

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

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

(повне найменування інституту, назва факультету)

Кафедра програмних засобів

(повне найменування кафедри)

Пояснювальна записка

до дипломного проєкту (роботи)

бакалавр

(ступінь вищої освіти)

на тему ПРОГРАМНА РЕАЛІЗАЦІЯ СИСТЕМИ УПРАВЛІННЯ
ПРОЄКТАМИ
PROGRAM REALIZATION OF PROJECT MANAGEMENT SYSTEM

Виконав: студент 4 курсу, групи КНТ-118сп
Спеціальності 121 Інженерія програмного
забезпечення

(код і найменування спеціальності)

Освітня програма (спеціалізація)

Інженерія програмного забезпечення

Філатов В.О.

(прізвище та ініціали)

Керівник Терещенко Е.В.

(прізвище та ініціали)

Рецензент Казурова А.Є.

(прізвище та ініціали)

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

Інститут, факультет ІРЕ, ФКНТ
Кафедра програмних засобів
Ступінь вищої освіти бакалавр
Спеціальність 121 Інженерія програмного забезпечення
(код і найменування)
Освітня програма (спеціалізація) Інженерія програмного забезпечення
(назва освітньої програми (спеціалізації))

ЗАТВЕРДЖУЮ

Завідувач кафедри ПЗ, д.т.н, проф.
С.О. Субботін
“ ” 2021 року

З А В Д А Н Н Я

НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА(КИ)

Філатова Віктора Олександровича

(прізвище, ім'я, по батькові)

1. Тема проєкту (роботи) Програмна реалізація системи управління проєктами.
Program Realization of Project Management System

керівник проєкту (роботи) Терещенко Еліна Валентинівна, к.ф.-м.н., доцент,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом закладу вищої освіти від “30” березня 2021 року № 103

2. Строк подання студентом проєкту (роботи) 27 травня 2021 року

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

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1. Аналіз предметної області. 2. Вибір та обґрунтування структури системи, яка проєктується. 3. Основні рішення щодо реалізації компонентів системи. 4. Керівництво програміста. 5. Керівництво оператора.

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

6. Консультанти розділів проєкту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-5 Основна частина	Терещенко Е.В., доцент		
Нормоконтроль	Липовець М. В., асистент		

7. Дата видачі завдання “27” березня 2021 року.

КАЛЕНДАРНИЙ ПЛАН

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

Студент

(підпис) Філатов В. О.
(прізвище та ініціали)

Керівник проєкту (роботи)

(підпис) Терещенко Е.В.
(прізвище та ініціали)

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи бакалавра:
98 с., 8 табл., 29 рис., 4 дод., 26 джерел.

УПРАВЛІННЯ ПРОЄКТОМ, ПРОЄКТ, ЗАВДАННЯ, ДЕЛЕГАЦІЯ
ЗАДАЧ, АВТОРИЗАЦІЯ, JS.

Об'єкт дослідження – система управління проєктами.

Предмет дослідження – створення проєктів та відповідних ним завдань.

Мета роботи – створення системи управління проєктами для конфіденціального контролю виконання проєктів та делегації задач.

Актуальність та проблематика роботи полягає в тому, що існуючі системи управління проєктами не надають можливості працювати без підключення до серверів компаній-розробників систем, що створює ризик втрати конфіденціальності даних, або втрати доступу до системи у випадку втрати доступу до мережі Інтернет.

Матеріали, методи та технічні засоби: структурне та об'єктно-орієнтоване програмування, мова програмування JS, персональний комп'ютер з процесором AMD Athlon 3500+ під управлінням операційної системи Microsoft Windows 7.

Результати. Створено програмний продукт для конфіденціального контролю виконання проєктів та делегації задач.

Висновки. Розроблено програмну систему управління проєктами для конфіденціального контролю виконання проєктів та делегації задач.

Галузь використання – даний додаток можна використовувати в підприємствах будь-якого рівня та будь-якої галузі.

ABSTRACT

Explanatory note to the final qualifying work of the bachelor: 98 pages, 8 tables, 29 figures, 4 appendixes, 26 sources.

MANAGING PROJECT, PROJECT, TASK, TASK DELEGATION, AUTHORIZATION, JS.

The object of study is project management system.

The subject of study is control of timely implementation of projects.

The purpose of the work is to create a project management system for confidential control of implementation of projects and task delegation.

The relevance and problems of the work is that existing solutions do not provide ability to use the system without connection to the servers of companies that develop these systems, which creates risk of losing confidentiality of data, or losing access to the system in case of losing access to the Internet.

Materials, methods and technical means is structural and object-oriented programming, JS programming language, personal computer with AMD Athlon 3500+ processor running Microsoft Windows 7 operating system.

Results. A software has been created that allows to confidentially control implementation of projects and delegate tasks.

Conclusions. A software system has been developed to confidentially control implementation of projects and delegate tasks.

Field of use is wide, this system can be used in any company of any size.

ЗМІСТ

	С.
Перелік скорочень та умовних позначок	8
Вступ	9
1 Аналіз предметної області	10
1.1 Основні поняття	10
1.2 Огляд існуючих систем управління проєктами	11
1.3 Огляд методів управління проєктами	12
1.4 Порівняльний аналіз існуючих методів управління проєктами	21
1.5 Постановка завдання до роботи	22
1.6 Висновок	23
2 Вибір та обґрунтування структури системи, яка проєктується	24
2.1 Вибір інструментарію розробки програмного продукту	24
2.2 Структурна схема програмного продукту	32
2.3 Опис інструмента для візуального відображення	34
2.4 Опис середовища виконання JavaScript Node.js	35
2.5 Висновок	36
3 Основні рішення щодо реалізації компонентів системи	37
3.1 Реалізація методу Agile	37
3.2 Логічна структура програми	37
3.3 Проєктування інтерфейсу	38
3.4 Висновок	38
4 Керівництво програміста	40
4.1 Призначення й умови використання	40
4.2 Характеристики програмного продукту	40
4.3 Звертання до програмного продукту	41
4.4 Повідомлення	41
4.5 Висновок	42
5 Керівництво оператора	43
5.1 Призначення й умови виконання програмного продукту	43

5.2 Виконання програмного продукту.....	43
5.3 Повідомлення оператору	48
5.4 Висновок.....	48
Висновки.....	49
Перелік джерел посилання	51
Додаток А Технічне завдання.....	53
Додаток Б Опис програмного продукту.....	58
Додаток В Текст програми	65
Додаток Д Слайди презентації	91

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

- БД – база даних;
ПБ – прізвище, ім'я, по батькові;
ПК – персональний комп'ютер;
ПП – програмний продукт;
СКБД – система керування базами даних;
СУП – система управління проектами.

ВСТУП

Одна з найважливіших частин кожного підприємства – управління проєктами, контроль виконання поставлених задач та збереження секретів робочого процесу підприємства. На сьогоднішній день існує безліч систем управління проєктами. Нажаль, через те що більшість існуючих систем надають доступ клієнтам лише через мережу Інтернет, зберігають дані клієнтів на власних серверах та не надають жодних гарантій в правилах користувань сервісами, у користувачів немає впевненості в безпеці їх даних.

Серед варіантів вирішення даної проблеми є такі як:

- шифрування даних;
- маскування даних;
- використання програм-антивірусів;
- реєстрація користувачів;

Але безпека даних знаходиться під загрозою не лише через ризик хакерської атаки, але й через ризик продажу даних третьої стороні. Це може бути як продаж даних рекламодавчим компаніям для створення прицільних рекламних кампаній, так і продаж даних компаніям-конкурентам для викриття секретів підприємств.

Мета роботи – створення системи управління проєктами для конфіденціального контролю виконання проєктів та делегації задач.

В результаті створено ПП для конфіденціального контролю виконання проєктів та делегації задач.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Основні поняття

Підприємство – це відокремлена спеціалізована господарча одиниця, основою якої є професійно організований трудовий колектив, який за допомогою засобів виробництва, що є в його розпорядженні, виготовляє необхідну споживачу продукцію, виконує роботи, надає послуги [1].

Проект – діяльність, спрямована на створення певного продукту чи послуги протягом визначеного терміну та за певних фінансових обмежень.

Завдання – наперед визначений, запланований обсяг робіт, якість доручення; те, що хочуть здійснити тощо [2].

Дані – відомості, отримані шляхом вимірювання, спостереження, логічних або арифметичних операцій і представлені у формі, придатній для постійного зберігання, передачі і (автоматизованої) обробки [3].

Управління проектами – методологія організації, планування, управління, координації трудових, фінансових і матеріально-технічних ресурсів протягом життєвого циклу проекту (проектного циклу), спрямована на ефективне досягнення його цілей шляхом застосування сучасних методів, техніки й технології управління для досягнення певних результатів у проекті щодо складу й обсягу робіт, вартості, часу, якості й задоволення учасників проекту. Це діяльність, спрямована на реалізацію проекту з максимально можливою ефективністю при заданих обмеженнях щодо часу, ресурсів і якості кінцевих результатів [4].

Токен, або JSON Web Token (в Node.js) – тимчасовий ключ користувача, що дозволяє йому працювати з модулями програми без верифікації після кожної дії. Токен видається після авторизації зареєстрованого користувача.

Система управління проектами має у собі інтерфейс призначений для запису, зміни, або читання даних про проекти та завдання з СКБД. Головною ціллю СУП має у собі поєднання контролю та координації виконання проектів та завдань.

Серед інших функцій є отримання завдань та звітування їх реалізації.

1.2 Огляд існуючих систем управління проєктами

На сьогоднішній день існує багато різних СУП, особливості деяких з них наведені в таблиці 1.1.

Таблиця 1.1 – Порівняльна характеристика існуючих СУП

Назва СУП	Особливості СУП
Asana [5]	Можливість вибору серед шести різних типів інтерфейсів: таблиця, дошка, графік, календар, лінія часу, графі. Звіти та повідомлення через електронну пошту. Третій рівень розподілу – робочий простір (workspace) що дозволяє об'єднувати проєкти в команди, створюючи ще більший рівень спеціалізації. Дані зберігаються на сервері компанії за допомогою хмарних технологій. Фільтрація звітності.
Basecamp [6]	Головна можливість – окремі чати для різних команд в великих проєктах, що дозволяє розподіляти проєкт на більш ефективні команди. Дані зберігаються на сервері компанії за допомогою хмарних технологій.
Trello [7]	Мінімалістичний дизайн дозволяє відображати всі завдання одного проєкту на одній дошці за допомогою використання однієї гнучкої картки на одне завдання. Прогрес проєкту можна оцінити без звітів лише одним поглядом на екран. Дані зберігаються на сервері компанії за допомогою хмарних технологій.
Wrike [8]	Має можливість контролювати проєкти із високим рівнем деталізованості – звіти день-в-день, інтеграція із сервісами компанії Google, розрахунок вартості проєкту до початку його виконання. Можливість створення завдань за допомогою електронної пошти. Дані зберігаються на сервері компанії за допомогою хмарних технологій.
Celoxis [9]	Деталізовані звіти, різні форми відображення інформації. Ця СУП має простий та гнучкий інтерфейс, що включає у собі відображення звітів та попереджень у формі таблиць, графіків, діаграм (кругових, та стовбичних). Дані зберігаються на сервері компанії за допомогою хмарних технологій.

З табл. 1.1 видно, що СУП можуть відрізнятися за дизайном, типом звітності, інтерфейсом та максимальною зручністю лише для команд певних розмірів.

Але всі вони мають недолік в тому, що вони дозволяють клієнтам користуватися їх сервісами лише при наявності мережі Інтернет, а також зберігають дані клієнтів на власних серверах, що створює ризик втрати секретів підприємства. Для того, щоб усунути цей недолік було вирішено створити офлайн сервіс, що використовує схожі технології, але дозволяє користувачам запускати програму на їх ПК і зберігати дані в локальній мережі.

1.3 Огляд методів управління проєктами

У випускній кваліфікаційній роботі бакалавра розглядаються різні методи управління проєктами. Головне завдання системи управління проєктами це відображення інформації про проєкти та завдання у тому вигляді, у якому це потребує сучасне підприємство, а також можливість її модифікації у тієї послідовності що відноситься до найкращого методу управління проєктами. Не існує такого методу що задовільнив би всі потреби кожного підприємства у світі, тому будуть розглянуті найбільш розповсюджені методи, і буде обраний той метод що задовольняє більшість потреб сучасного підприємства.

1.3.1 Класичний проєктний менеджмент

Найбільш очевидний шлях реалізації проєкту – розбити його на фази або окремі завдання. Найпростіший інструмент проєктного управління є чек-лист дій, які необхідно зробити для досягнення мети. Одним із інструментів, що дозволяє відстежувати часові витрати на кожен з елементів і час, коли вони повинні бути завершені є Діаграма Ганта (рис. 1.1).

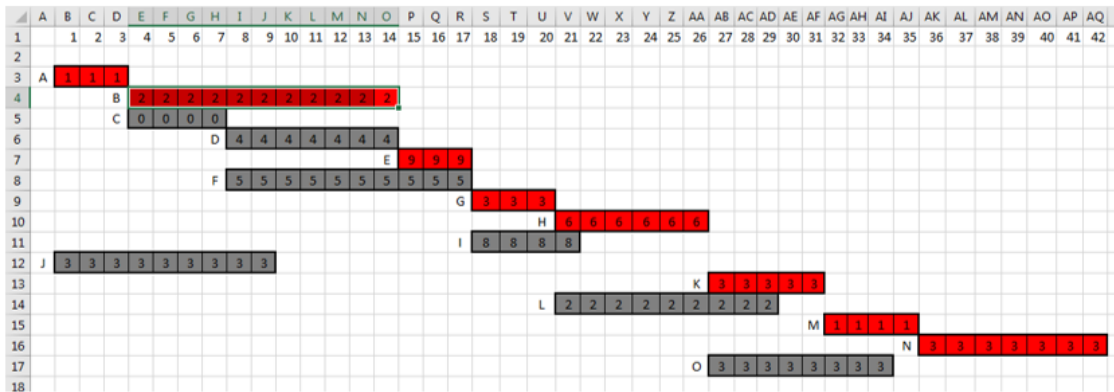


Рисунок 1.1 – Діаграма Ганта [10]

Діаграма Ганта (англ. Gantt chart, також стрічкова діаграма, графік Ганта) – це популярний тип діаграм, який використовується для ілюстрації плану, графіка робіт за будь-яким проєктом. Є одним з методів планування та управління проєктами.

Перший формат діаграми був розроблений Генрі Л. Гантом (Henry L. Gantt, 1861-1919) у 1910 році.

Діаграма Ганта являє собою відрізки (графічні плашки), розміщені на горизонтальній шкалі часу. Кожен відрізок відповідає окремому завданню. Завдання, складові плану, розміщуються по вертикалі. Початок, кінець і довжина відрізка на шкалі часу відповідають початку, кінцю і тривалості завдання. На деяких діаграмах Ганта також показується залежність між завданнями. Часто діаграма Ганта використовується спільно з таблицею зі списком робіт, рядки якої відповідають окремо взятій задачі, зображених на діаграмі, а стовпці містять додаткову інформацію про задачу.

Діаграма може використовуватися для представлення поточного стану виконання робіт: частина прямокутника, що відповідає завданню, заштриховується, відзначаючи відсоток виконання завдання; показується вертикальна лінія, що відповідає моменту «сьогодні». За всю історію проєктного управління було створено безліч різних методів управління проєктами під практично будь-які потреби. Для різних проєктів необхідні різні методи та інструменти. Головне зрозуміти, що найважливіше для проєкту –

дедлайни, ресурси, дотримання процесу, або відразу кілька факторів – а потім вибрати метод управління проектом, орієнтований на досягнення цього показника.

Перевагою даного підходу є те, що він вимагає від замовника і керівництва компанії визначити, що ж вони хочуть отримати, вже на першому етапі проекту. Раннє включення привносить певну стабільність в роботу проекту, а планування дозволяє впорядкувати реалізацію проекту. Крім того, цей підхід має на увазі моніторинг показників і тестування, що абсолютно необхідно для реальних проектів різного масштабу.

Потенційно, класичний підхід дозволяє уникнути стресів через наявність запасного часу на кожному етапі, закладеного на випадок будь-яких ускладнень і реалізації ризиків. Крім того, з правильно проведеним етапом планування, керівник проектів завжди знає, якими ресурсами він володіє. Навіть якщо ця оцінка не завжди точна.

Основна слабка сторона класичного проектного менеджменту – нетолерантність до змін. Класичний підхід зараз застосовується при реалізації будівельних та інженерних проектів, в яких зміст проекту залишається практично незмінним протягом всього проекту [10].

1.3.2 Метод Agile

Гнучкий ітеративно-інкрементальний підхід до управління проектами і продуктами, орієнтований на динамічне формування вимог і забезпечення їх реалізації в результаті постійної взаємодії всередині самоорганізованих робочих груп, що складаються з фахівців різного профілю.

Не всі проекти можуть бути структуровані таким чином, щоб бути реалізованими за класичним проектним підходом. Наприклад, приготування однієї страви ідеально лягає на «водоспадний» підхід, а ось вчасно приготувати і подати вечерю з чотирьох страв буде практично неможливо, якщо доведеться кожного разу чекати закінчення приготування однієї страви, щоб приступити

до приготування іншої. В такому випадку можна використовувати Agile. Відповідно до даного підходу, проєкт не розбивається на послідовні фази, він розбивається на маленькі підпроєкти, які потім «збираються» в готовий продукт (рис. 1.2).

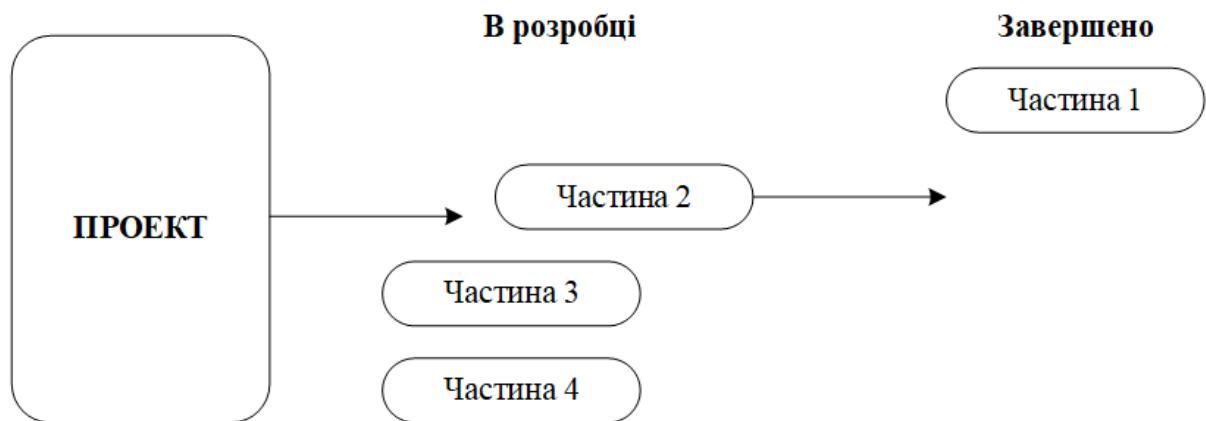


Рисунок 1.2 – Схема управління проєктом за методом Agile [11]

Таким чином, ініціація і планування проводяться для всього проєкту, а наступні етапи: розробка, тестування і інші проводяться для кожного міні-проєкту окремо. Це дозволяє передавати результати цих міні-проєктів швидше, а приступаючи до нового підпроєкту (ітерації) в нього можна додати зміни без великих витрат і впливу на інші частини проєкту.

Свою нинішню назву сімейство гнучких методологій отримало в 2001 році з публікації Маніфесту Agile (Agile Manifesto), закріпив основні цінності і принципи гнучкої розробки програмного забезпечення, в основі яких – командна робота і адаптація, навіть «любов» до змін.

Сам по собі Agile – це скоріше набір ідей і принципів того, як потрібно реалізовувати проєкти. Вже на основі цих принципів і кращих практик були розроблені окремі гнучкі методи або, як їх іноді називають, фреймворки (frameworks): Scrum, Kanban, Crystal, і багато інших. Ці методи можуть досить сильно відрізнятися один від одного, але вони використовують однакові принципи.

Сильна сторона Agile – його гнучкість і адаптивність. Він може підлаштовуватися під практично будь-які умови і процеси організації. Саме це зумовлює його нинішню популярність і то, скільки систем для різних областей було створено на його основі.

Але Agile – це набір принципів і цінностей. Слабка сторона полягає в тому, що кожній команді доведеться самотійно скласти свою систему управління, керуючись принципами Agile. Це непростий і тривалий процес, який потребує змін всієї організації, починаючи з процедур і закінчуючи базовими цінностями. Не всім організаціям він під силу [11].

1.3.3 Метод Scrum

Scrum розбиває проєкт на частини, які відразу можуть бути використані Замовником для отримання цінності, які називаються заділами продуктів (product backlog). Потім ці частини розбиваються за пріоритетом власником продукту – представником замовника в команді. Найважливіші «частини» першими відбираються для виконання в спринті (ітерації в Scrum), що тривають від 2 до 4 тижнів. В кінці спринту замовнику видається робочий інкремент продукту – найважливіші «частини», які вже можна використовувати. Наприклад, сайт з частиною функціоналу або програма, яка вже працює, нехай і частково. Після цього команда проєкту приступає до наступного спринту. Тривалість у спринті фіксована, але команда вибирає її самотійно на початку проєкту, виходячи з проєкту і власної продуктивності.

Щоб упевнитися в тому, що проєкт відповідає вимогам замовника, які мають властивість змінюватися з часом, перед початком кожного спринту відбувається переоцінка ще не виконаного змісту проєкту і внесення в нього змін. У цьому процесі бере участь вся команда проєкту. Scrum Майстер покликаний допомогти учасникам проєкту краще зрозуміти і прийняти цінності, принципи і норми практики Scrum. Він лідер і посередник між зовнішнім світом і командою. Його завдання – стежити, щоб ніхто не заважав

команді самостійно і комфортно працювати над поставленими завданнями. Команда ж відповідає за те, щоб в кінці спринту всі необхідні завдання були зроблені, а поставки виконані (рис. 1.3).



Рисунок 1.3 – Схема процесу Scrum [12]

Scrum був розроблений для проєктів, в яких необхідні «швидкі перемоги» в поєднанні з толерантністю до змін. Крім того, цей фреймворк підходить для ситуацій, коли не всі члени команди мають достатній досвід в тій сфері, в якій реалізується проєкт – постійні комунікації між членами командами дозволяють зменшити недостачу досвіду або кваліфікації одних співробітників за рахунок більш кваліфікованих колег.

Але, Scrum дуже вимогливий до команди проєкту. Вона повинна бути невеликою (5-9 чоловік) і кросфункціональною – тобто члени команди повинні володіти більш ніж однією компетенцією, необхідної для реалізації проєкту. Наприклад розробник програмного забезпечення повинен володіти знаннями в тестуванні і бізнес-аналітиці. Робиться це для того, щоб частина команди не «простоювала» на різних етапах проєкту, а також для того, щоб співробітники

могли допомагати і підміняти один одного. Крім того, члени команди повинні бути «командними гравцями», активно брати на себе відповідальність і вміти самоорганізовуватися. Підібрати таку зрілу команду дуже непросто.

Scrum підходить не для всіх команд і організацій ще й тому, що пропонуваний процес може не підійти для розробки конкретного продукту – наприклад промислового верстата або спорудження будівлі [12].

1.3.4 Метод Lean

Agile говорить нам, що необхідно розбивати на невеликі керовані пакети робіт, але нічого не говорить про те, як управляти розробкою цього пакету. Scrum пропонує нам свої процеси і процедури. Lean ж, в свою чергу, додає до принципів Agile схему потоку операцій (workflow) для того, щоб кожна з ітерацій виконувалася однаково якісно. У Lean, так само, як і в Scrum, робота розбивається на невеликі пакети поставки, які реалізуються окремо і незалежно. Але в Lean для розробки кожного пакету поставок існує потік операцій з етапами. Як і в класичному проєктному менеджменті, це можуть бути етапи планування, розробки, виробництва, тестування і поставки (рис. 1.4).



Рисунок 1.4 – Схема процесу Lean [13]

Етапи Lean і їх гнучкість дозволяють бути впевненими в тому, що кожна частина проєкту реалізується так, як потрібно. У Lean не прописані чіткі межі етапів, як в Scrum прописані обмеження спринтів. Крім того, на відміну від класичного проєктного менеджменту, Lean дозволяє паралельно виконувати кілька завдань на різних етапах, що підвищує гнучкість і збільшує швидкість виконання проєктів.

Як і Agile, Lean це скоріше концепція, образ мислення, ніж щось висічене в камені. Використовуючи ідеї Lean можна самостійно створити систему, що задовольняє вимогам в управлінні проєктами.

Якщо проєкт вимагає дуже якісного і чіткого виконання то Lean надає набір інструментів для того, щоб задовольнити ці вимоги. Lean поєднує гнучкість і структурованість, як Scrum, але в трохи іншому ключі.

Але, не кожна частина проєкту вимагає однаково детального опрацювання та уваги. Lean передбачає саме такий підхід до кожного завдання і етапу. Це основний мінус застосування Lean для великих і неоднорідних проєктів. Також Lean не пропонує чіткого робочого процесу для реалізації «частин» проєкту, що сприяє розтягуванню термінів проєкту. Ця проблема може бути вирішена за допомогою ефективного керівництва і чітких комунікацій-головне пам'ятати про це [13].

1.3.5 Метод Six Sigma

Методологія що використовується у корпоративному менеджменті для вдосконалення виробництва та усунення дефектів. Стратегічний підхід до вдосконалення бізнесу, в рамках якого проводяться заходи зі знаходження і виключення причин помилок або дефектів у бізнес-процесах, шляхом зосередження на тих вихідних параметрах, які є критично важливими для споживача.

Спочатку метод розроблявся компанією «Motorola» та її інженером Білом Смітом (з 1986 року), але зараз ним користуються багато компаній.

Значний вплив на його розвиток справили попередні концепції вдосконалення якості продукції, наприклад методології контролю якості, тотального управління якістю (Total Quality Management – TQM) та нульових дефектів.

Кінцева мета проєкту – задоволення замовника якістю продукту, якої можна досягти за допомогою безперервного процесу поліпшення всіх аспектів проєкту, заснованому на ретельному аналізі показників. У концепції 6 сигм приділяється окрема увага усуненню проблем що виникають.

Для цього було запропоновано процес з 5 кроків, відомих як DMEDI:

- визначення (Define);
- вимірювання (Measure);
- дослідження (Explore);
- розробка (Develop);
- контроль (Control).

Перевага цього методу в тому що концепція 6 сигм надає чітку схему для реалізації проєктів і постійного поліпшення процесів. Визначаючи цілі, потім ретельно аналізуючи їх (рис. 1.5).

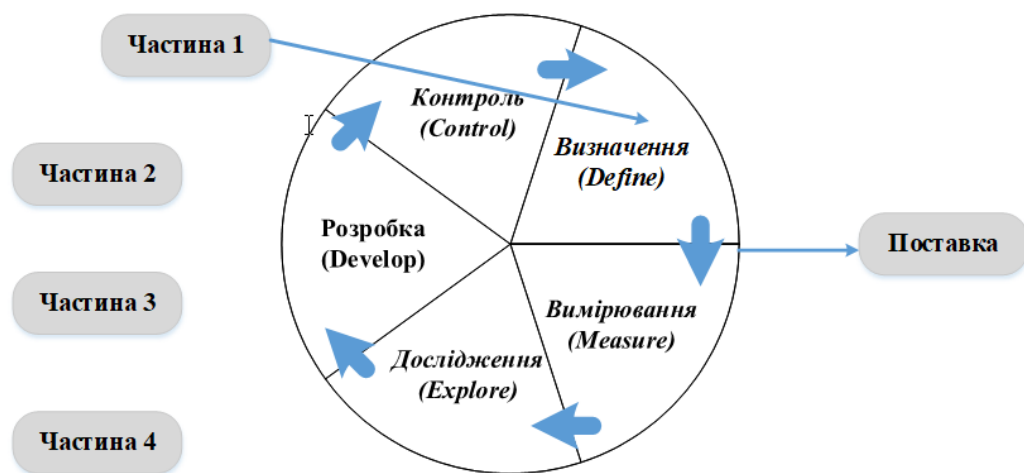


Рисунок 1.5 – Схема процесу Six Sigma [14]

Проблема 6 сигм в тому, що основною декларованою метою є зниження витрат і підвищення ефективності, але задоволення замовника часто виходить

на перший план. З огляду на деякі відмінності в цілях на різних етапах проєкту, часто у команд виникає плутанина в пріоритетах, і уникнути цього не просто.

Концепція 6 сигм надає чітку схему для реалізації проєктів і постійного поліпшення процесів. Визначаючи цілі, потім ретельно аналізуючи їх і переглядаючи отримуємо кількісні дані для більш глибокого розуміння проєкту та прийняття більш якісних рішень. І хоча збір, аналіз даних можуть зайняти певний час, це дозволить поліпшити і оптимізувати процеси реалізації проєкту і заощадити таким чином ресурси в майбутньому.

Якщо проєкт одиничний і компанія не планує в майбутньому реалізовувати подібні проєкти, всі витрати на аналіз і витяг уроків можуть виявитися марними [14].

1.4 Порівняльний аналіз існуючих методів управління проєктами

В ході дослідження та аналізу існуючих методів управління проєктами було зроблено порівняльний аналіз (табл. 1.2).

Таблиця 1.2 – Порівняльна характеристика існуючих методів управління проєктами

Можливості	Класичний проєктний менеджмент	Метод Agile	Метод Scrum	Метод Lean	Метод Six Sigma
Простота реалізації	—	+	+	—	—
Звичність для користувача	+	+	+	—	—
Зрозумілість	+	+	+	+	—

Продовження таблиці 1.2

Можливості	Класичний проєктний менеджмент	Метод Agile	Метод Scrum	Метод Lean	Метод Six Sigma
Гнучкість	—	+	+	—	—
Популярність	+	+	+	—	—
Можливість використання підприємствами будь-якого розміру	+	+	—	—	—
Стабільність	+	—	+	+	+

Як видно з таблиці 1.2, жоден з методів управління проєктами не відповідає всім необхідним вимогам підприємств. Однією з причин є два пункти що протирічать один одному – гнучкість та стабільність. Метод Agile є одним з найгнучкіших методів управління проєктами, але через те що він не має чітких обмежень, його застосування може змінюватись від підприємства до підприємства. Цей метод може бути гарним базисом для підприємств будь-якого розміру. Класичний проєктний менеджмент та метод Six Sigma є занадто застарілими, але із цим вони надають впевнену стабільність. Нажаль, у сучасному світі гнучкість більш важлива за стабільність, саме тому ці два методи не будуть використані.

Методи Lean та Scrum доволі непогані, відповідають великій кількості необхідних вимог, але вони мають більшу спеціалізованість ніж метод Agile, а тому підходять меншій кількості підприємств.

З табл. 1.2 видно, що метод Agile відповіде кожній вимозі, але не має стабільності. Це є як недоліком, так і перевагою. В сучасному світі є великий попит на гнучкість та миттєву адаптацію до нових ринкових умов. Отже, для того, щоб виконати поставлене завдання було вирішено використати метод Agile.

1.5 Постановка завдання до роботи

В даній роботі досліджується проблема створення СУП що дозволить підприємствам використовувати метод Agile або його варіації.

Для виконання поставленої мети потрібно виконати наступні завдання:

- створити метод для створення проєктів;
- створити метод для реєстрації та авторизації користувачів;
- створити метод для створення завдань;
- створити можливість для делегації завдань;
- створити можливість для встановлення дат початку та кінця завдань.

1.6 Висновок

В даному розділі було виконано дослідження предметної області. Було розглянуто та визначено основні поняття, які пов'язані з обраною предметною областю.

Також були проаналізовані та досліджені існуючі методи, кожен з яких можливо застосовувати для вирішення поставленої задачі, було визначено їх переваги та недоліки.

Був проведений порівняльний аналіз досліджених методів, які можуть бути використані для створення СУП.

Було сформульовано і визначено завдання до роботи.

2 ВИБІР ТА ОБҐРУНТУВАННЯ СТРУКТУРИ СИСТЕМИ, ЯКА ПРОЄКТУЄТЬСЯ

2.1 Вибір інструментарію розробки програмного продукту

Для розробки ПП необхідно обрати найкращі засоби розробки, що відповідатимуть всім вимогам поставленого завдання.

Основна задача ПП – це програмна реалізація системи управління проєктами для конфіденціального контролю виконання проєктів та делегації задач.

ПП повинен містити в собі такі модулі:

- зберігання даних;
- обчислювальний;
- вводу-виводу даних;
- авторизації.

Для проєктування програмного продукту треба обрати мову програмування, середовище розробки, СКБД та засіб створення інтерфейсу.

2.1.1 Вибір мови програмування

Для того, щоб створити систему управління проєктами для конфіденціального контролю виконання проєктів та делегації задач, треба проаналізувати низку сучасних мов програмування та зробити порівняння їх характеристик, і на основі цього порівняння зробити вибір на користь однієї з мов програмування для використання неї для розробки програмного продукту. Далі буде наведено опис мов програмування та визначено їх найголовніші переваги та недоліки.

Для того щоб проаналізувати найсучасніші мови програмування, необхідно визначити які це мови, і чи мають вони релевантність сьогодні та потенціал для розвитку.

Найпопулярнішими мовами програмування є наступні:

- JavaScript (18,1%);
- Java (14,7%);
- C# (14,3%);
- Python (12,1%);
- PHP (10,1%);
- TypeScript (6,9%);
- C++ (4,6%);
- Swift (3,1%);
- Ruby (3%) [15].

Як можна побачити, долю мов якими користуються програмісти вище ніж 10% мають лише п'ять мов програмування: JavaScript, Java, C#, Python, та PHP.

PHP (англійською Hypertext Preprocessor — гіпертекстовий препроцесор, попередня назва: Personal Home Page Tools — засоби для особистої домашньої сторінки) — мова програмування, створена для породження HTML-сторінок зі сторони веб-сервера.

PHP є однією з найпоширеніших мов, що використовуються у сфері веб-розробок разом із Java, .NET, Perl, Python і Ruby.

Веб-сервер інтерпретує код PHP у код HTML, який передає клієнту. Також код PHP можна вбудовувати безпосередньо в код html. На відміну від мови JavaScript, користувач не бачить PHP-коду, бо браузер клієнта отримує готовий html-код.

Переваги PHP:

- легкий у навчанні;
- легкий процес обслуговування, адже коли хтось використовує PHP, тоді ця людина може легко здійснити процес оновлення, якщо це потрібно, оскільки в основному PHP має властивості інтерпретатора.;
- відкритий код, тобто функціями цієї мови можна користуватися безкоштовно, оскільки вона є відкритим кодом. Таким чином, користувачі можуть вільно користуватися PHP без оплати зборів.

Недоліки PHP:

- висока вразливість через природу відкритості вихідного коду;
- менший престиж, так як PHP є високорівневою мовою програмування і доволі легкою в вивченні, існує багато низькоякісних за стосунків створених за допомогою цієї мови [16].

Python – інтерпретована об'єктно-орієнтована мова програмування високого рівня зі строгою динамічною типізацією.

Переваги Python:

- Python працює майже на всіх відомих операційних системах, включаючи: Linux/UNIX, Window, Macintosh, Solaris, macOS, iPhone OS, Palm OS, Windows Mobile, Symbian та Android.... Дозволяє в режимі реального часу взаємодіяти з інтерпретатором, не вимагає компіляцій для виконання коду;
- мова програмування Python досить проста у вивченні;
- синтаксис Python чіткий і стислий. Мова розроблена так, щоб вона була читабельною і була близькою до фактичної англійської, що дозволяє легко її розшифрувати. Python також вимагає менше рядків коду для досягнення результатів порівняно з мовами, такими як C або Java.;
- Python надає чудову масштабованість, адже неможливо передбачити коли кількість користувачів почне зростати, а масштабованість стане пріоритетом.

Недоліки Python:

- низька швидкість;
- відсутність статичної типізації;
- неможливість модифікувати вбудовані класи [17].

C# це об'єктно-орієнтовна мова програмування. C# була створена як нова мова на принципах C++, але з перевагами мови Java. Синтаксис цієї мови дуже схожий на синтаксис мов програмування C++ та Java, і ці дві мови є найбільшими конкурентами мові C#. Підтримує перевантаження операторів, літератори та анонімні функції.

В мові C# існує також має статичну типізацію, підтримує об'єктно-орієнтоване програмування та має систему збору сміття.

Переваги C#:

- затребуваний і перспективний;
- має високі показники безпеки;
- лаконічний синтаксис;
- є повністю об'єктно-орієнтованою мовою програмування;
- сі-шарп відмінно підходить для швидкого написання настільних додатків з зручним інтерфейсом.

Недоліки C#:

- створено компанією Майкрософт;
- необхідність компілювати код кожного разу при будь-яких змінах в ньому;
- можливості кросплатформенності нижчі за мови-конкуренти [18].

Java є типобезпечною мовою. Типобезпека гарантує відсутність у програмах помилок, які важко знайти і які пов'язані з неправильною інтерпретацією пам'яті комп'ютера. Це робить процес розробки надійнішим і передбачуваним, а отже швидшим. Так само це дозволяє залучати до розробки програмістів, що мають меншу кваліфікацію і мати великі групи розробників.

Переваги Java:

- повноцінне збирання сміття;
- кросплатформенність;
- чіткі стандарти.

Недоліки Java:

- необхідна віртуальна машина;
- низька швидкодія;
- реалізація високоякісної людино-машинної взаємодії може бути проблематичною[19].

Як і у всіх мовах програмування, JavaScript має певні переваги та недоліки. Багато з них пов'язані з тим, як JavaScript часто виконується

безпосередньо в браузері клієнта. Але є й інші способи використання JavaScript, які дозволяють йому мати ті ж переваги мов на стороні сервера.

Переваги JavaScript:

- простота – синтаксис JavaScript був натхненний Java і його порівняно легко вивчити порівняно з іншими популярними мовами, такими як C ++;
- JavaScript має тенденцію бути дуже швидким, оскільки він часто запускається відразу в браузері клієнта. Поки він не вимагає зовнішніх ресурсів, JavaScript не сповільнюється через виклики до серверного сервера. Крім того, всі основні браузери підтримують компіляцію JIT (вчасно) для JavaScript, що означає, що немає необхідності компілювати код перед його запуском;
- JavaScript працює на стороні клієнта, тому загалом зменшує попит на сервери, а простим програмам сервер може взагалі не знадобитися;
- є багато способів використовувати JavaScript через сервери Node.js. Наприклад Node.js за допомогою Express, використовувати базу даних документів, як MongoDB, і використовувати JavaScript на інтерфейсі для клієнтів, можна розробити цілу програму JavaScript, використовуючи лише JavaScript.

Недоліки JavaScript:

- оскільки код JavaScript виконується на стороні клієнта, помилки та недогляди іноді можуть бути використані для зловмисних цілей. Через це деякі люди вирішують повністю вимкнути JavaScript;
- хоча сценарії на стороні сервера завжди дають однакові результати, різні браузери іноді інтерпретують код JavaScript по-різному. У наш час відмінності мінімальні, і ця проблема не проявляється якщо проводити тестування в основних браузерах[20].

Під час дослідження різних мов програмування було створено порівняльну характеристику, яка наведена у табл. 2.1.

Таблиця 2.1 – Порівняння мов програмування

Критерії порівняння	JavaScript	PHP	Python	Java	C#
Кросплатформеність	+	–	+	+	+
Швидкодія	+	–	–	+	+
Простота вивчення	+	+	+	–	–
Простота розробки	+	+	+	–	–
Кросмовна взаємодія	+	+	+	+	+
Можливість використання додаткових бібліотек	+	+	+	+	+

Як видно з табл. 2.1. та порівняльного аналізу, мова програмування JavaScript відповідає усім вимогам до поставленої задачі, тому для розробки програмного продукту було обрано мову програмування JavaScript.

2.1.2 Вибір середовища розробки

Для досягнення максимальної ефективності при програмній реалізації системи управління проєктом, треба обрати середовище розробки, яке буде максимально реалізовувати функції та переваги обраної мови програмування.

Для мови програмування JavaScript підходить декілька середовищ розробки. Так само як і з порівнянням існуючих мов програмування, необхідно порівняти існуючі середовища розробки, виділити основні переваги та недоліки, і обрати найкращу з них.

Однією з найпопулярніших середовищ розробки є Microsoft Visual Studio розробленою всесвітньо відомою компанією Microsoft. Вона має низкою корисних функцій, що дозволить реалізувати всі вимоги до даного ПП.

Переваги Microsoft Visual Studio:

- безкоштовні видання для навчання;
- має офіційну підтримку багатьох мов програмування;
- підтримка платформи .NET.

Недоліки Microsoft Visual Studio:

- інтерфейс не завжди є зручним;
- великі і не зовсім потрібні зміни від версії до версії, що може зробити програму непрацездатною в новіших або старіших версіях;
- створена компанією Microsoft [21].

Eclipse є інтегрованим середовищем розробки від компанії Eclipse Foundation. Воно орієнтоване для розробки програмних продуктів на мові програмування Java, але також підтримує мову програмування C#. Використовується для розробки модульних програмних продуктів.

Переваги Eclipse:

- доволі проста для використання;
- повний функціонал повністю безкоштовний;
- завдяки різноманітності плагінів тонко налаштовується і розширюється додатковими функціями.

Недоліки Eclipse:

- займає велику кількість ваги;
- інколи можуть виникати великі проблеми із зворотною сумісністю, адже багато додаткових плагінів є неофіційними;
- функція налагодження програми не вдосконала, тут Eclipse сильно поступається своїм конкурентам;
- логування може розширити вже велику програму ще більше, а також процес логування може бути непотрібним;
- має занадто низьку швидкодію [22].

IntelliJ IDEA — комерційне інтегроване середовище розробки для різних мов програмування (Java, Python, Scala, PHP та інші) від компанії JetBrains.

Переваги IntelliJ IDEA:

- проста для використання після вивчення;
- має доволі високу кількість вузькоспеціалізованих функцій;
- підтримує велику кількість мов програмування.

Недоліки IntelliJ IDEA:

- незручний інтерфейс;
- невелика популярність, має низьку кількість додаткових плагінів;
- частина функцій доступна лише в платній версії;
- працює повільно [23].

Під час дослідження різних середовищ розробки було створено порівняльну характеристику, яка представлена в табл. 2.2.

Таблиця 2.2 – Порівняльна характеристика середовищ розробки

Критерії порівняння	Microsoft Visual Studio	IntelliJ IDEA:	Eclipse
Кросплатформенність	+	+	+
Низький поріг входження	+	–	+
Велика кількість безкоштовних функцій	+	–	+
Простота в використанні	+	–	+
Можливість безкоштовного використання	+	+	+
Надійність	+	–	–
Швидкодія	+	–	+
Економія ресурсів ПК	+	–	+

В табл. 2.2 можна побачити, що середовище розробки Microsoft Visual Studio підходить по всіх ключових параметрах, що дозволить провести програмну реалізацію системи управління проєктами на найвищому рівні, і при цьому зберегти ресурси, як фінансові так і фізичні ресурси ПК. Також Microsoft Visual Studio було створено доволі давно, і ця середа розробки має підтримку великої компанії, що створює відчуття впевненості в стабільності

працездатності програми. Саме тому Microsoft Visual Studio є найкращим вибором.

2.2 Структурна схема програмного продукту

ПП повинен містити в собі такі модулі:

- зберігання даних;
- реєстрація та авторизація;
- обробка даних;
- вводу-виводу даних.

В якості вхідних даних ПП повинен використовувати інформацію з бази даних, що містить в собі інформацію про проекти та їх відповідні завдання.

Обране сховище для зберігання даних повинно бути надійним, надавати швидкий доступ до даних, мати можливість для зберігання великої кількості даних та їх зручної модифікації. Під ці параметри найкраще підходить такий тип зберігання даних як JSON із використанням MongoDB.

JSON (або джсон) – це текстовий формат обміну даними між комп'ютерами. JSON базується на тексті, може бути прочитаним людиною. Формат дає змогу описувати об'єкти та інші структури даних. Найголовнішою перевагою джсон є легка масштабованість. При програмній реалізації СУП необхідно взяти до уваги той факт, що БД в такій програмі дуже швидко зростає, при цьому одна з можливостей програми – швидкий доступ до минулих проектів та завдань потребує швидкого аналізу, запису, та читання даних з БД. Саме для цього я вирішив використовувати джсон, а не його конкурент XML.

Для зберігання даних програмного продукту було вирішено використати СКБД що працює на основі JSON – MongoDB.

MongoDB зберігає дані у вигляді JSON-документів, згрупованих в «колекції». В даному форматі можна зберігати будь-які JSON-документи та зручно категоризувати їх по колекціям. Збережений в MongoDB JSON-документ називається двійковим JSON або BSON і, як і будь-який інший

документ цього формату є неструктурованим. Ось чому, на відміну від традиційних СКБД, в колекціях можна зберігати відомості будь-якого вигляду, і ця гнучкість поєднується з горизонтальним масштабуванням бази даних. Багатьом розробникам імпонує така можливість, проте не все так однозначно [24].

Під час проєктування БД було вирішено визначити сутності, які входять в БД та їх атрибути такі як назва атрибута, тип, обов'язковість заповнення та розмір. Результати представлені в табл. 2.3.

Таблиця 2.3 – Сутності та їх атрибути

Сутність	Атрибут	Тип	Розмір	Обов'язкове	Опис сутності
Акаунт	Код_акаунта	Лічильник	Довге ціле	Так	Містить інформацію про акаунти
	Логін	Текстовий	255	Так	
	Пароль	Текстовий	255	Так	
	Ім'я	Текстовий	255	Так	
Проєкт	Код_проєкта	Числовий	Довге ціле	Так	Містить інформацію про проєкти
	Назва_проєкта	Текстовий	255	Так	
	Опис_проєкта	Текстовий	255	Ні	
	Власник_проєкта	Текстовий	255	Так	
Завдання	Код_завдання	Лічильник	Довге ціле	Так	Містить інформацію про завдання
	Відповідний_проєкт	Текстовий	255	Так	
	Дата_початку	Дата	Довге ціле	Так	
	Дата_кінця	Дата	Довге ціле	Так	
	Відповідний_виконавець	Текстовий	255	Так	

Як можна побачити в табл. 2.3 визначено декілька сутностей, а саме сутність Акаунт, яка містить в собі назву проєкта, сутність Проєкт, яка містить інформацію про користувачів даної СУП, сутність Завдання, яка містить інформацію про завдання, дату початку і кінця, а також пріоритет та опис, і найголовніше –відповідний даному завданню проєкт.

2.3 Опис інструмента для візуального відображення

Так як даний продукт відображатиметься за допомогою браузера, для відображення даних необхідно створити шаблони CSS та HTML.

Існує безкоштовний набір інструментів для цього – Bootstrap.

Bootstrap – це безкоштовний набір інструментів з відкритим кодом, призначений для створення веб-сайтів та веб-додатків, який містить шаблони CSS та HTML для типографіки, форм, кнопок, навігації та інших компонентів інтерфейсу, а також додаткові розширення JavaScript. Він спрощує розробку динамічних веб-сайтів і веб-додатків.

Bootstrap – це клієнтський фреймворк, тобто інтерфейс для користувача, на відміну від коду серверної сторони, який знаходиться на сервері. Репозиторій із цим фреймворком є одним із найпопулярніших на GitHub. Серед інших, його використовує NASA.

Bootstrap має модульну структуру і складається переважно з наборів таблиць стилів LESS, які реалізують різні компоненти цього набору інструментів. Розробники можуть самостійно налаштовувати файли Bootstrap, обираючи компоненти для свого проєкту.

Основні інструменти Bootstrap:

- сітки (grid) — наперед задані, готові до використання колонки;
- шаблони (template) — фіксовані чи адаптивні шаблони сторінок;
- типографіка (typography) — опис та визначення класів для шрифтів, таких як шрифти для коду, цитат тощо;

- мультимедіа (media) — засоби управління зображеннями та відео;
- таблиці (table) — засоби оформлення таблиць, які зокрема забезпечують сортування;
- форми (form) — класи для оформлення як форм, так і деяких подій
- навігація (nav, navbar) — класи для оформлення вкладок, сторінок, меню і панелей навігації;
- сповіщення (alert) — класи для оформлення діалогових вікон, підказок і спливаючих вікон;
- іконочний шрифт (icon font) — набір іконок у вигляді шрифту, складається майже з 500 компонентів [25].

2.4 Опис середовища виконання JavaScript Node.js

Node.js – середовище виконання JavaScript, побудоване на рушії V8. Node.js використовує подійно-орієнтовану модель з неблокуючим введенням/виведенням, що робить його легким і ефективним.

Іншими словами: Node.js дає можливість писати сервери за допомогою JavaScript з неймовірною продуктивністю. Як сказано у документації: Node.js - середовище виконання, яке використовує той самий рушій V8 Javascript, який можна знайти в браузері Google Chrome. Але це не достатньо для успіху Node.js – Node.js використовує libuv, мультиплатформенну бібліотеку з фокусом на асинхронний I/O.

З точки зору розробника Node.js є однопотоким - але libuv обробляє потоки, file system івенти, імплементує event loop і т.д. У більшості випадків розробник не взаємодіє з цим безпосередньо. Завдяки цьому, розробка є доволі легкою, при цьому використовуються усі можливі потужності.

З моменту Node.js Foundation був сформований, Node.js має план релізу. Це дуже схоже на інші проєкти Linux Foundation. Це означає, що є два релізи: стабільний реліз і експериментальний. У Node.js стабільні версії з

довгостроковою підтримкою (LTS) є ті, які починаються з парних чисел (4, 6, 8 ...), а експериментальні версії - непарні числа (5, 7 ...) [26].

Для програмної реалізації СУП я використовуватиму стабільний реліз.

2.5 Висновок

В даному розділі були розглянуті різні мови програмування та середовища розробки, було проведено точний порівняльний аналіз, виділено основні переваги і недоліки та обрано найкращі засоби для рішення завдання. В ході цієї роботи було встановлено, що для програмного продукту найбільш ефективною мовою програмування є JavaScript, завдяки низькій перевазі та мінімальній кількості недоліків. Також найкращим середовищем було обрано середовище Microsoft Visual Studio, так як воно підтримує розробку мовою JavaScript, а також не потребує великих ресурсів ПК, а також дозволяє отримати безкоштовний доступ до обширних функцій.

Також було розроблено структурну схему програмного продукту. Для зберігання вхідних даних було обрано найкращий тип сховища, а саме за стосунок MongoDB.

Інтерфейс програмного продукту було вирішено розробляти за допомогою Bootstrap.

3 ОСНОВНІ РІШЕННЯ ЩОДО РЕАЛІЗАЦІЇ КОМПОНЕНТІВ СИСТЕМИ

В ході виконання роботи розглядається проблема програмної реалізації системи управління проєктами. Цю проблему можна вирішити реалізувавши метод Agile.

Завдання роботи полягає в програмній реалізації системи управління проєктами.

Для того, щоб виконати поставлене завдання треба проаналізувати методи СУП та обрати найкращий метод, а також визначити спосіб створення інтерфейсу.

3.1 Реалізація методу Agile

Для реалізації методу Agile необхідно забезпечити виконання головного принципу цього методу:

- можливість створити завдання на будь-яку тему для будь-якого проєкту;
- необмеженість створення, видалення, або зміни завдань;
- делегація завдань будь-якому користувачу;
- незалежність кожного окремого завдання від інших завдань;

Для реалізації даних функцій немає необхідності в створенні жодних додаткових функцій, адже звичайна реалізація роботи БД забезпечить роботу ключових функцій.

3.2 Логічна структура програми

При запуску програми завантажуються головний модуль програми App.js + middleware. В першу чергу проводиться перевірка наявності записів в БД та її

працездатність, після чого проводиться перевірка працездатності всіх модулів і створюється токен користувача

Головний модуль ПП контролює роботу з БД: запис, читання, перевірку, та видалення даних.

Серед інших модулів є: Config, UserTypes, Users, Tasks, Projects.

Модуль Config керує налаштуваннями інших модулів.

Модуль Users, Tasks, Projects, UserTypes керують відповідно: користувачами, завданнями, проєктами, та типами.

Кожна сторінка програми має стилі css та логічні файли типу .js

Для того щоб звернутися до відповідних модулів, необхідно перейти в директорію client/src/components та вибрати директорію чия назва відповідає назві модуля. В цій директорії будуть логічні файли типу .js а також файли стилів css.

3.3 Проєктування інтерфейсу

Для розробки графічного інтерфейсу ПП було використано фреймворк Bootstrap, що має низку шаблонів css та HTML.

Для встановлення даного фреймворку було використано програму-пакетний менеджер npm, що дозволяє завантажувати файли з git-репозиторія.

В програмі npm (після переходу до теки з файлами ПП) було введено дві команди:

```
clone https://github.com/twbs/bootstrap
```

```
npm install bootstrap
```

Після цього в ПП стало можливо використовувати шаблони css та HTML від Бутстрап.

3.4 Висновок

В ході виконання даного ПП було програмно реалізовано систему управління проєктами. Було використано усі обрані засоби для виконання поставленої задачі.

4 КЕРІВНИЦТВО ПРОГРАМІСТА

4.1 Призначення й умови використання

Розроблений ПП дозволяє проводити управління проектами та завданнями.

Рекомендовані системні вимоги для даного ПП:

- операційна система Microsoft Windows 7;
- 64-розрядна версія операційної системи;
- центральний процесор з тактовою частотою не менше 2,4ГГц;
- 1 ГБ оперативної пам'яті;
- 512 МБ вільного місця на жорсткому диску.

4.2 Характеристики програмного продукту

ПП містить в собі такі модулі:

– зберігання даних. Для зберігання даних застосовується СКБД MongoDB;

– вводу-виводу даних. Цей модуль розбитий на декілька, і розташований в наступних файлах: `tasks.js`, `types.js`, `users.js`, `projects.js`. Цей модуль знаходиться в директорії `/server/routes/`;

– авторизації. Цей модуль блокує доступ до БД за допомогою необхідності наявності логіну та паролю, а також дозволяє розмежувати доступ до програмного продукту і відокремлювати одного користувача від іншого. Для того, щоб внести зміни в модуль авторизації необхідно внести зміни напряму в файл `auth.js` в директорії `/server/routes/`;

– верифікації. Цей модуль розташований в файлі `verifyToken.js` в директорії `/server/routes/`. Для того щоб уникнути необхідності постійної верифікації користувачів, існує токени – одноразові ключі що надаються користувачам при успішній авторизації, і існують весь сеанс що користувач працює із ПП.

4.3 Звертання до програмного продукту

Для того, щоб встановити програму треба завантажити архів SUP, провести деархівізацію. Встановити наступні додатки на ПК де планується розташовувати БД: MongoDB, GitBash.

Після цього, в залежності від розташування деархівізованого ПП, за допомогою GitBash, перейти в директорії server та client, та виконати команди `npm install` та `npm start`.

Провести такі самі дії на комп'ютерах користувачів (окрім директорії server). Переконалися що ПК користувача має доступ до сервера.

При коректному налаштуванні СКБД Mongo, після виконання команди `npm start` на ПК користувача, користувач зможе використовувати даний ПП.

4.4 Повідомлення

Внаслідок роботи ПП та певних дій користувача виникати помилки. ПП обробляє всі помилки, а також відображує відповідні повідомлення, для того, щоб користувач мав можливість їх виправити. Повідомлення про помилки, що генеруються ПП описані в табл. 4.1.

Таблиця 4.1 – Повідомлення про помилки в програмі «SUP»

Повідомлення	Опис
Заповніть необхідні поля	Повідомлення в модулях реєстрації та авторизації, та створенні завдань та проєктів що є незаповнені обов'язкові поля
Помилка при введенні даних	Повідомлення в модулях реєстрації та авторизації, та створенні завдань та проєктів що деякі поля були заповнені з некоректними даними

4.5 Висновок

В даному розділі було описано призначення та умови використання розробленого ПП, визначено рекомендовані системні вимоги, при яких робота ПП буде можливою. Вказано дії, які потрібно виконати для запуску ПП. Було описано характеристики ПП, в яких визначено модулі та їх функції. Виконано опис вхідних та вихідних даних, а також перелік повідомлень для користувача, які ПП генерує при обробці помилок, що користувач може робити під час роботи.

5 КЕРІВНИЦТВО ОПЕРАТОРА

5.1 Призначення й умови виконання програмного продукту

Розроблений ПП дозволяє проводити управління проектами та завданнями.

Рекомендовані системні вимоги для даного ПП:

- операційна система Microsoft Windows 7;
- 64-розрядна версія операційної системи;
- центральний процесор з тактовою частотою не менше 2,4ГГц;
- 1 ГБ оперативної пам'яті;
- 512 МБ вільного місця на жорсткому диску.

5.2 Виконання програмного продукту

Після запуску програми, відкривається браузер та пересилається на першу сторінку, що є сторінкою авторизації (рис 5.1).

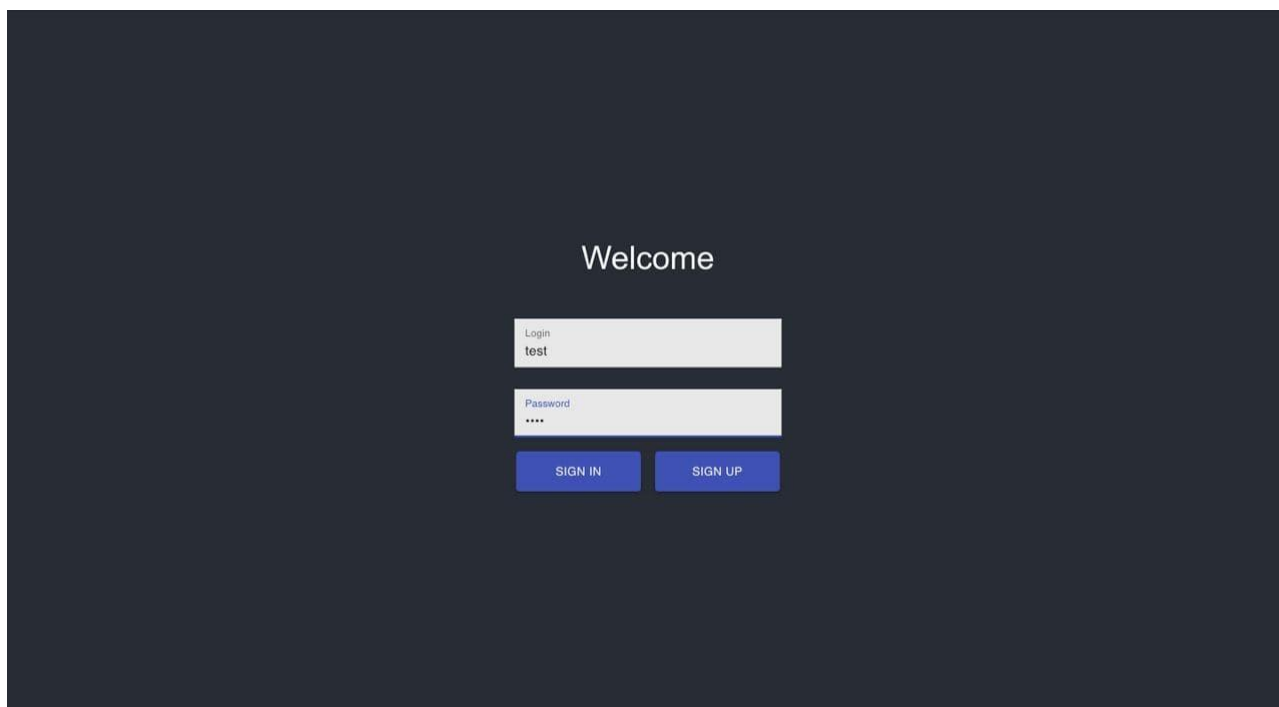


Рисунок 5.1 – Вікно авторизації ПП

Для вдалої авторизації необхідно коректно заповнити поля «Логін» та «Пароль». Після цього програма відправляє запит до БД для пошуку користувача з цим логіном, а також порівнює пароль до паролю цього користувача. У випадку, якщо даний профіль існує і пароль вірний, користувач переходить до наступного вікна. Якщо пароль не вірний, то користувач отримує повідомлення про помилку (детальніше в розділі 5.3 в таблиці 5.1). Якщо користувач з таким логіном не існує, то необхідно зареєструвати такого користувача. Щоб перейти до вікна реєстрації необхідно натиснути на кнопку «Sign up» (рис. 5.2).

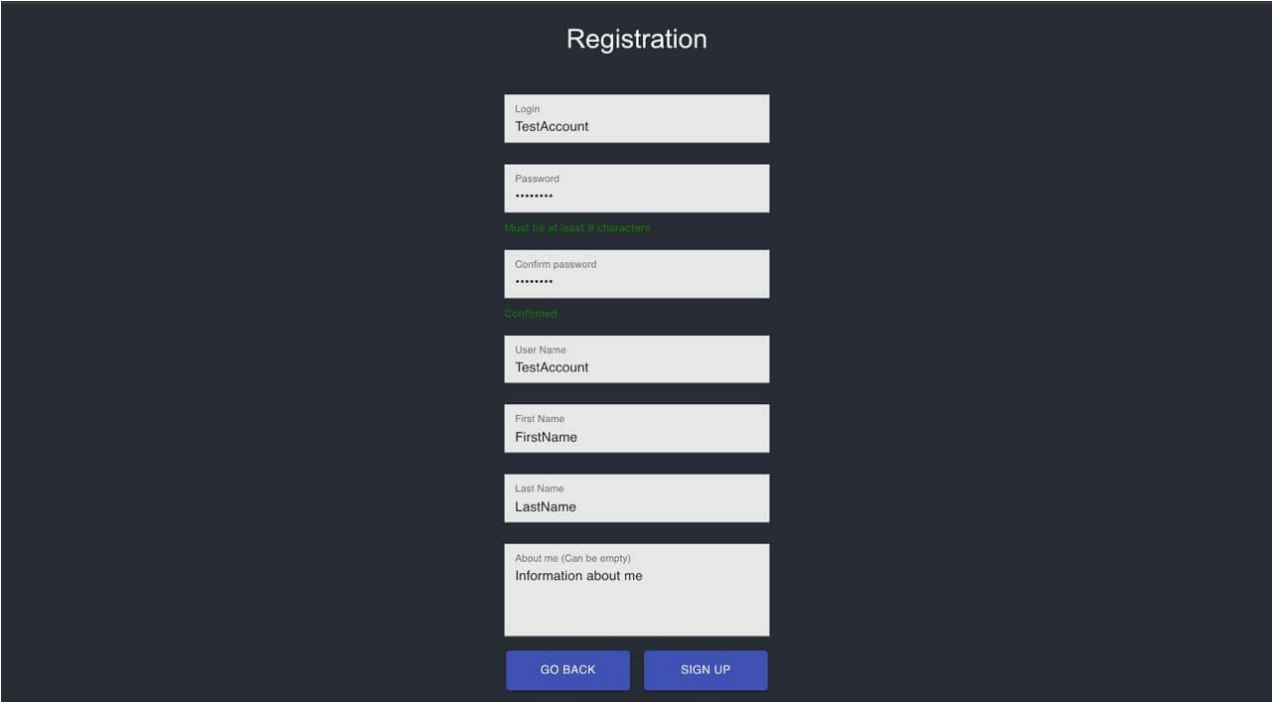
The image shows a registration form titled "Registration" on a dark background. The form consists of several input fields: "Login" with the value "TestAccount", "Password" with masked characters "*****" and a green error message "Must be at least 8 characters", "Confirm password" with masked characters "*****" and a green "Confirmed" message, "User Name" with the value "TestAccount", "First Name" with the value "FirstName", and "Last Name" with the value "LastName". There is also a text area labeled "About me (Can be empty)" with the placeholder text "Information about me". At the bottom of the form are two buttons: "GO BACK" and "SIGN UP".

Рисунок 5.2 – Вигляд сторінки «Реєстрація»

Для реєстрації необхідно обов'язково заповнити всі поля (окрім позначених «Can be empty»). Кожне поле має обмеження по типу символів що можуть бути введені, при їх введені буде зображено повідомлення про помилку (детальніше в розділі 5.3 в таблиці 5.1). Після коректного заповнення всіх полів, необхідно натиснути кнопку «Sign up». Якщо користувач перейшов до цієї сторінки випадково – існує кнопка «Go back», що дозволяє повернутися до

вікна авторизації. Після авторизації, користувач переходить до списку проєктів (рис. 5.3).

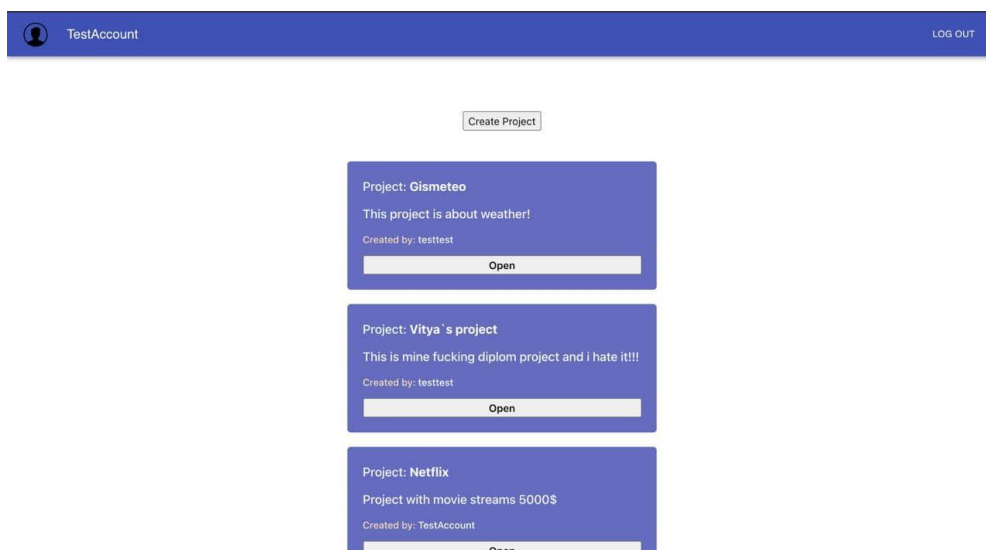


Рисунок 5.3 – Вигляд головної сторінки ПП «Проекти»

Якщо користувач помилково увійшов до некоректного аканту, або якщо користувач завершив роботу, існує кнопка «Log out». На сторінки проєктів користувач може створити новий проєкт або переглянути існуючі проєкти. Для перегляду існуючого проєкту необхідно натиснути на кнопку «Open», а для створення нового – «Create project» (рис 5.4).

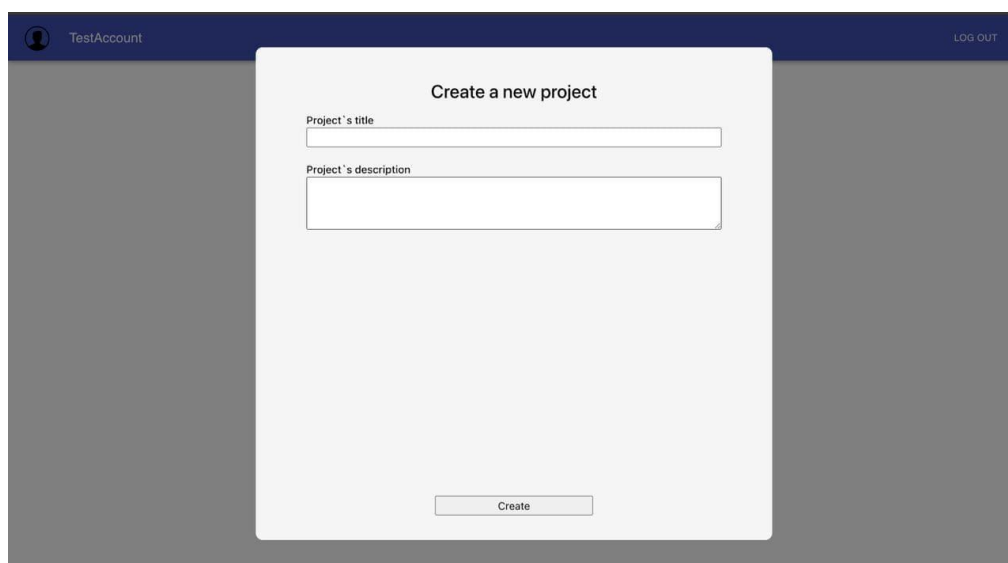


Рисунок 5.4 – Вигляд сторінки «Create project»

Для створення нового проєкту необхідно заповнити поля назви та опису проєкту, після чого натиснути кнопку «Create» (рис 5.5).

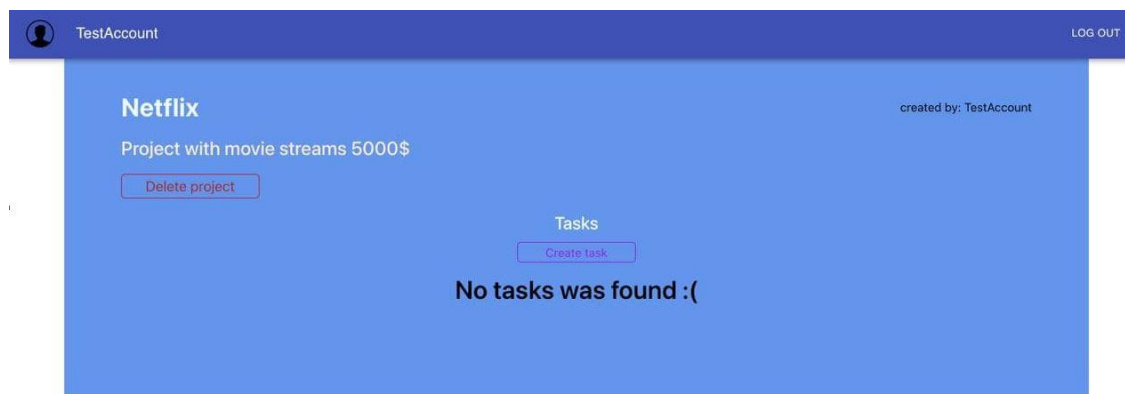


Рисунок 5.5 – Вигляд сторінки опису проєкта

На сторінці опису проєкта ми бачимо кнопки «Delete project» та «Create task». Перша кнопка видаляє проєкт, а за допомогою другої ми переходимо на сторінку створення завдання (рис 5.6).

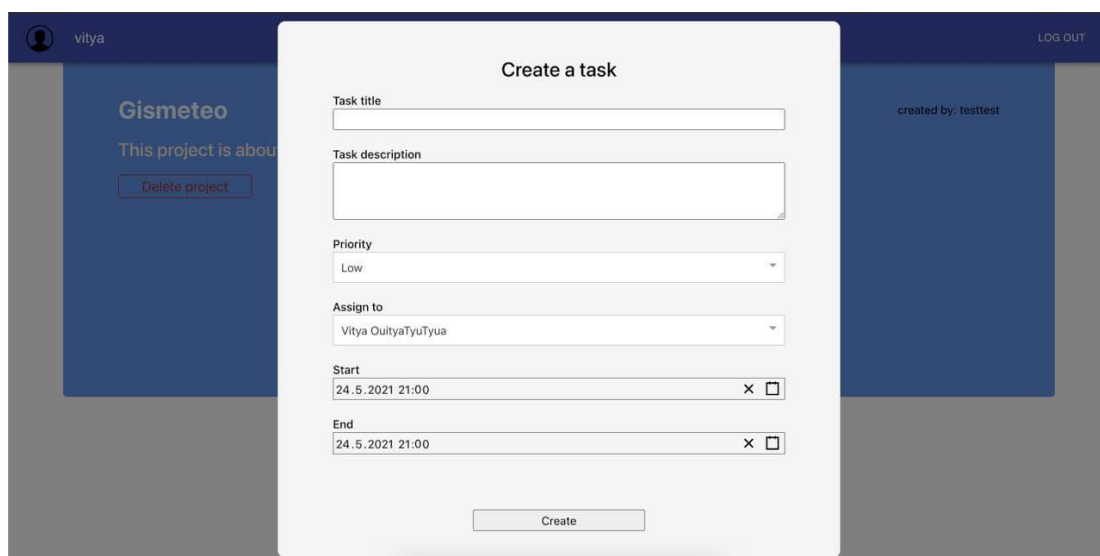


Рисунок 5.6 – Вигляд сторінки створення завдання

На сторінці створення завдання необхідно заповнити всі поля, серед яких є поля що дозволяють призначити завдання на іншого користувача, та

встановлення дат початку та кінця завдання, а також створення пріоритету. Серед інших сторінок є сторінка редагування профілю (рис 5.7).

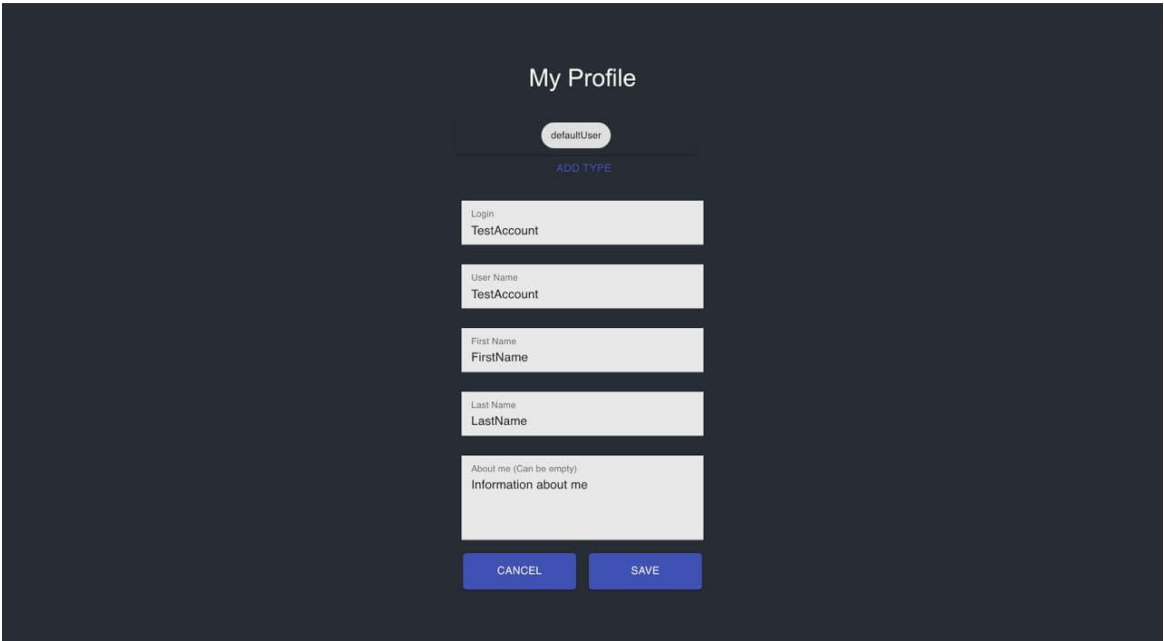


Рисунок 5.7 – Вигляд сторінки редагування профілю

На сторінці редагування профілю ми можемо змінити існуючі дані про користувача.

5.3 Повідомлення оператору

Внаслідок роботи ПП та певних дій користувача виникати помилки. ПП обробляє всі помилки, а також відображу відповідні повідомлення, для того, щоб користувач мав можливість їх виправити. Повідомлення про помилки, що генеруються ПП описані в табл. 5.1.

Таблиця 5.1 – Повідомлення про помилки в програмі «SUP»

Повідомлення	Опис
Помилка при введенні даних	Повідомлення що деякі поля були заповнені з некоректними даними
Заповніть необхідні поля	Повідомлення що є незаповнені обов’язкові поля

5.4 Висновок

В даному розділі було описано призначення та умови використання розробленого ПП, визначено рекомендовані системні вимоги, при яких робота ПП буде можливою. Вказано дії, які потрібно виконати для запуску ПП. Було описано послідовність дій з ПП та результати його роботи. Також в даному розділі було визначено перелік повідомлень для користувача, які ПП генерує при обробці помилок, що користувач може робити під час роботи.

ВИСНОВКИ

При створенні диплома було виконано необхідні завдання для вирішення поставленої проблеми. Для цього було досліджено предметну область та розглянуто та визначено основні поняття.

Для реалізації ПП було проаналізовано методи систем управління проектами, проведено оцінку їх переваг та недоліків. Було обрано найкращий метод для створення універсальної системи управління проектами. Було сформульовано завдання до роботи.

Метою даної дипломної роботи є створення системи управління проектами для конфіденціального контролю виконання проєктів та делегації задач.

Для виконання поставленої мети було виконано наступні завдання:

- створити метод для створення проєктів;
- створити метод для реєстрації та авторизації користувачів;
- створити метод для створення завдань;
- створити можливість для делегації завдань;
- створити можливість для встановлення дат початку та кінця завдань.

При створенні цієї дипломної роботи було розглянуто актуальні мови програмування та середовища розробки ПП. Було проведено їх аналіз, виділено основні переваги і недоліки у вигляді таблиці та на основі цього обрано найкращі. В ході цієї роботи було встановлено, що для програмного продукту найбільш ефективною мовою програмування є JavaScript, завдяки її гнучкості та актуальності. Microsoft Visual Studio підходить по всіх ключових параметрах, що дозволить провести програмну реалізацію системи управління проектами на найвищому рівні, і при цьому зберегти ресурси, як фінансові так і фізичні ресурси ПК. Також Microsoft Visual Studio було створено доволі давно, і ця середовище розробки має підтримку великої компанії, що створює відчуття впевненості в стабільності працездатності програми. Саме тому Microsoft Visual Studio є найкращим вибором.

Для роботи було розроблено структурну схему програмного продукту. Також, для зберігання даних було використано СКБД MongoDB, що дозволяє обробляти велику кількість даних за короткий час.

Інтерфейс програмного продукту створено за допомогою стандартних шаблонів Bootstrap.

В результаті було програмно реалізовано систему управління проєктами. Створений ПП дозволяє проводити управління проєктами та завданнями, делегувати завдання та встановлювати дати початку та кінця, зберігати дані в безпеці та ін.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1. Нестеренко В.Ю., Токар І.І. Конспект лекцій / В.Ю. Нестеренко – Харків, 2012. – 8с.
2. Як найчастіше помиляються журналісти [Електрон. ресурс] – Режим доступу: <https://medialab.online/news/pryjniaty/>
3. Інформатика – Access mode:
<https://stud.com.ua/97162/informatika/dani>
4. Онлайн тлумачний словник [Електрон. ресурс] – Режим доступу:
<http://slovopedia.org.ua/32/53399/30829.html>
5. Asana: pros and cons and alternatives [Electronic resource] – Access mode: <https://project-management.com/asana-pros-and-cons-with-3-asana-alternatives/>
6. Work better, together [Electronic resource] – Access mode:
<https://basecamp.com/how-it-works>
7. Getting started with Trello [Electronic resource] – Access mode:
<https://trello.com/guide/trello-101>
8. Agile enterprise [Electronic resource] – Access mode:
<https://www.wrike.com/>
9. The Celoxis story [Electronic resource] – Access mode:
<https://www.celoxis.com/about-us>
10. Старченко Г. В. Управління проєктами: теорія та практика: навч. посіб. / Г.В. Старченко – Чернігів, 2018. – 65 с.
11. Старченко Г. В. Управління проєктами: теорія та практика: навч. посіб. / Г.В. Старченко – Чернігів, 2018. – 72 с.
12. Старченко Г. В. Управління проєктами: теорія та практика: навч. посіб. / Г.В. Старченко – Чернігів, 2018. – 75 с.
13. Старченко Г. В. Управління проєктами: теорія та практика: навч. посіб. / Г.В. Старченко – Чернігів, 2018. – 76 с.
14. Старченко Г. В. Управління проєктами: теорія та практика: навч.

посіб. / Г.В. Старченко – Чернігів, 2018. – 92 с.

15. Опит: популярність мов програмування [Електрон. ресурс] – Режим доступу: <https://dou.ua/lenta/articles/language-rating-jan-2021/>
16. Сильні та слабкі сторони PHP, які вам потрібно знати [Електрон. ресурс] – Режим доступу: <https://altitudetvm.com/uk/komputer/1467-9-kelebihan-dan-kekurangan-php-yang-perlu-anda-ketahui.html>
17. Цікавий Python [Електрон. ресурс] – Режим доступу: <http://www.itschool.vn.ua/interesting-python/>
18. Джеффрі Ріхтер CLR via C#. Програмування на платформі Microsoft.NET Framework 4.5 мовою C# / Джеффрі Ріхтер Київ, 2016. –686 с
19. Порівняння C++ з мовами Java та C# [Електрон. ресурс] – Режим доступу: <http://cprpm12.blogspot.com/2018/05/java-i-c.html>
20. Переваги та недоліки JavaScript [Електрон. ресурс] – Режим доступу: <https://hackit-ukraine.com/627-the-advantages-and-disadvantages-of-javascript>
21. Visual Studio 2022 [Electronic resource] – Access mode: <https://habr.com/ru/company/microsoft/blog/553442/>
22. Кросплатформена розробка на Eclipse [Електрон. ресурс] – Режим доступу: <https://ukr.kagutech.com/4108944-eclipse-a-modular-cross-platform-application-development-environment>
23. IntelliJ IDEAD [Electronic resource] – Access mode: <https://www.jetbrains.com/idea/>
24. MongoDB use cases [Electronic resource] – Access mode: <https://www.mongodb.com/use-cases>
25. Стандартні засоби Bootstrap [Електрон. ресурс] – Режим доступу: <https://tokar.ua/tag/bootstrap>
26. Node.js [Electronic resource] – Access mode: <https://codeguida.com/post/517>

ДОДАТОК А
Технічне завдання

Вступ

Створений ПП застосовується для того, щоб управляти проєктами і конфіденційно контролювати виконання проєктів та делегації задач за методом Agile.

ПП «SUP» має функцію авторизації та реєстрації користувачів, дозволяє створювати проєкти, створювати завдання в проєктах, делегувати завдання іншим користувачам, слідкувати за здійсненням завдань та проєктів.

A.1 Мета розробки і її призначення

Об'єкт дослідження – система управління проєктами.

Предмет дослідження – створення проєктів та відповідних ним завдань.

Мета роботи – створення системи управління проєктами для конфіденційного контролю виконання проєктів та делегації задач.

Актуальність та проблематика роботи полягає в тому, що існуючі системи управління проєктами не надають можливості працювати без підключення до серверів компаній-розробників систем, що створює ризик втрати конфіденційності даних, або втрати доступу до системи у випадку втрати доступу до мережі Інтернет.

A.2 Стислий зміст основних теоретичних посилок до розробки на основі проведених пошуків

Одна з найважливіших частин кожного підприємства – управління проєктами, контроль виконання поставлених задач та збереження секретів робочого процесу підприємства. На сьогоднішній день існує безліч систем управління проєктами. Нажаль, через те що більшість існуючих систем надають доступ клієнтам лише через мережу Інтернет, зберігають дані клієнтів на

власних серверах та не надають жодних гарантій в правилах користувань сервісами, у користувачів немає впевненості в безпеці їх даних.

А.3 Підстави для розробки

Підставою для виконання розробки є завдання на дипломну роботу бакалавра на тему «Програмна реалізація системи управління проєктами», затверджене наказом по НУ «Запорізька політехніка» № 103 від 30 березня 2021 року.

А.4 Основні вимоги до програми, що розробляється, початковим даним і результату

А.4.1 Вимоги до функціональних характеристик

Вимоги до функціональних характеристик:

- повинна бути можливість для створення проєктів;
- повинна бути можливість для реєстрації та авторизації користувачів;
- повинна бути можливість для створення завдань;
- повинна бути можливість для делегації завдань;
- повинна бути можливість встановлення дат початку та кінця завдань.

А.4.2 Вимоги до надійності

ПП повинен відповідати таким вимогам до надійності:

- захищати ПП від несанкціонованого доступу;
- захищати ПП від введення некоректних даних користувачем.

ПП повинен функціонувати без частих програмних збоїв та помилок. У разі виникнення проблем із роботою програми, нормальна робота ПП може бути відновлена після перезавантаження серверу.

A.4.3 Умови експлуатації

Умови експлуатації:

- встановлення серверної частини ПП потребує базового знання системної адміністрації;
- для того щоб використовувати клієнтську частину ПП не потрібна додаткова кваліфікація.

A.4.4 Вимоги до складу та параметрів технічних засобів

Рекомендовані системні вимоги для даного ПП:

- операційна система Microsoft Windows 7;
- 64-розрядна версія операційної системи;
- центральний процесор з тактовою частотою не менше 2,4ГГц;
- 1 ГБ оперативної пам'яті;
- 512 МБ вільного місця на жорсткому диску.

A.4.5 Вимоги до маркування та упакування

Програмне забезпечення не потребує додаткових дій при маркуванні та упакуванні.

A.4.6 Вимоги до транспортування та збереження

Вимоги до транспортування й збереження є аналогічні вимогам, які висуваються до носія, на якому записаний ПП.

A.5 Вимоги до програмної документації

Склад програмної документації повинен включати в себе:

- технічне завдання;
- керівництво програміста;
- керівництво оператора;
- опис програми.

A.6 Основні обмеження на установку і використання програмного продукту

Даний ПП потребує інсталяції на сервер. Програмний продукт рекомендовано використовувати тільки на тих комп'ютерах, які задовільняють рекомендованим вимогам та параметрам технічних засобів. Програмний продукт не конфліктує з іншими програмними продуктами та його використання може здійснюватись користувачем лише при влюченому сервері.

ДОДАТОК Б
Опис програмного продукту

Б.1 Загальні відомості

Назва програмного продукту – «SUP».

Створений ПП застосовується для того, щоб вирішити проблему створення СУП що дозволить підприємствам використовувати метод Agile або його варіації.

ПП «SUP» має функцію авторизації та реєстрації користувачів, дозволяє створювати проєкти та завдання, встановлювати дату початку та кінця завдань, а також делегувати завдання іншим користувачам.

Рекомендовані системні вимоги для даного ПП:

- операційна система Microsoft Windows 7;
- 64-розрядна версія операційної системи;
- центральний процесор з тактовою частотою не менше 2,4ГГц;
- 1 ГБ оперативної пам'яті;
- 512 МБ вільного місця на жорсткому диску.

Б.2 Функціональне призначення

Вимоги до функціональних характеристик:

- повинна бути можливість для створення проєктів;
- повинна бути можливість для реєстрації та авторизації користувачів;
- повинна бути можливість для створення завдань;
- повинна бути можливість для делегації завдань;
- повинна бути можливість встановлення дат початку та кінця завдань.

Б.3 Опис логічної структури

ПП повинен містити в собі такі модулі:

- зберігання даних;
- верифікації;

- вводу-виводу даних;
- авторизації.

Опис сутностей БД та їх атрибути такі як назва атрибута, тип, обов'язковість заповнення та розмір представлені в табл. Б.1.

Таблиця Б.1 – Сутності та їх атрибути

Сутність	Атрибут	Тип	Розмір	Обов'язкове	Опис сутності
Акаунт	Код_акаунта	Лічильник	Довге ціле	Так	Містить інформацію про акаунти
	Логін	Текстовий	255	Так	
	Пароль	Текстовий	255	Так	
	Ім'я	Текстовий	255	Так	
Проект	Код_проекта	Числовий	Довге ціле	Так	Містить інформацію про проекти
	Назва_проекта	Текстовий	255	Так	
	Опис_проекта	Текстовий	255	Ні	
	Власник_проекта	Текстовий	255	Так	
Завдання	Код_завдання	Лічильник	Довге ціле	Так	Містить інформацію про завдання
	Відповідний_проект	Текстовий	255	Так	
	Дата_початку	Дата	Довге ціле	Так	
	Дата_кінця	Дата	Довге ціле	Так	
	Відповідний_виконавець	Текстовий	255	Так	
	Опис_завдання	Текстовий	255	Так	
	Пріоритет	Текстовий	255	Так	

Як можна побачити в табл. Б.1 визначено декілька сутностей, а саме сутність Акаунт, яка містить в собі назву проєкта, сутність Проєкт, яка містить інформацію про користувачів даної СУП, сутність Завдання, яка містить інформацію про завдання, дату початку і кінця, а також пріоритет та опис, і найголовніше –відповідний даному завданню проєкт.

Наступною важливою складовою програмного продукту є графічний інтерфейс, за допомогою якого користувач взаємодіє з програмою.

Основною функцією графічного інтерфейсу є можливість переглядати інформацію про проєкти та завдання.

Основним модулем є модуль авторизації. Цей модуль блокує доступ до БД за допомогою необхідності наявності логіну та паролю, а також дозволяє розмежувати доступ до програмного продукту і відокремлювати одного користувача від іншого.

Модуль вводу-виводу даних дозволяє отримувати дані про проєкти та завдання, їх опис, дати початку та кінцю, їх назви, а також ім'я користувача на якого призначено завдання.

Важливим модулем також є модуль верифікації, що дозволяє користувачу взаємодіяти із ПП без необхідності постійної авторизації.

Б.4 Вибір інструментарію розробки програмного продукту

Для розробки ПП було обрано мову програмування JavaScript, тому що вона має високу швидкодію, високу популярність, стабільність, широкий вибір різноманітних бібліотек та низькі вимоги до ПК.

Для розробки ПП було обрано середовище програмування Microsoft Visual Studio, завдяки широкому вибору безкоштовних бібліотек та функцій. Для зберігання даних програмного продукту була обрана використати СКБД MongoDB.

Б.5 Виклик і завантаження

Для того, щоб встановити програму треба завантажити архів SUP, провести деархівізацію. Встановити наступні додатки на ПК де планується розташовувати БД: MongoDB, GitBash.

Після цього, в залежності від розташування деархівізованого ПП, за допомогою GitBash, перейти в директорії server та client, та виконати команди `npm install` та `npm start`.

Провести такі самі дії на комп'ютерах користувачів (окрім директорії server). Переконалися що ПК користувача має доступ до сервера.

Б.6 Приклади реалізації

Після запуску програми, відкривається браузер та пересилається на першу сторінку, що є сторінкою авторизації, яке представлено на рис Б.1.

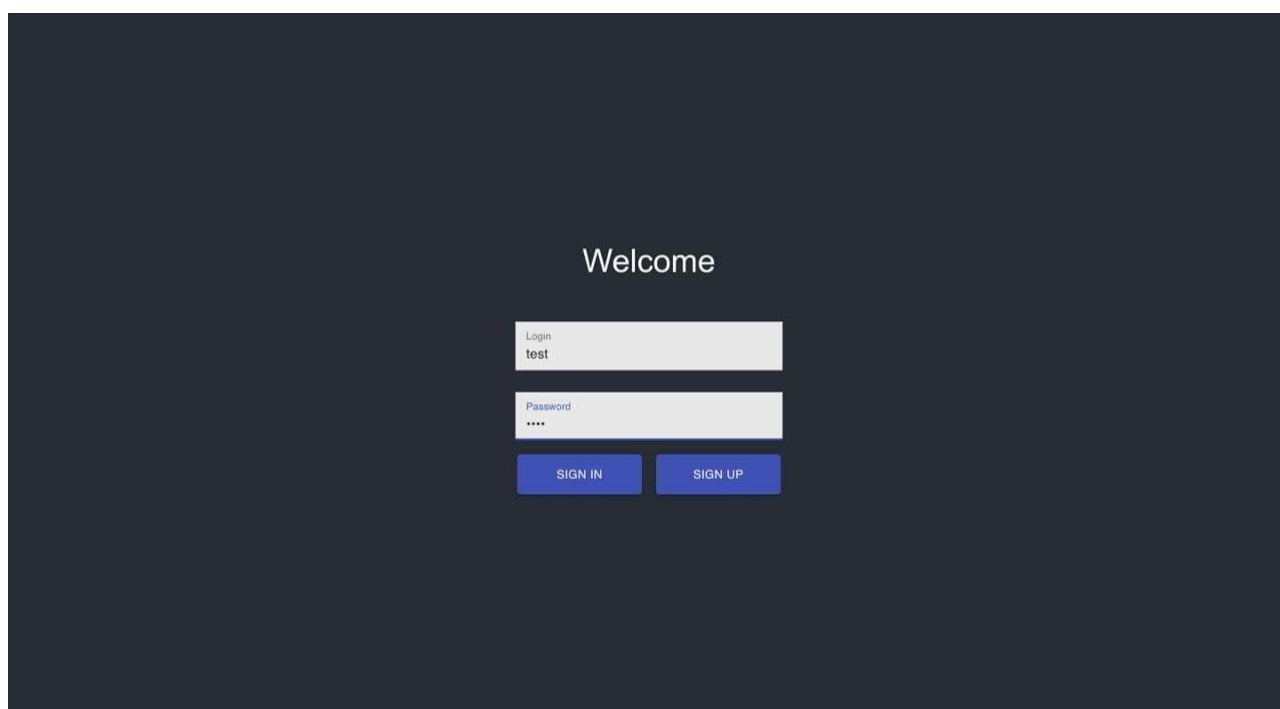


Рисунок Б.1 – Вікно авторизації ПП

Для вдалої авторизації необхідно коректно заповнити поля «Логін» та «Пароль». Після цього програма відправляє запит до БД для пошуку користувача з цим логіном, а також порівнює пароль до паролю цього користувача. У випадку, якщо даний профіль існує і пароль вірний, користувач переходить до наступного вікна. Якщо пароль не вірний, то користувач отримує повідомлення про помилку. Якщо користувач з таким логіном не існує, то необхідно зареєструвати такого користувача. Щоб перейти до вікна реєстрації необхідно натиснути на кнопку «Sign up».

Після авторизації, користувач переходить до списку проєктів, сторінка проєктів представлена на рис Б.2.

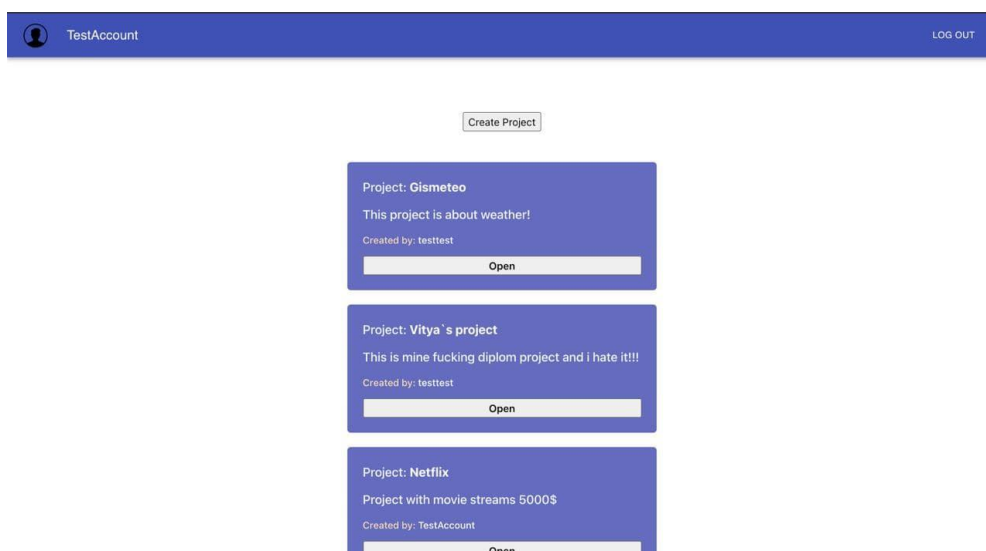


Рисунок Б.2 – Вигляд головної сторінки ПП «Проекти»

Якщо користувач помилково увійшов до некоректного аканту, або якщо користувач завершив роботу, існує кнопка «Log out».

На сторінки проєктів користувач може створити новий проєкт або переглянути існуючі проєкти. Для перегляду існуючого проєкту необхідно натиснути на кнопку «Open», а для створення нового – «Create project».

Б.7 Вхідні та вихідні дані

В якості вхідних даних ПП є вміст текстових полів, що заповнює користувач, а в якості вихідних даних ПП є результати запитів до БД. Серед типів даних що можуть зчитуватися та записуватися в БД: дата, текст, число.

ДОДАТОК В
Текст програми

В.1 Файл Login.js

```
import React from 'react';
import PropTypes from 'prop-types';
import Button from '@material-ui/core/Button';
import TextField from '@material-ui/core/TextField';
import useStyles from './styles';

const LoginCont = (props) => {
  const classes = useStyles();

  const {
    error,
    login,
    onChangeLogin,
    password,
    onChangePassword,
    signIn,
    goToRegistration,
  } = props;
  return (
    <div className="App-header">
      <form className={classes.container} noValidate autoComplete="off">
        <h1>Welcome</h1>
        <span className={classes.error}>{error || ""}</span>
        <div>
          <TextField
            value={login}
            id="filled-basic"
            className={classes.textField}
            label="Login"
            margin="normal"
            variant="filled"
            onKeyPress={event => event.key === 'Enter' ? signIn() : null}
            onChange={onChangeLogin}
          />
        </div>
        <div>
          <TextField
            value={password}
            id="filled-basic"
```

```

        type="password"
        className={classes.textField}
        label="Password"
        margin="normal"
        variant="filled"
        onChange={onChangePassword}
        onPress={event => event.key === 'Enter' ? signIn() : null}
      />
    </div>
    <div className="buttonsCont">
      <Button
        variant="contained"
        color="primary"
        className={classes.button}
        onClick={signIn}
      >
        Sign In
      </Button>
      <Button
        variant="contained"
        color="primary"
        className={classes.button}
        onClick={goToRegistration}
      >
        Sign Up
      </Button>
    </div>
  </form>
</div>
)
};

```

```

LoginCont.propTypes = {
  error: PropTypes.string,
  login: PropTypes.string,
  onChangeLogin: PropTypes.func,
  password: PropTypes.string,
  onChangePassword: PropTypes.func,
  signIn: PropTypes.func,
  goToRegistration: PropTypes.func,
}

```

```
export default LoginCont;
```

В.2 Файл index.js директорії client/src/components/Registration

```
import React, { useEffect, useReducer } from 'react';
import PropTypes from 'prop-types';
import { connect } from 'react-redux';
import { withRouter } from 'react-router';
import Button from '@material-ui/core/Button';
import CustomTextField from '../customTextField';
import { singUpAction } from '../../redux/actions/auth';
import { pushError } from '../../redux/actions/error';
import useStyles from './styles';
import './registration.css';

const initialState = {
  userName: "",
  firstName: "",
  lastName: "",
  userInfo: "",
  login: "",
  password: "",
  confirmPassword: "",
};

function reducer(state, action) {
  switch (action.type) {
    case 'changeUserName':
      return { ...state, userName: action.payload };
    case 'changeFirstName':
      return { ...state, firstName: action.payload };
    case 'changeLastName':
      return { ...state, lastName: action.payload };
    case 'changeUserInfo':
      return { ...state, userInfo: action.payload };
    case 'changeLogin':
      return { ...state, login: action.payload };
    case 'changePassword':
      return { ...state, password: action.payload };
    case 'changeConfirmPassword':
      return { ...state, confirmPassword: action.payload };
  }
}
```

```

    default:
      return state
    }
  }

const Registration = (props) => {
  const [state, dispatch] = useReducer(reducer, initialState);
  const {
    userName,
    firstName,
    lastName,
    userInfo,
    login,
    password,
    confirmPassword,
  } = state;
  const classes = useStyles();

  useEffect(() => {
    props.onPushError("");
    const token = JSON.parse(localStorage.getItem('token'));
    if (token !== null) {
      props.history.push('/home');
    }
  })

  const onChangeUserName = event => dispatch({ type: 'changeUserName', payload: event.target.value});
  const onChangeFirstName = event => dispatch({ type: 'changeFirstName', payload: event.target.value});
  const onChangeLastName = event => dispatch({ type: 'changeLastName', payload: event.target.value});
  const onChangeUserInfo = event => dispatch({ type: 'changeUserInfo', payload: event.target.value});
  const onChangeLogin = event => dispatch({ type: 'changeLogin', payload: event.target.value});
  const onChangePassword = event => dispatch({ type: 'changePassword', payload: event.target.value});
  const onChangeConfirmPassword = event => dispatch({ type: 'changeConfirmPassword', payload:
event.target.value});

  const singUp = () => props.onSingUp({ firstName, lastName, userName, login, password, userInfo,
goHome });
  const goBack = () => props.history.push('/');
  const goHome = () => props.history.push('/home');

  const isConfirmedPassword = password === confirmPassword && password.length >= 8 &&
password.length >= 8;

```

```

const isFilledAll = userName.trim() !== " " && firstName.trim() !== " " && lastName.trim() !== " " &&
login.trim() !== " " && password.trim() !== " " && confirmPassword.trim() !== " " ? true : false;
return (
  <div className="App">
    <header className="App-header">
      <form className={classes.container} noValidate autoComplete="off">
        <h2>Registration</h2>
        <span className={classes.error}>{props.error}</span>
        <CustomTextField
          id="filled-basic"
          value={login}
          onChange={onChangeLogin}
          label='Login'
        />
        <CustomTextField
          id="filled-basic"
          value={password}
          onChange={onChangePassword}
          label='Password'
          type="password"
        />
        <span className={password.length >= 8 ? classes.passwordInfoConf : classes.passwordInfoError}>
          Must be at least 8 characters
        </span>
        <CustomTextField
          id="filled-basic"
          value={confirmPassword}
          onChange={onChangeConfirmPassword}
          label='Confirm password'
          type="password"
        />
        <span className={isConfirmedPassword ? classes.passwordInfoConf : classes.passwordInfoError}>
          {isConfirmedPassword ? 'Confirmed' : 'Confirm the password'}
        </span>
        <CustomTextField
          id="filled-basic"
          value={userName}
          onChange={onChangeUserName}
          label='User Name'
        />
        <CustomTextField
          id="filled-basic"

```

```

        value={firstName}
        onChange={onChangeFirstName}
        label='First Name'
      />
      <CustomTextField
        id="filled-basic"
        value={lastName}
        onChange={onChangeLastName}
        label='Last Name'
      />
      <CustomTextField
        id="outlined-multiline-static"
        value={userInfo}
        onChange={onChangeUserInfo}
        label='About me (Can be empty)'
        multiline
        rows="4"
      />
      <div className="buttonCont">
        <Button
          variant="contained"
          color="primary"
          className={classes.button}
          onClick={goBack}
        >
          Go Back
        </Button>
        <Button
          disabled={!isFilledAll}
          variant="contained"
          color="primary"
          className={classes.button}
          onClick={singUp}
        >
          Sign Up
        </Button>
      </div>
    </form>
  </header>
</div>
);
};

```

```

Registration.propTypes = {
  error: PropTypes.string,
  onSingUp: PropTypes.func,
  onPushError: PropTypes.func,
}

const mapStateToProps = function(state) {
  return {
    error: state.error.error,
  }
};

const mapDispatchToProps = (dispatch) => ({
  onSingUp: (payload) => dispatch(singUpAction(payload)),
  onPushError: (payload) => dispatch(pushError(payload))
})

export default connect(mapStateToProps, mapDispatchToProps)(withRouter(Registration));

```

В.3 Файл index.js директорії client/src/components/Project

```

import React, { useState, useEffect } from 'react';
import { connect } from 'react-redux';
import { withRouter } from 'react-router';
import { createConfirmation } from 'react-confirm';

import { createTaskAction, getTasksAction, deleteTaskAction, setTasksAction, setProjectsAction,
deleteProjectAction } from '../redux/actions/tasks';
import { getMyProfileAction } from '../redux/actions/users';
import axios from '../utils/axios';
import moment from 'moment';

import TopPanel from '../TopPanel';
import CreateTaskModal from '../CreateTaskModal';
import './Projects.css';

const Project = (props) => {
  const {
    user,
    tasks,

```

```

current,
history,
getTasks,
setTasks,
getMyProfile,
createTask,
deleteTaskAction,
deleteProjectAction,
setProjectsAction,
projects,
} = props;

```

```

const [isOpen, _isOpen] = useState(false);
const [title, _title] = useState("");
const [description, _description] = useState("");
const [priority, _priority] = useState('Low');
const [assigned, _assigned] = useState(null);
const [usersList, _usersList] = useState([]);
const [start, _start] = useState(new Date());
const [end, _end] = useState(new Date());

```

```

useEffect(() => {
  getMyProfile();
  if (!current || !current?._id) {
    setTimeout(() => history.goBack());
  }
}, []);

```

```

useEffect(() => {
  getUser();
  current && current?._id && getTasks({ project: current._id });
}, []);

```

```

const getUser = async () => {
  const config = (token) => {
    return {
      headers: {
        'auth-token': token
      }
    }
  };
};

```

```

const token = JSON.parse(localStorage.getItem('token'));
const { data: users } = await axios.get('/tasks/users', config(token));
const names = users?.map(_ => `${_.firstName} ${_.lastName}`);
_usersList(names);
};

const deleteTask = (_id) => {
  deleteTaskAction({ _id });
  setTasks(tasks.filter(_ => _.id !== _id));
};

const deleteProjectFunction = () => {
  deleteProjectAction({ _id: current._id });
  setProjectsAction(projects?.filter(_ => _.id !== current._id));

  history.goBack();
};

const renderTasks = () => !tasks || tasks.length === 0
? <text className="empty">No tasks was found :(</text>
: <div className='projectsContainerColumn'>
  { tasks?.map((_) => (
    <div className='taskContainer'>
      <div className='taskTop'>
        <text className='taskTitle'>{ _.title }</text>
        <button className='deleteTaskButton' onClick={() => deleteTask(_.id)}>
          <text className='deleteX'>x</text>
        </button>
      </div>
      <text className='taskDescription'>{ _.description }</text>
      <text className='taskDescription'>{ _.priority } Priority</text>
      <text className='taskDescription'>Assigned to: { _.assignedTo }</text>
      <text className='taskDescription' style={{ fontSize: 12 }}>Started at: {
moment(_.startedAt).format('DD-MM-YYYY HH:MM')}</text>
      <text className='taskDescription' style={{ fontSize: 12 }}>Ended at: {
moment(_.endedAt).format('DD-MM-YYYY HH:MM')}</text>
    </div>
  )))}
</div>

const handleTitle = (e) => _title(e.target.value);

```

```
const handleDescription = (e) => _description(e.target.value);
```

```
const openModal = () => {
  _isOpened(true);
  _title("");
  _description("");
  _priority('Low');
  _assigned(null);
  _start(new Date());
  _end(new Date());
};
```

```
const close = () => {
  _isOpened(false);
};
```

```
const create = () => {
  if (
    !title
    || title.trim() === ""
    || !description
    || description.trim() === ""
    || !priority
    || !assigned
    || !start
    || !end
  ) {
    alert('Fill all fields, please!');
  } else {
    createTask({
      title,
      description,
      priority,
      assignedTo: assigned,
      project: current._id,
      startedAt: start,
      endedAt: end,
    });
    _isOpened(false);
  }
};
```

```

if (!user) {
  return null;
}

return (
  <div>
    <TopPanel user={user}/>
    <CreateTaskModal
      isOpened={isOpened}
      close={close}
      handleTitle={handleTitle}
      title={title}
      handleDescription={handleDescription}
      description={description}
      create={create}
      priority={priority}
      _priority={_priority}
      assigned={assigned}
      _assigned={_assigned}
      usersList={usersList}
      start={start}
      _start={_start}
      end={end}
      _end={_end}
    />
    <div className='card'>
      <div className='topCard'><text className='title'>{ current.title }</text><text>created by: {
current.createdBy }</text></div>
      <text className='description'>{ current.description }</text>
      <button className='delete' onClick={deleteProjectFunction}>
        <text className='deleteLabel'>Delete project</text>
      </button>

      <div className='tasks'>
        <text className='tasksLabel'>Tasks</text>
        <button className='createTask' onClick={openModal}>
          <text className='createTaskLabel'>Create task</text>
        </button>

        { renderTasks() }
      </div>

```

```

    </div>
  </div>
);
};

const mapStateToProps = (state) => ({
  user: state.user.user,
  tasks: state.tasks.tasks,
  current: state.current.current,
  projects: state.projects.projects,
});

const mapDispatchToProps = (dispatch) => ({
  setTasks: (payload) => dispatch(setTasksAction(payload)),
  getTasks: (payload) => dispatch(getTasksAction(payload)),
  getMyProfile: () => dispatch(getMyProfileAction()),
  createTask: (payload) => dispatch(createTaskAction(payload)),
  setProjectsAction: (payload) => dispatch(setProjectsAction(payload)),
  deleteTaskAction: (payload) => dispatch(deleteTaskAction(payload)),
  deleteProjectAction: (payload) => dispatch(deleteProjectAction(payload)),
});

export default connect(mapStateToProps, mapDispatchToProps)(withRouter(Project));

```

B.4 Файл CreateTaskModal.js

```

import React from 'react';
import Modal from '@material-ui/core/Modal';
import Dropdown from 'react-dropdown';
import DateTimePicker from 'react-datetime-picker'
import 'react-dropdown/style.css';
import './Projects.css';

const CreateTaskModal = ({ isOpened, create, close, title, description, handleTitle, handleDescription,
priority, _priority, assigned, _assigned, usersList, start, _start, end, _end }) => {
  const options = ['Low', 'Medium', 'High'];

  return (
    <div>
      <Modal
        aria-labelledby="simple-modal-title"

```

```

aria-describedby="simple-modal-description"
open={isOpen}
onClose={close}
>
  <div style={{ top: `50%`, left: `50%`, transform: `translate(-50%, -50%)` }}
className='modalContainer'>
    <text className='modalTitle'>Create a task</text>

    <div className='modalInputContainer'>
      <text className='modalInputLabel'>Task title</text>
      <input value={title} onChange={handleTitle}/>
    </div>

    <div className='modalInputContainer'>
      <text className='modalInputLabel'>Task description</text>
      <textarea className='modalTextArea' value={description} onChange={handleDescription}
cols="40" rows="3"></textarea>
    </div>

    <div className='modalInputContainer'>
      <text className='modalInputLabel'>Priority</text>
      <Dropdown options={options} onChange={({ value }) => _priority(value)} value={priority}
placeholder="Select an option" />
    </div>

    <div className='modalInputContainer'>
      <text className='modalInputLabel'>Assign to</text>
      <Dropdown options={usersList} onChange={({ value }) => _assigned(value)} value={assigned}
placeholder="Select an option" />
    </div>

    <div className='modalInputContainer'>
      <text className='modalInputLabel'>Start</text>
      <DateTimePicker value={start} onChange={(value) => _start(value)} />
    </div>

    <div className='modalInputContainer'>
      <text className='modalInputLabel'>End</text>
      <DateTimePicker value={end} onChange={(value) => _end(value)} />
    </div>

    <button className='modalCreate' onClick={create}>

```

```

        <text>Create</text>
      </button>
    </div>
  </Modal>
</div>
);
};

export default CreateTaskModal;

```

В.5 Файл `index.js` директорії `client/src/components/Tasks`

```

import React, { useState, useEffect } from 'react';
import { connect } from 'react-redux';
import { withRouter } from 'react-router';

import AddProjectModal from '../AddProjectModal';
import TopPanel from '../TopPanel';
import './Tasks.css';

import { getTasksAction, getProjectsAction, createProjectAction, setCurrentProjectAction } from
'../../redux/actions/tasks';

import { getMyProfileAction } from '../../redux/actions/users';

const TasksPage = (props) => {
  const {
    user,
    projects,

    getProfile,
    getProjects,
    createProject,
    setCurrentProject,

    history,
  } = props;

  const [isOpen, _isOpen] = useState(false);
  const [title, _title] = useState("");
  const [description, _description] = useState("");

```

```

useEffect(() => {
  getProfile();
  getProjects();
}, []);

useEffect(() => {
  if (!isOpen) {
    setDefault();
  }
}, [isOpen]);

const renderProjects = () => !projects || projects.length === 0
  ? <text className="empty">No projects was found :(</text>
  : <div className='projectsContainerColumn'>
    { projects?.map((item) => (
      <div className='project'>
        <text className='projectTitle'>Project: <text style={{ color: 'white', fontWeight: '700' }}>{ item.title
      }</text></text>
        <text className='projectTitle'>{ item.description }</text>
        <text className='createdBy'>Created by: <text style={{ color: 'white' }}>{ item.createdBy
      }</text></text>
        <button onClick={() => open(item)}>
          <text style={{ fontWeight: '600' }}>Open</text>
        </button>
      </div>
    ))}
  </div>

const addButton = () => (
  <button onClick={() => _isOpen(true)}>
    <text>Create Project</text>
  </button>
);

const open = (project) => {
  setCurrentProject(project);
  history.push('/project');
}

const handleTitle = (e) => _title(e.target.value);

const handleDescription = (e) => _description(e.target.value);

```

```

const close = () => _isOpen(false);

const setDefault = () => {
  _title("");
  _description("");
};

const create = () => {
  createProject({ title, description });
  _isOpen(false);
};

if (!user) {
  return null;
};

return (
  <div style={{ alignItems: 'center', justifyContent: 'center', flex: '1' }}>
    <TopPanel user={user} />
    <AddProjectModal
      title={title}
      description={description}
      isOpen={isOpen}
      handleTitle={handleTitle}
      handleDescription={handleDescription}
      close={close}
      create={create}
    />
    <div className="container">
      { addButton() }
      { renderProjects() }
    </div>
  </div>
);
}

const mapStateToProps = (state) => ({
  user: state.user.user,
  tasks: state.tasks.tasks,
  projects: state.projects.projects,

```

```

});

const mapDispatchToProps = (dispatch) => ({
  getTasks: () => dispatch(getTasksAction()),
  getProjects: () => dispatch(getProjectsAction()),
  getProfile: () => dispatch(getMyProfileAction()),
  createProject: (payload) => dispatch(createProjectAction(payload)),
  setCurrentProject: (payload) => dispatch(setCurrentProjectAction(payload)),
});

export default connect(mapStateToProps, mapDispatchToProps)(withRouter(TasksPage));

```

В.6 Файл AddTasksModal.js директорії client/src/components/Tasks

```

import React from 'react';
import Modal from '@material-ui/core/Modal';

import './Tasks.css';

const AddProjectModal = ({ isOpen, close, title, description, handleTitle, handleDescription, create }) => (
  <div>
    <Modal
      aria-labelledby="simple-modal-title"
      aria-describedby="simple-modal-description"
      open={isOpen}
      onClose={close}
    >
      <div style={{ top: `50%`, left: `50%`, transform: `translate(-50%, -50%)` }}
        className='modalContainer'>
          <text className='modalTitle'>Create a new project</text>
          <div className='modalInputContainer'>
            <text className='modalInputLabel'>Project`s title</text>
            <input value={title} onChange={handleTitle}/>
          </div>

          <div className='modalInputContainer'>
            <text className='modalInputLabel'>Project`s description</text>
            <textarea className='modalTextArea' value={description} onChange={handleDescription} cols="40"
              rows="3"></textarea>
          </div>

          <button className='modalCreate' onClick={create}>

```

```

        <text>Create</text>
      </button>
    </div>
  </Modal>
</div>
);

export default AddProjectModal;

```

В.7 Файл `index.js` директорії `client/src/components/Home`

```

import React, { useEffect, useState } from 'react';
import PropTypes from 'prop-types';
import { connect } from 'react-redux';
import { withRouter } from 'react-router';
import {
  getProfileInfoAction,
  getUsersListAction,
  saveUserData,
  deleteUserAction,
  removeUserTypeAction,
  addTypeToUserAction,
  getMyProfileAction,
} from '../redux/actions/users';
import { getTypesAction } from '../redux/actions/types';
import TopPanel from '../TopPanel';
import Filter from '../Filter';
import Pagination from '../Pagination';
import UserDetails from '../UserDetails';
import './Home.css';
import Button from '@material-ui/core/Button';

const Home = (props) => {
  const [open, setOpen] = useState(false);
  const [pages, changePages] = useState(0);
  const [type, setType] = useState('none');
  const [selectedUser, selectUser] = useState(null);
  const [pageNumber, changePageNumber] = useState(1);
  const [usersPerPage, changeUsersPerPage] = useState(5);
  const [userDetailsOpen, changeUserDetailsOpen] = useState(false);

```

```

const token = JSON.parse(localStorage.getItem('token'));

const {
  onGetMyProfile,
  history,
  onGetTypesAction,
  onGetUsersListAction,
} = props;

useEffect(() => {
  if (props.count % usersPerPage === 0) {
    let countPages = props.count / usersPerPage;
    changePages(countPages)
  } else {
    let countPages = props.count / usersPerPage;
    changePages(~countPages + 1)
  }
}, [props.count, usersPerPage])

useEffect(() => {

  props.usersList.map((item) => {
    if (selectedUser && item._id === selectedUser._id) {
      selectUser(item);
    }
  })
}, [props.usersList])

useEffect(() => {
  token !== null ? onGetMyProfile() : history.push('/');
  onGetTypesAction();
  onGetUsersListAction({ number: pageNumber, usersPerPage: 5 });
}, [token, pageNumber, onGetMyProfile, onGetTypesAction, onGetUsersListAction, history]);

const renderUsers = () => {
  return (
    props.usersList.map((item) => {
      return (
        <li key={item._id} className="w3-bar cardCont">
          <Button
            onClick={() => handleArrow(item)}
            className="w3-bar-item w3-button w3-white w3-xlarge w3-right arrow"

```

```

        variant="contained"
      >
        →
      </Button>
      

      <div className="w3-bar-item">
        <span className="w3-large">{item.userName}</span><br />
        <span>{item.firstName} {item.lastName}</span>
      </div>
    </li>
  )
})
)
}

const handleArrow = (user) => {
  selectUser(user);
  handleOpen();
}

const handleChangeUsersPerPage = event => {
  let selectedValue = event.target.value;
  props.count % selectedValue === 0
  ? changePages(props.count / selectedValue)
  : changePages(~ ~(props.count / selectedValue) + 1)

  if (pageNumber > pages) {
    changePageNumber(pages);
  }
  changeUsersPerPage(selectedValue)
  if (type !== 'none') {
    props.onGetUsersListAction({ number: pageNumber, filter: type, usersPerPage: selectedValue });
  } else {
    props.onGetUsersListAction({ number: pageNumber, usersPerPage: selectedValue });
  }
}

const handleChangePage = (newPage) => {
  if (type !== 'none') {
    props.onGetUsersListAction({ number: newPage, filter: type, usersPerPage });
    changePageNumber(newPage);
  }
}

```

```

    } else {
      props.onGetUsersListAction({ number: newPage, usersPerPage });
      changePageNumber(newPage);
    }
  };

const handleChangeType = event => {
  setType(event.target.value);
  if (event.target.value !== 'none') {
    changePageNumber(1);
    props.onGetUsersListAction({ number: 1, filter: event.target.value, usersPerPage });
  } else {
    props.onGetUsersListAction({ number: 1, usersPerPage });
  }
};

const goToLogin = () => props.history.push('/');

const deleteUser = (id) => {
  props.onDeleteUser({ userId: id, goToLogin });
  handleClose();
}

const handleOpen = () => {
  setOpen(true);
};

const handleClose = () => {
  setOpen(false);
};

const addType = (userId, userName, firstName, lastName, userInfo, userTypes, typeTitle) => {
  if (selectedUser.userTypes.indexOf(typeTitle) < 0 ) {
    props.onAddTypeToUser({ userId, userName, firstName, lastName, userInfo, userTypes: [...userTypes,
typeTitle], typeTitle });

    const newTypes = selectedUser.userTypes;
    newTypes.push(typeTitle);
    const newUser = {
      _id: selectedUser._id,
      userName: selectedUser.userName,
      firstName: selectedUser.firstName,
      lastName: selectedUser.lastName,

```

```

        userInfo: selectedUser.userInfo || "",
        userTypes: newTypes,
    };
    selectUser(newUser);
} else {
    alert('Already in use!');
}
}

const removeType = (id, userName, firstName, lastName, userInfo, userTypes, typeTitle) => {
    if (type === 'none') {
        props.onRemoveUserType({ userId: id, userName, firstName, lastName, userInfo, userTypes:
userTypes.filter(it => it !== typeTitle), typeTitle: typeTitle, pageNumber: pageNumber });
    } else {
        props.onRemoveUserType({ userId: id, userName, firstName, lastName, userInfo, userTypes:
userTypes.filter(it => it !== typeTitle), typeTitle: typeTitle, pageNumber: pageNumber, type });
    }
}

const newUser = {
    _id: selectedUser._id,
    userName: selectedUser.userName,
    firstName: selectedUser.firstName,
    lastName: selectedUser.lastName,
    userInfo: selectedUser.userInfo || "",
    userTypes: selectedUser.userTypes.filter((item) => item !== typeTitle)
};
selectUser(newUser);
}

if (props.user === null ) {
    return null;
}
return (
    <div>
        <TopPanel user={props.user} />
        <div className="homeCont">
            <Filter
                type={type}
                types={props.types}
                handleChangeType={handleChangeType}
            />
            <div className="w3-container" style={{ minWidth: 350 }}>
                <ul className="w3-ul w3-card-4 userCont">

```

```

        { renderUsers() }
      </ul>
    </div>
    <Pagination
      pages={pages}
      pageNumber={pageNumber}
      count={props.count || 0}
      usersPerPage={usersPerPage}
      handleChangePage={handleChangePage}
      handleChangeUsersPerPage={handleChangeUsersPerPage}
    />
    <UserDetails
      open={open}
      addType={addType}
      user={selectedUser}
      types={props.types}
      usersList={props.usersList}
      deleteUser={deleteUser}
      removeType={removeType}
      handleClose={handleClose}
      userDetailsOpen={userDetailsOpen}
      changeUserDetailsOpen={changeUserDetailsOpen}
    />
  </div>
</div>
);
};

```

```

Home.propTypes = {
  type: PropTypes.array,
  user: PropTypes.object,
  count: PropTypes.number,
  usersList: PropTypes.array,
  onDeleteUser: PropTypes.func,
  onGetMyProfile: PropTypes.func,
  onAddTypeToUser: PropTypes.func,
  onRemoveUserType: PropTypes.func,
  onGetTypesAction: PropTypes.func,
  onGetUsersListAction: PropTypes.func,
}

```

```
const mapStateToProps = function(state) {  
  return {  
    types: state.types.types,  
    user: state.user.user,  
    usersList: state.usersList.usersList,  
    count: state.count.count,  
  }  
};
```

```
const mapDispatchToProps = (dispatch) => ({  
  onGetMyProfile: () => dispatch(getMyProfileAction()),  
  onAddTypeToUser: (payload) => dispatch(addTypeToUserAction(payload)),  
  onRemoveUserType: (payload) => dispatch(removeUserTypeAction(payload)),  
  onDeleteUser: (payload) => dispatch(deleteUserAction(payload)),  
  onGetTypesAction: () => dispatch(getTypesAction()),  
  onGetProfileInfoAction: (payload) => dispatch(getProfileInfoAction(payload)),  
  onSaveUserData: (payload) => dispatch(saveUserData(payload)),  
  onGetUsersListAction: (payload) => dispatch(getUsersListAction(payload)),  
});
```

```
export default connect(mapStateToProps, mapDispatchToProps)(withRouter(Home));
```

ДОДАТОК Д
Слайди презентації

Національний університет "Запорізька політехніка"

ПРОГРАМНА РЕАЛІЗАЦІЯ СИСТЕМИ УПРАВЛІННЯ ПРОЄКТАМИ

Виконав:
студент групи КНТ-118сп

В.О. Філатов

Керівник:
к.ф.-м.н., доцент

Е.В. Терещенко

1

Рисунок Д.1 – Слайд № 1

Мета, об'єкт, та предмет дослідження

Об'єкт дослідження – система управління проектами.

Предмет дослідження – створення проектів та відповідних ним завдань.

Мета роботи – створення системи управління проектами для конфіденційного контролю виконання проектів та делегації задач.

Актуальність та проблематика роботи полягає в тому, що існуючі системи управління проектами не надають можливості працювати без підключення до серверів компаній-розробників систем, що створює ризик втрати конфіденційності даних, або втрати доступу до системи у випадку втрати доступу до мережі Інтернет.

2

Рисунок Д.2 – Слайд № 2

Постановка завдання для роботи

Для виконання поставленої мети потрібно виконати наступні завдання:

- створити метод для створення проєктів;
- створити метод для реєстрації та авторизації користувачів;
- створити метод для створення завдань;
- створити можливість для делегації завдань;
- створити можливість для встановлення дат початку та кінця завдань.

3

Рисунок Д.3 – Слайд № 3

Порівняльна характеристика методів управління проєктами

Можливості	Класичний проектний менеджмент	Метод Agile	Метод Scrum	Метод Lean	Метод Six Sigma
Простота реалізації	–	+	+	–	–
Звичність для користувача	+	+	+	–	–
Зрозумілість	+	+	+	+	–
Гнучкість	–	+	+	–	–
Популярність	+	+	+	–	–
Можливість використання підприємства-ми будь-якого розміру	+	+	–	–	–
Стабільність	+	–	+	+	+

4

Рисунок Д.4 – Слайд № 4

Порівняльна характеристика мов програмування

Критерії порівняння	JavaScript	PHP	Python	Java	C#
Кросплатформеність	+	–	+	+	+
Швидкодія	+	–	–	+	+
Простота вивчення	+	+	+	–	–
Простота розробки	+	+	+	–	–
Кросмовна взаємодія	+	+	+	+	+
Можливість використання додаткових бібліотек	+	+	+	+	+

5

Рисунок Д.5 – Слайд №5

Порівняльна характеристика середовищ розробки

Критерії порівняння	Microsoft Visual Studio	IntelliJ IDEA:	Eclipse
Кросплатформенність	+	+	+
Низький поріг входження	+	–	+
Велика кількість безкоштовних функцій	+	–	+
Простота в використанні	+	–	+
Можливість безкоштовного використання	+	+	+
Надійність	+	–	–
Швидкодія	+	–	+
Економія ресурсів ПК	+	–	+

6

Рисунок Д.6 – Слайд № 6

Сутності бази даних

Сутність	Атрибут	Тип	Розмір	Обов'язкове	Опис сутності
Акаунт	Код акаунта	Лічильник	Довге ціле	Так	Містить інформацію про акаунти
	Логін	Текстовий	255	Так	
	Пароль	Текстовий	255	Так	
	Ім'я	Текстовий	255	Так	
Проект	Код_проекта	Числовий	Довге ціле	Так	Містить інформацію про проекти
	Назва_проекта	Текстовий	255	Так	
	Опис_проекта	Текстовий	255	Ні	
	Власник_проекта	Текстовий	255	Так	
Завдання	Код_завдання	Лічильник	Довге ціле	Так	Містить інформацію про завдання
	Відповідний_проект	Текстовий	255	Так	
	Дата_початку	Дата	Довге ціле	Так	
	Дата_кінця	Дата	Довге ціле	Так	
	Відповідний_виконавець	Текстовий	255	Так	

7

Рисунок Д.7 – Слайд № 7

Інтерфейс: вікно логіну

Welcome

Login

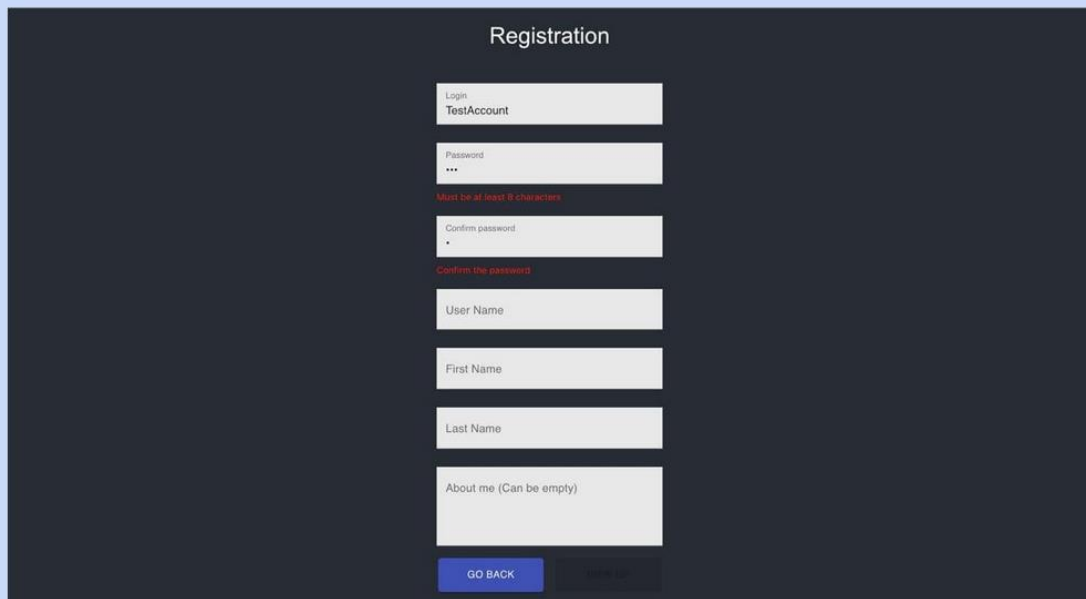
Password

SIGN IN SIGN UP

8

Рисунок Д.8 – Слайд № 8

Інтерфейс: вікно реєстрації



Registration

Login
TestAccount

Password

Must be at least 8 characters

Confirm password
*

Confirm the password

User Name

First Name

Last Name

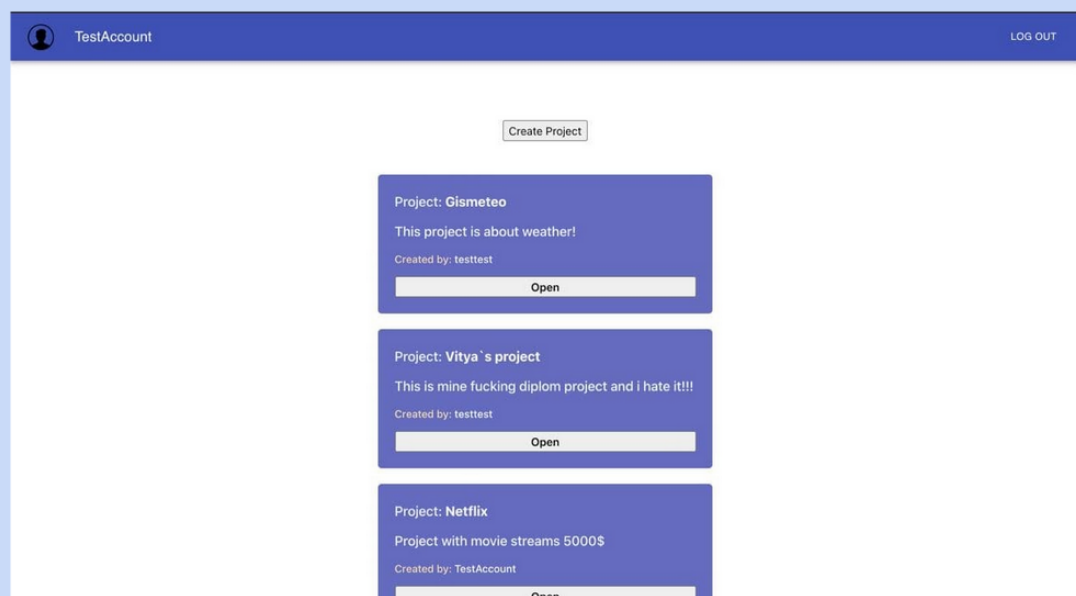
About me (Can be empty)

GO BACK

9

Рисунок Д.9 – Слайд № 9

Інтерфейс: вікно проєктів



TestAccount

LOG OUT

Create Project

Project: Gismeteo
This project is about weather!
Created by: testtest
Open

Project: Vitya`s project
This is mine fucking diplom project and i hate it!!!
Created by: testtest
Open

Project: Netflix
Project with movie streams 5000\$
Created by: TestAccount
Open

10

Рисунок Д.10 – Слайд № 10

Інтерфейс: вікно проєкта

TestAccount LOG OUT

Netflix created by: TestAccount

Project with movie streams 5000\$

Delete project

Tasks

Create task

No tasks was found :(

11

Рисунок Д.11 – Слайд № 11

Інтерфейс: вікно профіля

My Profile

defaultUser

ADD TYPE

Login
TestAccount

User Name
TestAccount

First Name
FirstName

Last Name
LastName

About me (Can be empty)
Information about me

CANCEL SAVE

12

Рисунок Д.12 – Слайд № 12

Інтерфейс: вікно створення проєкта

The screenshot shows a web application interface with a dark blue header. On the left, there is a user profile icon and the text 'TestAccount'. On the right, there is a 'LOG OUT' link. In the center, a white modal dialog box titled 'Create a new project' is displayed. Inside the dialog, there are two text input fields: 'Project's title' and 'Project's description'. Below these fields is a 'Create' button. The background of the application is a solid grey color.

13

Рисунок Д.13 – Слайд № 13

Інтерфейс: вікно створення завдання

The screenshot shows a web application interface with a dark blue header. On the left, there is a user profile icon and the text 'vitya'. On the right, there is a 'LOG OUT' link. In the center, a white modal dialog box titled 'Create a task' is displayed. Inside the dialog, there are several form fields: 'Task title' (text input), 'Task description' (text area), 'Priority' (dropdown menu with 'Low' selected), 'Assign to' (dropdown menu with 'Vitya OuityaTyuTyua' selected), 'Start' (datetime picker with '24.5.2021 21:00'), and 'End' (datetime picker with '24.5.2021 21:00'). Each datetime picker has a close button (X) and a calendar icon. Below these fields is a 'Create' button. The background of the application is a solid grey color. To the left of the dialog, a sidebar is partially visible with the text 'Gismeteo', 'This project is about', and a 'Delete project' button. To the right, a sidebar is partially visible with the text 'created by: testtest'.

14

Рисунок Д.14 – Слайд № 14

Висновок

При створенні диплома було виконано необхідні завдання для вирішення поставленої проблеми. Для цього було досліджено предметну область та розглянуто та визначено основні поняття.

Для реалізації ППІ було проаналізовано методи систем управління проєктами, проведено оцінку їх переваг та недоліків. Було обрано найкращий метод для створення універсальної системи управління проєктами. Було сформульовано завдання до роботи.

В результаті було програмно реалізовано систему управління проєктами. Створений ППІ дозволяє проводити управління проєктами та завданнями, делегувати завдання та встановлювати дати початку та кінця, зберігати дані в безпеці та ін.

Рисунок Д.15 – Слайд № 15