних.

ВИСНОВКИ

Отже, розробка веб-додатку для командного управління проєктами є значним кроком вперед у сучасній практиці колективної роботи. Ця комплексна і багатофункціональна програма пропонує широкий спектр функцій, які задовольняють різноманітні потреби проєктних команд. Завдяки зручному інтерфейсу, інтуїтивно зрозумілій навігації та безперешкодній інтеграції з серверними сервісами, веб-додаток забезпечує надійну платформу для ефективного планування, виконання та співпраці над проєктами.

Однією з ключових переваг цієї програми є її здатність оптимізувати управління завданнями. Завдяки таким функціям, як створення, призначення та відстеження завдань, члени команди можуть легко залишатися структурованими та визначати пріоритети своєї роботи. Додаток дозволяє призначати дедлайни, відстежувати прогрес і враховувати залежності між завданнями, гарантуючи своєчасне досягнення етапів проекту. Такий рівень наочності та контролю підвищує ефективність проєкту і дозволяє командам завчасно реагувати на будь-які вузькі місця чи затримки.

Веб-додаток також враховує потребу в гнучкості та кастомізації. Завдяки налаштованим дошкам проєктів, статусам завдань, категоріям і тегам, команди можуть адаптувати додаток до своїх унікальних методологій і робочих процесів. Така гнучкість покращує адаптацію користувачів і гарантує, що програма відповідає специфічним вимогам різних проєктів і галузей.

Загалом, розробка цього веб-додатку є значним прогрессом у розширенні можливостей організацій ефективно планувати, виконувати та відстежувати свої проекти. Використовуючи комплексний набір функцій, інтуїтивно зрозумілий інтерфейс та можливості інтеграції, проєктні команди можуть оптимізувати свою співпрацю, підвищити продуктивність та досягти

успішних результатів проєкту. Програма слугує цінним інструментом для керівників проєктів, членів команд та зацікавлених сторін, забезпечуючи централізовану платформу для оптимізації управління проектами та сприяючи розвитку культури прозорості, співпраці та постійного вдосконалення.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Командна робота. hmn.wiki. URL: https://hmn.wiki/uk/Teamwork (дата звернення: 17.06.2023).
2. Бабаєв В. М. Цифровой репозиторій ХНУГХ ім.А.Н.Бекетова. URL: https://eprints.kname.edu.ua/4464/2/Навчальний_посибник_Лекции_1-16.pdf (дата звернення: 17.06.2023).
3. Сучасні методології ведення проєктів. LoginCasino UA. URL: https://logincasino.ua/article/suchasni-metodologiiy-vedennya-proektiv (дата звернення: 17.06.2023).
4. Kaur H. Waterfall Process Operations in the Fast-paced World: Project Management Exploratory Analysis. 2021. Vol. 6, no. 1. URL: http://www.ijabms.com/wp-content/uploads/2021/05/05_ARORAL_PB.pdf (дата звернення: 18.06.2023).
5. Rasnacis A., Berzisa S. Adaptation of Agile Project Management Methodology for Project Team. Information Technology and Management Science. 2015. Vol. 18, no. 1. URL: https://doi.org/10.1515/itms-2015-0019 (дата звернення: 18.06.2023).
6. Adi P. Scrum Method Implementation in a Software Development Project Management. International Journal of Advanced Computer Science and Applications. 2015. Vol. 6, no. 9. URL: https://doi.org/10.14569/ijacsa.2015.060927 (дата звернення: 18.06.2023).
7. A statistical analysis of the effects of Scrum and Kanban on software development projects / H. Lei et al. Robotics and Computer-Integrated Manufacturing. 2017. Vol. 43. P. 59–67. URL: https://doi.org/10.1016/j.rcim.2015.12.001 (дата звернення: 18.06.2023).
8. Michael Horman & Russell Kenley. The application of lean production to project management.1996. URL: https://www.researchgate.net/publication/229021871_The_application_of_lean_production_to_project_management (дата звернення: 18.06.2023).

9. Mallon M. Project Management. Public Services Quarterly. 2015. Vol. 11, no. 2. P. 114–122. URL: https://doi.org/10.1080/15228959.2015.1026624 (дата звернення: 18.06.2023).
10. Deniz Özkan, Alok Mishra. Agile Project Management Tools: A Brief Comprative View. CYBERNETICS AND INFORMATION TECHNOLOGIES Volume 19, No 4 URL: https://doi.org/10.1109/icrito.2016.7784913 (дата звернення: 18.06.2023).
11. Ilag B. N., Sabale A. M. Microsoft Teams Overview. Troubleshooting Microsoft Teams. Berkeley, CA, 2022. P. 17–74. URL: https://doi.org/10.1007/978-1-4842-8622-7_2 (дата звернення: 18.06.2023).
12. Overview. Notion API. URL: https://developers.notion.com/docs (дата звернення: 18.06.2023).
13. Featherstone R. Basecamp. Journal of the Medical Library Association : JMLA. 2009. Vol. 97, no. 1. P. 67. URL: https://doi.org/10.3163/1536-5050.97.1.019 (дата звернення: 18.06.2023).
14. Why Should I Use Tasks & Projects in Project Management Software?. Versatile & Robust Project Management Software | Wrike. URL: https://www.wrike.com/project-management-guide/faq/why-should-i-use-tasks-projects-in-project-management-software/ (дата звернення: 18.06.2023).
15. The Total Economic Impact™ Of Microsoft 365 E3. URL: https://query.prod.cms.rt.microsoft.com/cms/api/am/binary/RE5970p (дата звернення: 18.06.2023).
16. Getting Started – React. React – A JavaScript library for building user interfaces. URL: https://legacy.reactjs.org/docs/getting-started.html (дата звернення: 19.06.2023).
17. Документація TypeScript. TypeScript: JavaScript With Syntax For Types. URL: https://www.typescriptlang.org/docs/ (дата звернення: 19.06.2023).

18. Документація Redux. | Redux. Redux - A predictable state container for JavaScript apps. | Redux. URL: https://redux.js.org/ (дата звернення: 19.06.2023).
19. Документація MUI: The React component library you always wanted. URL: https://mui.com/ (дата звернення: 19.06.2023).
20. Basic Tutorial for PouchDB and CouchDB · usePouchDB. Home | Christophers thoughts. URL: https://terreii.github.io/use-pouchdb/docs/introduction/pouchdb_couchdb (дата звернення: 19.06.2023).