Державний торговельно-економічний університет

Кафедра комп'ютерних наук та інформаційних систем

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА

на тему:

«Розробка Web-додатку для управління проєктами»

Студента 2 курсу, 4м групи
спеціальності
122 «Комп'ютерні науки» _____ *підпис студента* Стус Олег Андрійович

Науковий керівник
кандидат педагогічних наук, доцент _____ *підпис керівника* Базурін Віталій Миколайович

Гарант освітньої програми
доктор фізико-математичних наук, професор _____ *підпис керівника* Пурський Олег Іванович

Київ 2023

Державний торговельно-економічний університет

Факультет інформаційних технологій

Кафедра комп'ютерних наук та інформаційних систем

Спеціальність 122 «Комп'ютерні науки»

Освітня програма «Комп'ютерні науки»

Затверджую

Зав. кафедри _____Пурський О.І.

«9» грудня 2022р.

Завдання

на випускну кваліфікаційну роботу студенту

Стусу Олегу Андрійовичу

(прізвище, ім'я, по батькові)

1. Тема випускної кваліфікаційної роботи

«Розробка Web-додатку для управління проєктами»

Затверджена наказом ректора від *«06» грудня 2022 р. № 3284*

2. Строк здачі студентом закінченої роботи *24 листопада 2023 року*

3. Цільова установка та вихідні дані до роботи

Мета роботи: Метою є розробка Web-додатку для управління проєктами, який надасть зручний та ефективний інструмент для планування, керування та моніторингу проєктів.

Об'єкт дослідження: Об'єктом дослідження є web-додаток для управління проєктами.

Предмет дослідження технології, методи та процеси, пов'язані з розробкою web-додатку для управління проєктами.

4. Перелік графічного матеріалу_____
_____
_____
_____

5. Консультанти по роботі із зазначенням розділів, за якими здійснюється консультування:

Розділ	Консультант (прізвище, ініціали)	Підпис, дата	
		Завдання видав	Завдання прийняв
1	Базурін В.М.	09.12.2022 р.	09.12.2022 р.
2	Базурін В.М.	09.12.2022 р.	09.12.2022 р.
3	Базурін В.М.	09.12.2022 р.	09.12.2022 р.

6. Зміст випускного кваліфікаційної роботи (перелік питань за кожним розділом)

ВСТУП

РОЗДІЛ 1. ТЕОРЕТИЧНІ ОСНОВИ УПРАВЛІННЯ ПРОЄКТАМИ

1.1. Визначення управління проєктами та його важливість.

1.2. Огляд основних методологій управління проєктами (наприклад, PMBOK, PRINCE2).

1.3. Ключові принципи та практики управління проєктами.

РОЗДІЛ 2. ВИЗНАЧЕННЯ ВИМОГ ТА ФУНКЦІОНАЛЬНИХ МОЖЛИВОСТЕЙ WEB-ДОДАТКУ

2.1. Аналіз потреб користувачів та вимог до web-додатку для управління проєктами.

2.2. Специфікація функціональних можливостей додатку.

РОЗДІЛ 3. ПРОЕКТУВАННЯ WEB-ДОДАТКУ

3.1. Вибір технологічного стеку для розробки web-додатку.

3.2. Розробка архітектури web-додатку.

3.3. Дизайн інтерфейсу користувача (UI/UX) для додатку.

РОЗДІЛ 4. РОЗРОБКА ТА ТЕСТУВАННЯ WEB-ДОДАТКУ

4.1. Програмування та розробка функціональності додатку.

4.2. Програмний код.

ВИСНОВКИ

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

7. Календарний план виконання роботи

№ Пор.	Назва етапів випускної кваліфікаційної роботи	Строк виконання етапів роботи	
		За планом	фактично
1	2	3	4
1	*Вибір теми випускної кваліфікаційної роботи*	*01.11.2022*	*01.11.2022*
2	*Розробка та затвердження завдання на випускну кваліфікаційну роботу*	*09.12.2022*	*09.12.2022*
3	*Вступ*	*01.05.2023*	*01.05.2023*
4	*РОЗДІЛ 1. Теоретичні основи управління проєктами*	*14.06.2023*	*14.06.2023*
5	*Підготовка статті у збірник наукових статей магістрів*	*20.06.2023*	*20.06.2023*

6	*РОЗДІЛ 2. Визначення вимог та функціональних можливостей web-додатку*	*08.09.2023*	*08.09.2023*
7	*РОЗДІЛ 3.Проектування web-додатку*	*20.10.2023*	*20.10.2023*
8	*РОЗДІЛ 4. Розробка та тестування web-додатку*		
9	*Висновки*	*02.11.2023*	*02.11.2023*
10	*Здача випускної кваліфікаційної роботи на кафедру науковому керівнику*	*07.11.2023*	*07.11.2023*
11	*Попередній захист випускної кваліфікаційної роботи*	*17.11.2023*	*17.11.2023*
12	*Виправлення зауважень, зовнішнє рецензування випускної кваліфікаційної роботи*	*22.11.2023*	*22.11.2023*
13	*Представлення готової зшитої випускної кваліфікаційної роботи на кафедру*	*24.11.2023*	*24.11.2023*
14	*Публічний захист випускної кваліфікаційної роботи*	*За розкладом роботи ЕК*	

8. Дата видачі завдання «9» грудня 2022 р.

9. Керівник випускної кваліфікаційної роботи Базурін В.М.

(прізвище, ініціали, підпис)

10. Гарант освітньої програми Пурський О.І.

(прізвище, ініціали, підпис)

11. Завдання прийняв до виконання студент Стус.О.А

(прізвище, ініціали, підпис)

12. Відгук керівника випускної кваліфікаційної роботи

_____

_____

_____

_____

_____

_____

_____

_____

_____

_____

_____

_____

_____

_____

_____

_____

_____

_____

_____

_____

_____

_____

Керівник випускної кваліфікаційної роботи

_____

(підпис, дата)

13. Висновок про випускну кваліфікаційну роботу

Випускна кваліфікаційна робота студента _____

(прізвище, ініціали)

може бути допущена до захисту в екзаменаційній комісії.

Гарант освітньої програми_____Пурський О.І.____

(підпис, прізвище, ініціали)

Завідувач кафедри _____Пурський О.І.____

(підпис, прізвище, ініціали)

«_____»_____2023 р.

Анотація

У даній випускній кваліфікаційній роботі було проведено дослідження та розробку web-додатку для управління проєктами. Актуальність цього додатку полягає в тому, що сучасний бізнес потребує ефективного та організованого управління своїми проєктами. Традиційні методи вже не завжди задовольняють потреби компаній у швидкому доступі до інформації, спільній роботі команд, контролі за ресурсами та здійсненні стратегічних рішень.

Проекти можуть бути різних типів, таких як навчальні, логістичні, IT проекти та інші. Кожен проєкт має свою специфіку та вимоги, і саме цей web-додаток надає можливість користувачам ефективно керувати проектами різних типів.

У ході дослідження були досліджені основні принципи та практики управління проєктами, включаючи методології як PMBOK та PRINCE2. Це надає можливість користувачам здійснювати управління проєктами, дотримуючись найкращих практик.

У результаті роботи було створено web-додаток, який дозволяє організовувати проекти, встановлювати пріоритети та дедлайни, а також зберігати дані в локальному сховищі. Додаток може бути використаний для покращення ефективності управління проєктами та сприяти розвитку сучасних бізнес-практик у цій області.

Anotation

In this final qualification work, we researched and developed a web application for project management. The relevance of this application lies in the fact that modern business needs efficient and organized project management. Traditional methods no

longer always meet the needs of companies for quick access to information, teamwork, resource control, and strategic decision-making.

Projects can be of different types, such as training, logistics, IT, and other projects. Each project has its own specifics and requirements, and this web application allows users to effectively manage projects of various types.

The study examined the basic principles and practices of project management, including methodologies such as PMBOK and PRINCE2. This enables users to manage projects following best practices.

As a result of the work, a web application was created that allows you to organize projects, set priorities and deadlines, and store data in local storage. The application can be used to improve the efficiency of project management and promote the development of modern business practices in this area.

ЗМІСТ

ВСТУП

У сучасному світі ефективне управління проєктами є критично важливим для досягнення успіху в бізнесі. Швидкі технологічні зміни та розширення глобального ринку вимагають зручних та потужних інструментів для планування, виконання та моніторингу проєктів. У цьому контексті Web-додатки для управління проєктами стають все більш популярними, оскільки вони забезпечують зручний доступ до необхідних інструментів з будь-якого місця та пристрою.

Актуальність розробки Web-додатку для управління проєктами полягає в тому, що сучасний бізнес все більше прагне ефективного та організованого управління своїми проєктами. Традиційні методи вже не завжди задовольняють потреби компаній у швидкому доступі до інформації, спільній роботі команд, контролі за ресурсами та здійсненні стратегічних рішень.

Ось декілька аргументів, які підкреслюють актуальність розробки Web-додатку для управління проєктами:

1. Централізоване управління: Web-додаток дозволяє зберігати всю інформацію про проєкти, завдання, терміни, ресурси та комунікацію на одній платформі. Це дозволяє керівникам та командам з легкістю орієнтуватися в проєктній діяльності, забезпечувати взаємодію та злагодженість роботи.

2. Забезпечення ефективності: Web-додаток дозволяє автоматизувати багато рутинних завдань, таких як планування, моніторинг прогресу, призначення завдань, збір даних та звітність. Це допомагає економити час та зусилля, збільшує продуктивність команди та забезпечує вчасну реалізацію проєктів.

3. Комунікація та спільна робота: Web-додаток надає можливість зручної комунікації та спільної роботи між учасниками проєкту. Команди можуть обмінюватися повідомленнями, файлами, документами, а також відстежувати

зміни та коментувати завдання. Це сприяє збільшенню ефективності командної роботи та покращує комунікацію між учасниками проєкту.

4. Моніторинг та аналітика: Web-додаток забезпечує можливість моніторити прогрес проєктів, аналізувати дані, створювати звіти та панелі управління. Це дозволяє керівникам та командам вчасно виявляти проблеми, приймати обґрунтовані рішення та забезпечувати успішне завершення проєктів.

Загалом, розробка Web-додатку для управління проєктами є актуальною, оскільки вона дозволяє підвищити продуктивність, забезпечити ефективність та організованість управління проєктами, полегшити комунікацію та спільну роботу, а також надати зручні інструменти для моніторингу та аналітики. Вибір правильних технологій та інструментів важливий для досягнення успіху в розробці такого Web-додатку.

Мета роботи. Метою даної магістерської роботи є розробка Web-додатку для управління проєктами, який надасть зручний та ефективний інструмент для планування, керування та моніторингу проєктів.

Web-додаток для управління проектами є інтерактивним програмним забезпеченням, яке розробляється з метою допомогти організаціям та командам ефективно керувати своїми проектами через Інтернет. Він надає зручний та централізований доступ до інструментів та функціоналу, необхідних для планування, виконання та моніторингу проектних завдань.

Web-додаток для управління проектами може мати різноманітні функції, які допомагають користувачам керувати всіма аспектами свого проекту. Деякі загальні складові такого додатку включають:

1. Створення та організація проектів: Користувачі можуть створювати нові проекти, надавати їм назву, опис та інші важливі деталі. Вони також можуть організовувати проекти за категоріями або клієнтами.

2. Планування та розподіл завдань: Додаток надає засоби для планування та розподілу завдань між учасниками команди проекту. Він дозволяє

встановлювати терміни виконання, пріоритети та відповідальних осіб за кожне завдання.

3. Керування ресурсами: Додаток допомагає керувати ресурсами, такими як люди, матеріали та обладнання, необхідні для виконання проекту. Він може включати функції для призначення ресурсів, відстежування доступності та використання ресурсів.

4. Моніторинг прогресу: Користувачі можуть відстежувати прогрес виконання проекту, переглядати завершені та незавершені завдання, контролювати витрати часу та ресурсів. Додаток може надавати графіки, діаграми та звіти для візуалізації прогресу проекту.

5. Спільна робота та комунікація: Веб-додаток надає засоби для комунікації та спільної роботи між учасниками проекту. Він може включати можливості для обміну повідомленнями, коментування завдань, спільної роботи над документами та спілкування в реальному часі.

6. Збереження даних та архівування: Додаток забезпечує збереження даних про проекти, завдання та комунікацію. Він може надавати можливість створювати резервні копії, архівувати проекти та забезпечувати доступ до історії змін.

Завдання дослідження

Для досягнення поставленої мети необхідно виконати такі завдання:

1. Аналіз потреб та вимог користувачів у Web-додатку для управління проєктами.

2. Вибір технологій та інструментів для розробки Web-додатку.

3. Проектування архітектури Web-додатку, включаючи базу даних, серверну та клієнтську частини.

4. Розробка функціоналу Web-додатку, включаючи можливості планування проєктів, призначення завдань, відстеження прогресу та звітності.

5. Тестування та валідація Web-додатку для переконання в його працездатності та відповідності вимогам.

6. Впровадження Web-додатку та забезпечення підтримки його роботи.

Об’єкт дослідження:

Об’єктом дослідження є веб-додаток, спрямований на управління проєктами. Включаючи елементи інтерфейсу, базу даних, функціональні можливості та інші компоненти, які визначають його структуру та функціональність.

Предмет дослідження:

Предметом дослідження є процес розробки та впровадження веб-додатку для управління проєктами. Це включає в себе аналіз потреб користувачів, визначення функціональних вимог, вибір технологічного стеку, програмування, тестування та імплементацію.

Методи дослідження:

Дослідження базується на методах аналізу літературних джерел, вивченні сучасних підходів до управління проєктами, а також на практичних етапах розробки веб-додатку. Методологія включає в себе аналіз вимог, проектування архітектури, програмування, тестування та вдосконалення на кожному етапі розробки.

Наукова новизна одержаних результатів:

Наукова новизна полягає в розробці веб-додатку, який інтегрує сучасні підходи до управління проєктами з використанням сучасних технологій веб-

розробки. Отримані результати сприяють покращенню ефективності управління проєктами та використанню новітніх інструментів для зручності користувачів.

Практичне значення:

Розроблений веб-додаток має практичне значення для бізнесу та організацій, які прагнуть покращити свої процеси управління проєктами. Його впровадження може призвести до збільшення ефективності, зменшення ризиків та поліпшення комунікації в командах проєктів.

Структура та обсяг випускної кваліфікаційної роботи. Випускна кваліфікаційна робота складається із вступу, чотирьох розділів, висновків, списку використаних джерел із 38 найменувань, і містить 33 сторінки основного тексту і 3 рисунки.

РОЗДІЛ 1. ТЕОРЕТИЧНІ ОСНОВИ УПРАВЛІННЯ ПРОЄКТАМИ

1.1. Визначення управління проєктами та його важливість

Управління проєктами - це систематичний підхід до планування, виконання, контролю та завершення проєктів з метою досягнення конкретних цілей і завдань. Проєкти є тимчасовими зусиллями, спрямованими на створення унікального продукту, послуги або результату в рамках обмежених ресурсів, таких як бюджет, час та обмеження ресурсів.

Важливі аспекти визначення управління проєктами:

Систематичний підхід: Управління проєктами включає в себе використання структурованих методологій, процесів та інструментів для керування проєктами від початку до завершення. Це означає, що усі аспекти проєкту, включаючи цілі, ресурси, ризики та комунікації, ретельно плануються та систематично контролюються.

Цільове спрямування: Основною метою управління проєктами є досягнення конкретних цілей та результатів. Ці цілі можуть включати в себе створення нового продукту, впровадження нової послуги, покращення існуючих процесів, або будь-яке інше завдання, яке потребує координації та управління.

Тимчасовий характер: Проєкти мають тимчасовий характер, що означає, що вони мають чітко визначений початок і кінець. Ця обмеженість у часі допомагає визначити обсяг проєкту та планувати роботу.

Обмежені ресурси: Управління проєктами проводиться в умовах обмежених ресурсів, включаючи бюджет, людські ресурси, матеріали та обладнання. Діяльність з управлінням проєктами полягає в оптимізації використання цих ресурсів.

Унікальність: Кожен проєкт є унікальним і вимагає специфічного підходу. Навіть якщо організація вже виконувала подібні проекти, кожен з них має свої власні особливості та вимоги.

Важливість управління проєктами:

Управління проєктами є надзвичайно важливим аспектом в сучасному бізнес-середовищі. Ось деякі аспекти, що підкреслюють його важливість:

Досягнення стратегічних цілей: Управління проєктами допомагає організаціям досягати своїх стратегічних цілей, впроваджувати нові ідеї, розширювати ринки та покращувати існуючі продукти та послуги.

Ефективне розподілення ресурсів: Управління проєктами допомагає організаціям оптимізувати використання ресурсів, включаючи фінансові, людські, матеріальні та технічні ресурси.

Мінімізація ризиків: Завдяки управлінню ризиками, організації можуть передбачати можливі проблеми та вживати заходи для їх запобігання або легшого вирішення.

Підвищення продуктивності та якості: Ефективне управління проєктами сприяє підвищенню продуктивності роботи та якості виконання завдань.

Забезпечення відповідності строкам і бюджету: Управління проєктами допомагає виконувати проекти в обумовлених рамках строків та бюджету.

Удосконалення комунікації та співпраці: Управління проєктами покращує комунікацію в організації та сприяє співпраці між різними відділами та командами.

В цілому, управління проєктами є необхідною складовою успішного бізнесу та допомагає організаціям досягати своїх стратегічних цілей, зберігати конкурентну перевагу та забезпечувати стабільний розвиток.

1.2. Огляд основних методологій управління проєктами (PMBOK, PRINCE2)

Управління проєктами може використовувати різні методології та підходи для досягнення своїх цілей. Двома з найбільш відомими та широко використовуваними методологіями є PMBOK (Project Management Body of Knowledge) та PRINCE2 (Projects IN Controlled Environments).

PMBOK (Project Management Body of Knowledge):

PMBOK є однією з найбільш відомих та використовуваних методологій управління проєктами. Вона була розроблена Інститутом управління проєктами (PMI) і визначає стандартні підходи та керівництво для управління проєктами. Важливим є те, що PMBOK не є конкретною методологією для управління проєктами, а замість цього вона надає набір загальних принципів та підходів, які можуть бути використані в різних сферах та галузях.

Основні характеристики PMBOK:

Процеси та групи процесів: PMBOK визначає п'ять основних груп процесів управління проєктом, включаючи ініціацію, планування, виконання, моніторинг та контроль, а також закриття. Кожна з цих груп містить рядок процесів, які повинні бути виконані для успішного управління проєктом.

Знання з управління проєктом: PMBOK визначає десять ключових знань, які включають в себе усі аспекти управління проєктом, від створення статусних звітів до управління комунікаціями та ризиками.

Ролі та обов'язки: PMBOK надає чіткі визначення ролей та обов'язків для учасників проєкту, включаючи проектного менеджера, спонсора проєкту, стейкхолдерів та інших.

Засоби та техніки: PMBOK надає огляд засобів та технік, які можуть бути використані для керування проєктом, включаючи методи планування, аналіз ризиків, графічні інструменти та багато інших.

Шаблони та документація: PMBOK включає в себе шаблони та приклади документації, які можуть бути використані для створення необхідних документів під час управління проєктом.

PMBOK є важливим ресурсом для професіоналів, які працюють у галузі управління проєктами та бажають дотримуватися найкращих практик. Вона допомагає стандартизувати підходи до управління проєктами та забезпечує

базовий каркас для успішного виконання проєктів у різних галузях та контекстах.

PRINCE2 (Projects IN Controlled Environments):

PRINCE2 є методологією управління проєктами, розробленою в Великобританії. Ім'я PRINCE2 походить від "Projects IN Controlled Environments," що підкреслює фокус на контрольованих середовищах для управління проєктами. PRINCE2 надає структурований підхід для керівництва проєктом та забезпечує фреймворк для управління проєктом в усіх аспектах його життєвого циклу.

Основні характеристики PRINCE2:

Сім принципів: PRINCE2 визначає сім принципів, на яких базується методологія. Ці принципи включають в себе забезпечення бізнес-цілей, розділення проєкту на керовані етапи, оцінку змін та ризиків, а також включення стейкхолдерів у процес управління.

Ролі та обов'язки: PRINCE2 надає чіткі ролі та обов'язки учасників проєкту. Ключові ролі включають керівника проєкту (Project Manager), замовника проєкту (Project Board), керівників етапів (Stage Managers) та інших.

Специфікація етапів та продуктів: PRINCE2 вимагає визначення чітких етапів проєкту та встановлення критеріїв успішності для продуктів, які мають бути створені протягом цих етапів.

Процеси: PRINCE2 визначає сім ключових процесів, включаючи старт проєкту (Starting up a Project), оцінку і ініціацію проєкту (Initiating a Project), керування проєктом (Directing a Project), ініціювання та планування проєкту (Initiating and Planning a Project), контроль стадій (Controlling a Stage), управління змінами (Managing Product Delivery), а також закриття проєкту (Closing a Project).

Документація та шаблони: PRINCE2 включає в себе шаблони та приклади документації, які допомагають учасникам проєкту створювати необхідні документи та звіти.

PRINCE2 надає структурований та професійний підхід до управління проєктами, що допомагає забезпечити успішну реалізацію проєктів в усіх галузях та галузях. Вона дозволяє організаціям забезпечити контроль, розклад та відповідність бізнес-цілям у кожному етапі проєкту.

1.3 . Ключові принципи та практики управління проєктами.

Управління проєктами - це складний процес, який включає в себе ряд принципів та практик для досягнення успішних результатів. Давайте розглянемо детально ключові принципи та практики управління проєктами:

Ключові принципи управління проєктами:

1. Спрямованість на цілі (Goal Orientation): Кожен проєкт повинен мати чітку мету і цілі, які визначають, що має бути досягнуто в результаті проєкту. Ці цілі повинні бути спрямовані на задоволення потреби організації або клієнта.

2. Систематичний підхід (Systematic Approach): Управління проєктом передбачає використання структурованих методологій та процесів для планування, виконання, моніторингу та контролю проєкту. Цей систематичний підхід допомагає уникнути хаосу та непередбачуваності.

3. Максимальна використаність ресурсів (Resource Optimization): Ефективне управління проєктом передбачає оптимізацію використання ресурсів, включаючи бюджет, час, людські ресурси та обладнання. Принцип полягає в тому, щоб досягти максимальної продуктивності та ефективності при обмежених ресурсах.

4. Стейкхолдерський підхід (Stakeholder Focus): Управління проєктом враховує потреби та очікування всіх стейкхолдерів, включаючи замовника,

команду проєкту, користувачів та інших зацікавлених сторін. Важливо забезпечити співпрацю та комунікацію з усіма стейкхолдерами.

5. Адаптивність (Adaptability): Управління проєктом повинно бути готовим до змін та реагувати на них. Зміни вимагають адаптації стратегії, планування та ресурсів для досягнення цілей проєкту.

Ключові практики управління проєктами:

1. Планування проєкту (Project Planning): Розроблення докладного плану проєкту, включаючи визначення завдань, ресурсів, графіків та бюджету. Планування допомагає забезпечити структурованість та організацію усіх аспектів проєкту.

2. Моніторинг та контроль (Monitoring and Control): Постійне відстеження та контроль прогресу проєкту за допомогою метрик та ключових показників продуктивності. Це дозволяє вчасно виявляти проблеми та реагувати на них.

3. Звітність та комунікація (Reporting and Communication): Регулярна звітність про стан проєкту та комунікація з усіма стейкхолдерами. Грамотна комунікація допомагає уникнути недорозумінь і підтримувати співпрацю.

4. Управління ризиками (Risk Management): Виявлення, оцінка та керування ризиками проєкту. Підходить для передбачення можливих проблем та впровадження стратегій для їх управління.

5. Закриття проєкту (Project Closure): Оцінка результатів проєкту та виконання всіх завершальних процедур, включаючи документацію та звіти. Закриття проєкту важливо для підведення підсумків та винесення уроків для майбутніх проєктів.

6. Управління змінами (Change Management): Систематичний підхід до управління змінами, включаючи виявлення, оцінку, внесення змін та їх впровадження в проєкт.

7. Розвиток команди (Team Development): Створення та підтримка ефективної та спільної команди проєкту. Це включає в себе мотивацію, навчання та співпрацю учасників проєкту.

Застосування цих принципів та практик в управлінні проєктами допомагає забезпечити успішну реалізацію проєкту, досягнення цілей та вдосконалення процесів.

РОЗДІЛ 2. ВИЗНАЧЕННЯ ВИМОГ ТА ФУНКЦІОНАЛЬНИХ МОЖЛИВОСТЕЙ WEB-ДОДАТКУ

2.1 Аналіз потреб користувачів та вимог до web-додатку

Аналіз потреб користувачів та вимог до web-додатку для управління проєктами є ключовим етапом у розробці програмного забезпечення, оскільки визначає функціональність та характеристики, які додаток повинен надавати для задоволення вимог своїх користувачів. Врахування потреб користувачів гарантує, що розроблений продукт буде корисним і задовольнить їхні очікування. Нижче подано детальний аналіз потреб користувачів та вимог до web-додатку для управління проєктами:

1. Створення та Організація Проєктів:

- Потреба:
    - Можливість легко створювати нові проекти та організовувати їх для зручного використання.

2. Завдання та Пріоритети:

- Потреба:
    - Можливість додавати завдання до проектів.
    - Встановлення пріоритетів та термінів виконання для завдань.

3. Локальне Зберігання та Доступ:

- Потреба:
    - Збереження даних проекти та завдань локально на пристрої користувача.
    - Зручний доступ до збережених даних.

4. Інтерфейс та Взаємодія:

- Потреба:
    - Інтуїтивний та дружелюбний інтерфейс користувача.
    - Можливість легко взаємодіяти з проектами та завданнями.

5. Відстеження Термінів та Прогресу:

- Потреба:
    - Відстеження термінів виконання завдань та проектів.
    - Відображення прогресу виконання.

6. Модульність та Розширення:

- Потреба:
    - Можливість розширення функціональності за допомогою модульної системи.

7. Безпека та Приватність:

- Потреба:
    - Забезпечення конфіденційності та безпеки збережених даних.

8. Кросплатформеність:

- Потреба:
    - Забезпечення доступу до додатку на різних платформах (пристрої з ОС Windows, macOS, Linux; мобільні пристрої з ОС iOS та Android).

9. Оптимізація Продуктивності:

- Потреба:
    - Оптимізація роботи додатку для швидкого та ефективного використання.

10. Підтримка Аналітики:

- Потреба:
    - Можливість відстеження та аналізу продуктивності проектів та користувачів.

2.2. Специфікація функціональних можливостей додатку

Специфікація функціональних можливостей додатку є ключовим етапом в розробці web-додатку для управління проєктами. Вона визначає, які конкретні функції та можливості повинен надавати додаток для задоволення потреб користувачів. Нижче наведено детальну специфікацію функціональних можливостей додатку:

1. **Створення проєкту**: Користувачі повинні мати можливість створювати нові проекти та надавати їм назву, опис, терміни виконання і інші важливі атрибути.
2. **Планування проєкту**: Додаток повинен дозволяти користувачам створювати графіки проєкту, розподіляти завдання та ресурси, визначати терміни та пріоритети.
3. **Призначення завдань**: Користувачі повинні мати можливість призначати завдання учасникам команди, встановлювати терміни виконання та створювати зв'язки між завданнями.

4. **Відстеження прогресу**: Додаток повинен надавати можливість відстежувати прогрес виконання завдань, оцінювати завдання та фіксувати зміни.
5. **Спільна робота команди**: Користувачі повинні мати можливість спільно працювати над завданнями, обмінюватися коментарями, вкладати файли та спілкуватися в реальному часі.
6. **Аналітика проєкту**: Додаток повинен надавати можливість аналізувати дані проєкту, включаючи витрати, часові рамки, відомості про завдання та інші показники.
7. **Звітність**: Користувачі повинні мати можливість створювати звіти про прогрес проєкту, включаючи графіки, таблиці та графічну інформацію.
8. **Безпека та доступність**: Додаток повинен забезпечувати безпеку даних, обмеження доступу до функцій відповідно до ролей користувачів та забезпечення доступності додатку.
9. **Інтеграція з іншими інструментами**: Додаток повинен мати можливість інтеграції з іншими інструментами, такими як електронна пошта, календарі, чати та інші сервіси.
10. **Масштабованість та продуктивність**: Додаток повинен бути масштабованим і забезпечувати стабільну продуктивність навіть при великій кількості користувачів і проектів.
11. **Підтримка та навчання користувачів**: Додаток повинен надавати можливість підтримки користувачів, включаючи онлайн-підтримку та навчання користувачів.

РОЗДІЛ 3. ПРОЕКТУВАННЯ WEB-ДОДАТКУ

3.1. Вибір технологічного стеку для розробки web-додатку.

Вибір технологічного стеку для розробки web-додатку має велике значення, оскільки від цього залежить якість, продуктивність і майбутні можливості додатку. У даному випадку, технологічний стек для розробки web-додатку включає в себе HTML, CSS, JavaScript (JS), NPM та Webpack. Давайте розглянемо, чому саме цей стек було обрано:

1. HTML, або HyperText Markup Language, є стандартною мовою розмітки для створення web-сторінок. Вона використовується для визначення структури та вмісту web-сторінок, які браузери можуть інтерпретувати та відображати користувачам. HTML включає в себе набір тегів і атрибутів, які використовуються для розміщення тексту, зображень, відео, посилань та інших елементів на web-сторінці. Розглянемо основні аспекти HTML:

1. **Теги**: HTML використовує теги для визначення елементів на сторінці. Теги зазвичай мають вигляд **<назва_тегу>**, наприклад, **<p>** для параграфу або **<img>** для зображення.
2. **Елементи**: Кожен HTML-тег створює відповідний HTML-елемент на сторінці. Елемент може містити текст, інші теги, атрибути та інше.
3. **Атрибути**: Теги можуть мати атрибути, які використовуються для надання додаткової інформації про тег або його змінювання. Наприклад, атрибут **src** вказує на джерело зображення в тезі **<img>**.
4. **Вкладеність**: HTML-теги можуть бути вкладені один в одного, утворюючи ієрархію елементів на сторінці. Наприклад, **<ul>** (список) може містити багато **<li>** (пункти списку).

5. **Структура**: Валідний HTML-документ має визначену структуру, яка включає в себе заголовок **<head>**, тіло **<body>**, посилання на зовнішні ресурси, такі як стилі і скрипти, та інші елементи.
6. **Версія HTML**: Кожен HTML-документ повинен вказувати версію HTML, яку він використовує. Наприклад, для HTML5 вказується **<!DOCTYPE html>**.
7. **Коментарі**: У HTML можна вставляти коментарі, які не відображаються у відвідувачів, але можуть бути корисними для розробників.
8. **Семантика**: HTML включає в себе багато тегів, які використовуються для надання семантичного значення вмісту. Наприклад, теги **<header>**, **<nav>**, **<section>** та інші допомагають вказати роль та значення кожного елемента на сторінці.
9. **Посилання**: HTML дозволяє створювати посилання на інші сторінки, ресурси чи місця на поточній сторінці. Посилання використовують тег **<a>**.
10. **Зображення та мультимедіа**: HTML підтримує вставку зображень, аудіо та відео за допомогою відповідних тегів, таких як **<img>**, **<audio>** та **<video>**.

HTML є основою для створення web-сторінок та web-додатків. Він працює разом з CSS (для стилізації) та JavaScript (для додавання інтерактивності), щоб надавати користувачам багатофункціональний та зручний досвід в інтернеті.

2. CSS, або Cascading Style Sheets, є мовою, яка використовується для стилізації та вигляду web-сторінок. Вона визначає, як елементи HTML відображаються на екрані, включаючи кольори, розміри, розташування та інші аспекти дизайну. CSS розділяється на декілька ключових аспектів:

1. **Селектори**: CSS використовує селектори для вибору конкретних елементів HTML, до яких будуть застосовані стилі. Селектори можуть бути тегами (наприклад, **p** для параграфів), класами (наприклад, **.клас**),

ідентифікаторами (наприклад, **#ідентифікатор**), псевдокласами (наприклад, **:hover** для стану наведення) та іншими способами.

2. **Властивості**: Кожен стиль CSS включає одну або більше властивостей, які визначають конкретний аспект дизайну. Наприклад, властивість **color** визначає колір тексту, а властивість **font-size** - розмір шрифту.
3. **Значення**: Для кожної властивості визначаються значення, які вказують, як саме ця властивість має бути застосована до вибраних елементів. Значення можуть бути вказані в різних форматах, таких як кольори, розміри, шрифти, відстані тощо.
4. **Правила**: CSS правила складаються з селектора та відповідних властивостей та значень.
5. **Каскадність**: CSS має механізм каскаду, що дозволяє визначити, який стиль буде використовуватися, якщо на один і той же елемент застосовано кілька правил. Це дозволяє робити зміни в стилях залежно від контексту та пріоритету.
6. **Спадкування**: Багато стилів спадковуються від батьківських елементів. Це означає, що стиль, визначений на вищому рівні структури сторінки, може впливати на всі дочірні елементи. Наприклад, стиль, визначений на рівні **<body>**, може застосовуватися до всього вмісту сторінки.
7. **Властивості Box Model**: Box Model - це модель, яка описує, як розраховується розмір і розташування блокових елементів в CSS. Вона включає в себе властивості, такі як **margin**, **border**, **padding** та **width**, які визначають, як елементи розміщуються і відображаються відносно один одного.

3. JavaScript є однією з найпоширеніших мов програмування, яка використовується для розробки web-сайтів. Вона надає можливість створення динамічного та інтерактивного web-змісту. Ось ключові аспекти JavaScript:

1. **Скриптова мова програмування**: JavaScript - це мова, яка виконується на клієнтському боці (в браузері), тобто вона використовується для взаємодії з користувачем або зміни вмісту web-сторінки під час її відображення.
2. **Динамічність**: Використання JavaScript дозволяє змінювати вміст сторінки в залежності від подій, які відбуваються (наприклад, клік на кнопку або завантаження сторінки), що робить сторінку більш інтерактивною та зручною для користувачів.
3. **Можливості маніпулювання DOM**: JavaScript надає доступ до DOM (Document Object Model), що представляє структуру сторінки. Це дозволяє змінювати, видаляти та додавати елементи, змінювати їх стилі, обробляти події тощо.
4. **Функції та обробники подій**: JavaScript використовує функції як основний будівельний блок програм. Також, він дозволяє призначати обробники подій, наприклад, реагувати на клік миші або натискання клавіші.
5. **Застосування AJAX**: JavaScript може використовуватися для виконання асинхронних запитів до сервера без перезавантаження сторінки. Це дозволяє оновлювати вміст сторінки без перерви в роботі користувача.
6. **Об'єктно-орієнтоване програмування (ООП)**: JavaScript підтримує об'єктно-орієнтований підхід до програмування, де дані та функції упаковані в об'єкти, що сприяє більшій організації та повторному використанню коду.
7. **Розширені можливості**: З появою ES6 (ECMAScript 2015) та подальших версій стандарту, JavaScript отримав нові можливості, такі як стрілкові функції, генератори, класи, розширені можливості роботи з рядками, масивами та інші зручні засоби для програмістів.

4. NPM, або Node Package Manager, є одним з найпоширеніших менеджерів пакетів для JavaScript. Він є частиною Node.js, платформи для виконання JavaScript поза браузером, і використовується для управління залежностями та пакетами у JavaScript-проєктах.

Основні аспекти NPM:

1. **Управління залежностями**: Однією з ключових можливостей NPM є здатність управляти залежностями проекту. Він дозволяє вказувати необхідні пакети та їх версії в файлі **package.json**, та потім завантажувати та встановлювати їх локально в проекті.
2. **Робота з пакетами**: NPM надає доступ до тисяч пакетів та модулів, які можна використовувати в проєктах. Розробники можуть шукати, встановлювати та оновлювати пакети для використання у своїх програмах.
3. **Сценарії виконання (scripts)**: **package.json** дозволяє визначати спеціальні скрипти, які можна викликати через NPM. Це може бути все, від скриптів для компіляції коду до автоматизації тестування чи розгортання проекту.
4. **Локальне кешування та керування версіями**: NPM зберігає завантажені пакети локально, що дозволяє їх використання без постійного завантаження з мережі. Також, він дозволяє контролювати версії пакетів для уникнення конфліктів у версіях під час розробки.
5. **Модульність та розширені можливості**: NPM надає широкі можливості для розробників, спрощуючи процес створення та розповсюдження власних пакетів, створення приватних репозиторіїв, а також дозволяє використовувати різні версії Node.js для різних проєктів.
6. **Інтеграція з репозиторіями**: В NPM є можливість інтегруватися з такими системами контролю версій, як Git, дозволяючи встановлювати пакети безпосередньо з репозиторію.

NPM є потужним інструментом у web-розробці, що допомагає розробникам керувати залежностями, швидко встановлювати пакети та автоматизувати різні процеси розробки програмного забезпечення.

5. Webpack - це інструмент для збірки (бандлінгу) і оптимізації ресурсів у web-розробці. Він дозволяє розробникам об'єднувати та оптимізувати файли JavaScript, CSS, зображення та інші ресурси для web-сайту або додатку. Ось ключові аспекти Webpack:

1. **Збірка модульна**: Webpack підтримує модульний підхід до розробки, де код розбивається на окремі модулі. Це дозволяє розробникам організовувати свій код та використовувати залежності між модулями.
2. **Збільшення продуктивності**: Webpack автоматично оптимізує та зменшує розмір файлів, що завантажуються на сторінку. Це допомагає підвищити продуктивність web-сайту та зменшити час завантаження.
3. **Підтримка різних ресурсів**: Ви не обмежені лише JavaScript і CSS. Webpack підтримує зображення, шрифти, JSON-файли та інші ресурси. Він також дозволяє використовувати препроцесори CSS, такі як SASS або LESS.
4. **Плагіни та лоадери**: Webpack може розширюватися за допомогою плагінів та лоадерів. Плагіни дозволяють виконувати різні завдання під час збірки (наприклад, стискувати код або генерувати HTML-сторінки), а лоадери дозволяють обробляти різні типи файлів під час їх інтеграції в проект.
5. **Code splitting**: Webpack підтримує розділення коду (code splitting), що дозволяє завантажувати лише необхідний код для конкретної сторінки або дії, покращуючи продуктивність.
6. **Гаряча заміна модулів (Hot Module Replacement, HMR)**: Webpack підтримує HMR, що дозволяє розробникам збільшити швидкість

розробки, оновлюючи тільки ті частини сторінки, які були змінені, замість повного перезавантаження.

7. **Спільнота і підтримка**: Webpack - це відкрите джерело, і він має активну спільноту розробників, яка постійно оновлює та покращує інструмент. Є багато різних плагінів та лоадерів, що розроблені спільнотою, щоб спростити роботу з Webpack.

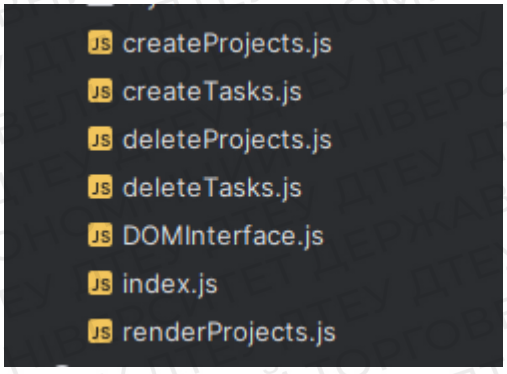
Webpack допомагає розробникам підтримувати чистий та організований код, оптимізувати продуктивність web-сайту та полегшує процес збірки та розгортання проєктів web-розробки.

3.2. Розробка архітектури web-додатку.

Компоненти архітектури:

1. Фронтенд-інтерфейс: Цей компонент відповідає за користувацький інтерфейс та взаємодію з користувачем. Він включає HTML, CSS і JavaScript для створення і відображення списків проектів і завдань, форм та інших елементів інтерфейсу.

2. Модулі JavaScript: Код JavaScript можна організувати в модулі на основі функціональності. Наприклад, ви можете мати модулі для управління проектами, управління завданнями, автентифікації користувачів та роботи з локальним сховищем. Ці модулі повинні взаємодіяти з DOM для оновлення інтерфейсу користувача.



Мій додаток складається з таких модулів:

1. createProjects – створення проєктів
2. createTasks – створення тасків
3. deleteProjects – видалення проєктів
4. deleteTasks – видалення тасків
5. DOMInterface – інтерфейс
6. Index – вхідна точка для модулів
7. renderProjects – рендер зображень, такі як гіфка, таблиці та інші, які знаходяться в папці dist

3. Webpack: Webpack використовується як пакувальник модулів для об'єднання всіх модулів JavaScript, таблиць стилів та інших ресурсів в єдиний пакет, який може бути поданий браузеру клієнта. Це також допомагає з мінімізацією та оптимізацією ресурсів.

4. Локальне сховище: Використовуйте локальне сховище браузера для зберігання даних проекту і завдань на стороні клієнта. Це дозволяє користувачам зберігати свої дані і мати доступ до них навіть після закриття браузера.

Де знаходиться localStorage?

Найпростіший спосіб побачити, який фізичний вигляд має сховище localStorage, - це відкрити вкладку Application всередині так званих Інструментів Розробника браузера (Google Chrome).

5. Бібліотека date-fns: Використовуйте бібліотеку date-fns для роботи з датами та управління дедлайнами у вашому додатку. Вона надає різні функції маніпулювання датою і часом для спрощення операцій, пов'язаних з датами.

6. Завантажувач CSS: Використовуйте завантажувач CSS для імпорту та завантаження стилів CSS у ваш додаток. Це допоможе стилізувати ваш web-додаток і зробити його візуально привабливим.

7. Завантажувач ресурсів: Реалізуйте завантажувач ресурсів для ефективного завантаження ресурсів, пов'язаних з проектом, таких як зображення, іконки та інші ресурси, необхідні для користувацького інтерфейсу.

Робочий процес користувача:

1. Користувачі взаємодіють з інтерфейсом, де вони можуть створювати проекти, додавати завдання, встановлювати пріоритети і терміни.

2. Модулі JavaScript обробляють дії користувачів, оновлюють DOM та керують даними, використовуючи локальне сховище.

3. Webpack об'єднує весь JavaScript і таблиці стилів для оптимізованої доставки клієнту.

Потік даних:

1. Дії користувача викликають події у фронтенд-інтерфейсі.

2. Модулі JavaScript перехоплюють ці події, обробляють дані та відповідно оновлюють DOM.

3. Локальне сховище використовується для збереження і отримання даних проекту і завдань, забезпечуючи постійність між сеансами користувача.

4. Бібліотека date-fns допомагає керувати даними, пов'язаними з датами, зокрема дедлайнами.

3.3. Дизайн інтерфейсу користувача (UI/UX) для додатку.

Мій web-додаток вітає користувачів на головній сторінці, де першим елементом, який привертає увагу, є особливе поєднання кольорів - світло-сірого та вкрай блідого блакитного. Цей вибір кольорів створює атмосферу спокою та зосередженості, ідеальну для роботи та планування.

Однією з ключових частин головної сторінки є панель проєктів, розташована видною області, що включає бокове меню. Однак, якщо користувач прагне зосередитися на завданнях, він має можливість приховати цю панель. Панель проєктів миттєво пропонує огляд створених проєктів і дає

можливість їх видалити, а також запрошує створити новий проєкт, коли це необхідно.

Нижче розташована панель тасків, яка завжди відкрита і доступна. Тут ви можете призначити завдання конкретному проєкту. При цьому обрання проєкту стає обов'язковим етапом перед створенням завдання, що гарантує структурованість та організованість вашого робочого процесу. Під час створення завдання, вам належить визначити назву завдання, визначити його пріоритетність, та встановити дату виконання.

Коли завдання створено, панель тасків автоматично відображає вам час, який залишився до дедлайну, що надає вам реального контролю над вашими завданнями. Якщо ви виконали завдання раніше запланованої дати, ви завжди можете позначити його як виконане і відзначити свій успіх.

У разі, якщо завдання втратило актуальність або було завершено і не потребується більше відслідковування, є можливість його видалити для підвищення ефективності та чіткості списку завдань.

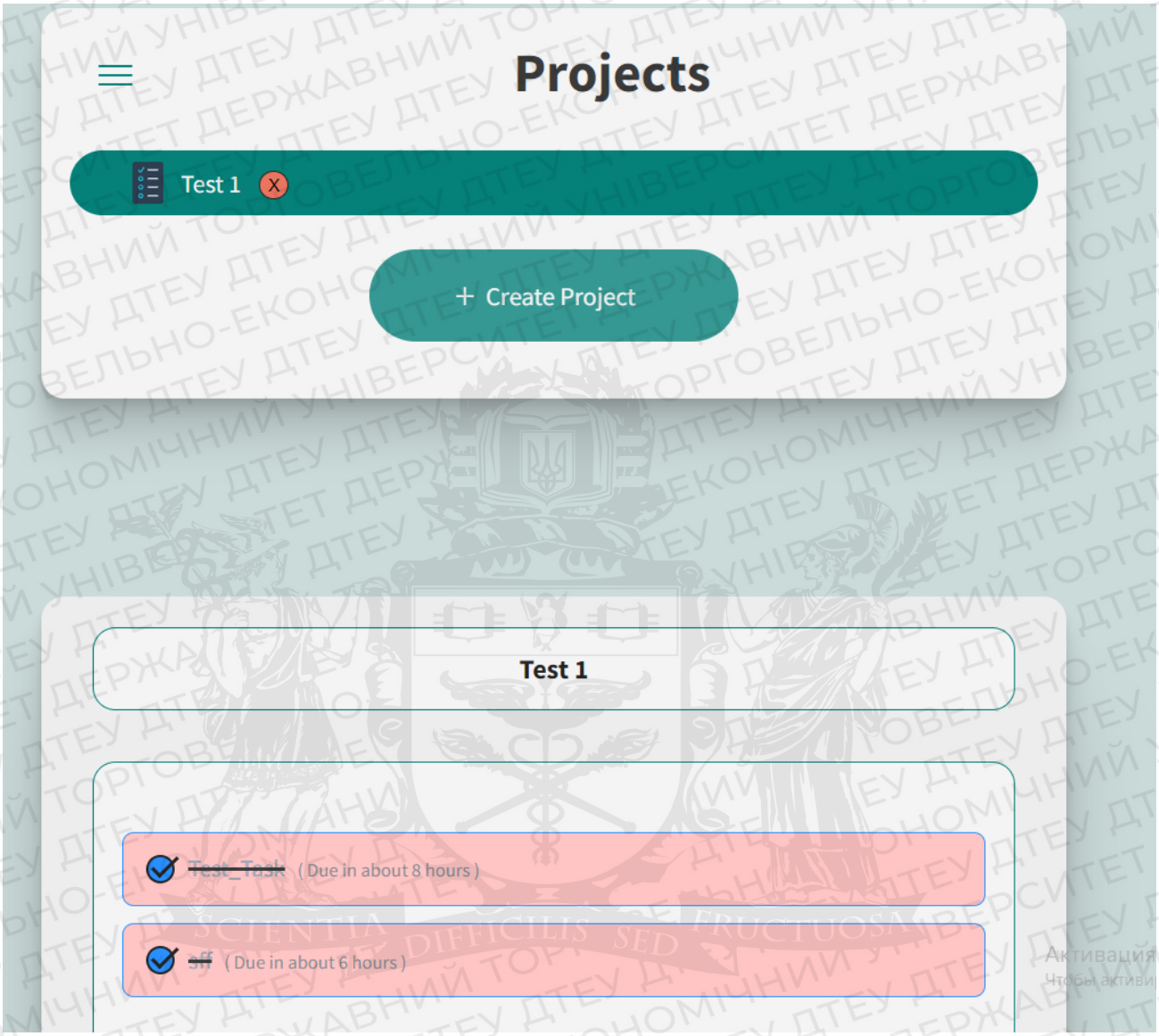
Завершуючи роботу в додатку, на вас чекає невелика гіфка, яка призначена для підняття вашого настрою та надання позитивної нотки після завершення робочого дня.

РОЗДІЛ 4. РОЗРОБКА ТА ТЕСТУВАННЯ WEB-ДОДАТКУ

4.1. Програмування та розробка функціональності додатку.

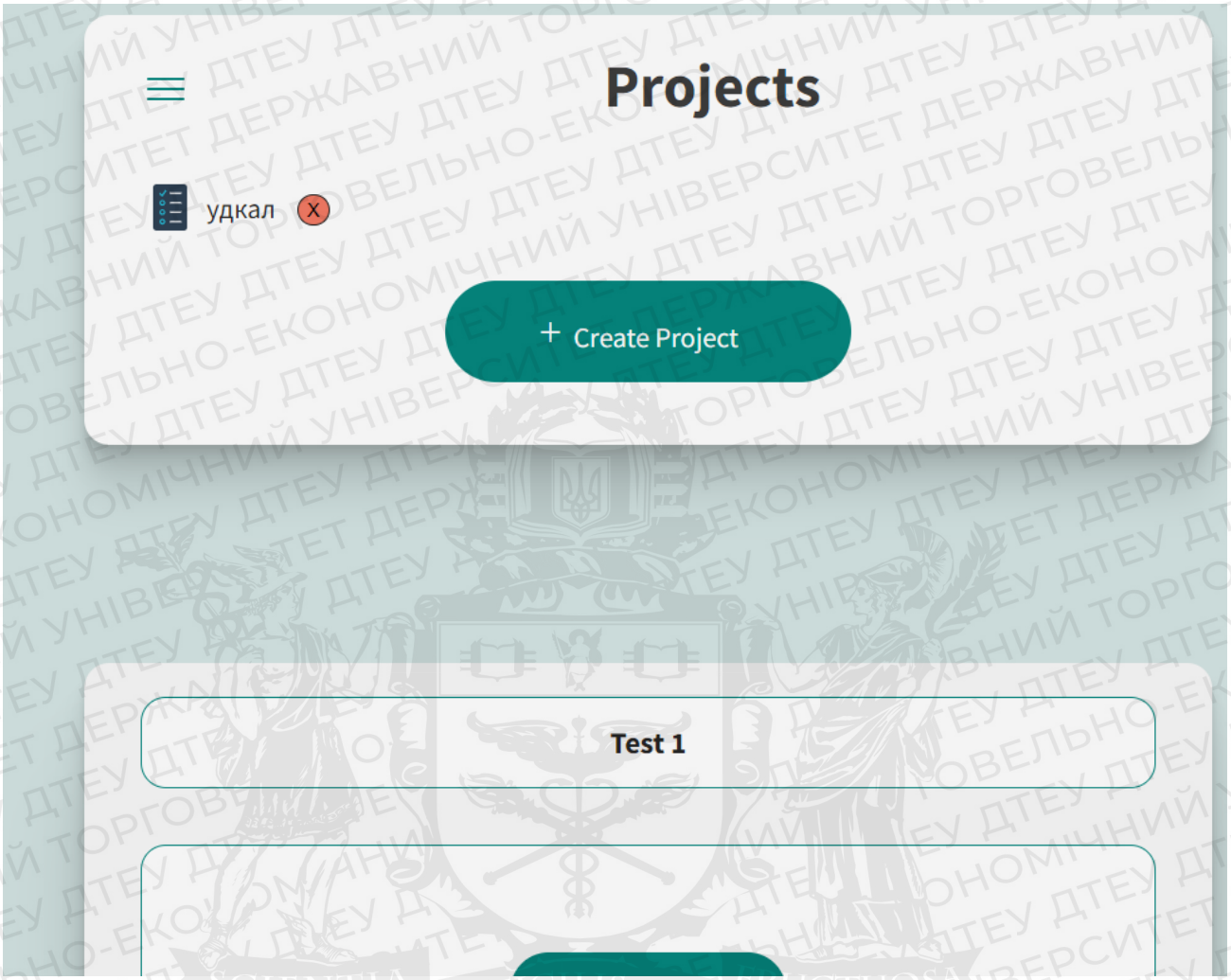
Програмування та розробка функціональності web-додатку для управління проєктами включає в себе створення коду та реалізацію всіх необхідних функцій, які дозволять користувачам планувати, виконувати та моніторити проекти. Ось декілька ключових етапів та функцій, які можуть бути важливими при програмуванні додатку:

1. **Дашборд та головне меню**:
    - Створення головного екрану (дашборду) для відображення списку проектів та основних функцій.



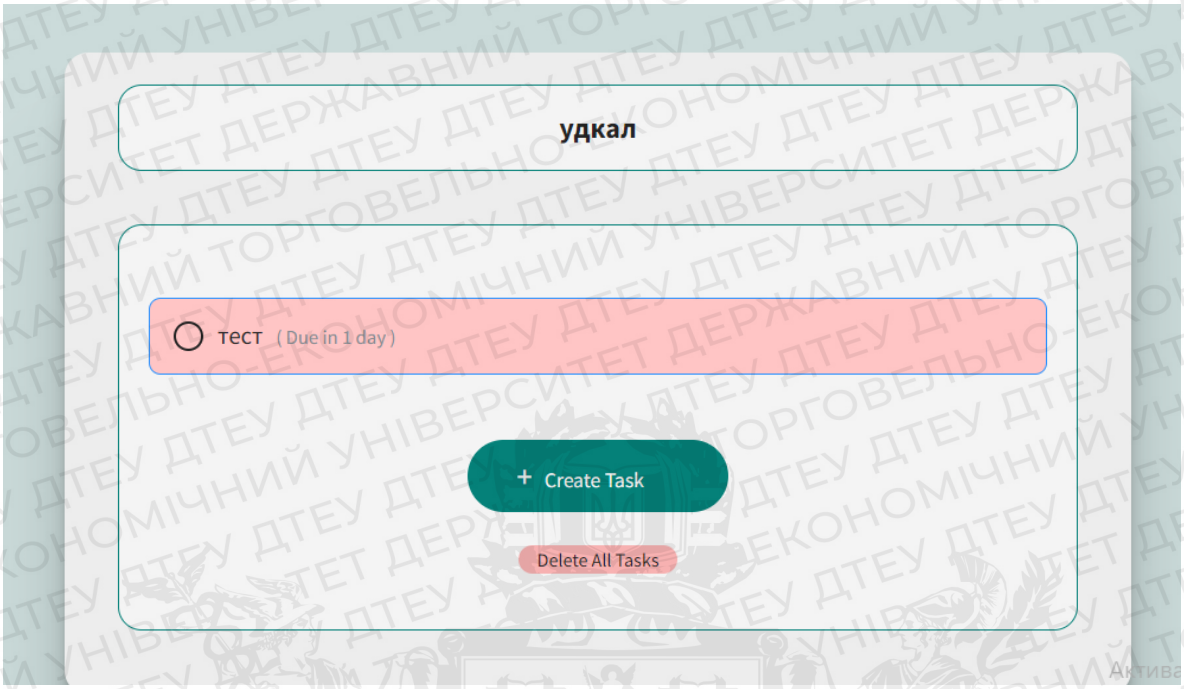
2. **Створення та керування проектами**:

- Можливість створювати нові проекти та вказувати інформацію про них (назва, опис, дедлайни тощо).
- Можливість редагувати та видаляти проекти та завдання.



3. **Моніторинг та відстеження прогресу**:

- Відображення статусу та прогресу виконання завдань та проектів.



4. **Тестування та валідація**:
    - Виконання тестів для переконання в правильності функціональності та відсутності помилок.
5. **Документація та підтримка**:
    - Створення документації для користувачів та розробників.
    - Надання підтримки користувачам та виправлення помилок.
6. **Розгортання та підтримка виробництва**:
    - Розгортання додатку на локальному сервері.
    - Забезпечення підтримки та оновлень після розгортання.

4.2 Програмний код

Проєкт називається Project-Management-App-main, що має в собі такі папки та файли:

1. Dist – основа нашого додатку, його вигляд та запуск відбувається саме з цієї папки.
2. Src – код роботи нашого дотатку на Javascript, створений за допомогою бандлеру

Dist

Index.html

```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Project Management App</title>
    <meta name="author" content="Harmeet Matharoo">
    <link rel="preconnect" href="https://fonts.googleapis.com">
    <link rel="preconnect" href="https://fonts.gstatic.com" crossorigin>
    <link
href="https://fonts.googleapis.com/css2?family=Source+Sans+Pro:wght@300;400;700&
display=swap"
        rel="stylesheet">
    <link rel="icon" sizes="32x32" href='df79ab56babc9160b9cf.svg'>
</head>

<body>

    <div id="container">


        <div id="navContainer">
            <nav role="navigation">
                <div id="menuToggle">
                    <!--
                    hidden checkbox is used as click reciever,
                    so you can use the :checked selector on it.
                    -->
                    <input id="inputMenu" name="toggle" type="checkbox" />
                    <h1 class="task-list-title">Projects</h1>
                    <label id="labelMenu" for="toggle">
                        <span>menu</span>
                        <!--
                        divs to be styled as a hamburger button
                        -->
                        <div></div>
                        <div></div>
                        <div></div>

                    </label>
```

```
                <!--
                the menu that will slide in from the left
                -->

                <ul id="menu" data-projects>

                   <div id="projectsContainer">

                   </div>

                   <div id="addProjectForm">
                      <button id="addProjectButton"> <span>+</span>Create
Project</button>
                      <div class="form-popup" id="myForm">
                         <form data-new-project-form id="newProjectForm"
action="">
                            <label id="projectLabel" for="projectName">Project
Name</label><br>
                            <input data-new-project-input type="text"
id="projectInput" name="projectInput"><br>
                            <input id="submitBtn" type="submit"
value="Create"><br>
                         </form>
                      </div>
                   </div>
                </ul>
             </div>
          </nav>
       </div>

       <div id="taskList" class="todo-list">
          <div class="todo-header">
             <h2 id="projectTitle" class="list-title">Select a Project</h2>
          </div>

          <div class="todo-body" id="todoBody">
             <div id="tasksContainer" class="tasks">

             </div>
             <div id="createTaskContainer">
                <button id="addTaskButton"> <span>+</span>Create Task</button>

                <div id="myTasks" class="new-task-creator">
                   <form data-new-task-form id="newTaskForm" action="">
                      <label id="taskLabel" for="taskName">Task Name</label><br>
                      <input type="text" class="new-task" id="taskInput"
name="taskName"><br>
                      <div id="priorityDiv">
                         <label for="priority">Priority:</label>
                         <select name="priority" id="priorities">
                            <option value="high">High</option>
                            <option value="medium">Medium</option>
                            <option value="low">Low</option>
                         </select>
                         <br><br>
                      </div>
                      <div id="dueDateDiv">
                         <label for="dueDate">Due Date:</label>
                         <input type="date" name="dueDate" id="dueDate">
```

```
                        <br><br>
                    </div>
                    <input id="taskSubmitButton" class="new-task"
type="submit" value="Create"><br>

                </form>
            </div>

            <div class="delete-stuff">
                <!-- <button id="clearBtn" class="btn delete">Clear Completed
Tasks</button> -->
                <button id="deleteBtn" class="btn delete">Delete All
Tasks</button>
            </div>
        </div>


    </div>
</div>
<div id="gitLogoDivContainer">
    <a id="gitLogoDiv"></a>
</div>
</div>

<script defer src="bundle.js"></script>
</body>

</html>
```

Src

createProjects.js

```
import DOMInterface from './DOMInterface.js';
document.addEventListener('DOMContentLoaded', DOMInterface);
import projectImgIcon from './images/svg/list.svg';

function createProjects() {

    const project = (() => {

        //create new Project
        const createProject = (a) => {

            const project = document.createElement('li');
            project.classList.add('list-name');
            project.setAttribute('data-project', a);
            project.id = a;
            const projectImg = document.createElement('img');
            projectImg.classList.add('list-img');
            // projectImg.src = '../src/images/svg/list.svg';
            projectImg.src = projectImgIcon;
            projectImg.alt = 'list';
            const deleteProjectButton = document.createElement('button');
            deleteProjectButton.classList.add('deleteProjectBtn');
```

```
        deleteProjectButton.innerText = 'X';
        project.append(projectImg, a, deleteProjectButton);
        projectsContainer.append(project);
        addProject();

    };

    // add new project to local storage
    const addProject = () => {
        //if input is empty, don't add project
        if (projectInput.value === '') {
            return;
        } else {
            let newProject = projectInput.value;
            let newProjectObj = {
                projectName: newProject
            };
            let newProjectString = JSON.stringify(newProjectObj);
            localStorage.setItem(newProject, newProjectString);
            projectInput.value = '';
            loadProjects();
        }
    }

    //load projects from localStorage when page loads
    const loadProjects = () => {
        let projects = Object.keys(localStorage);
        // remove tasks key from projects variable
        function deleteByVal(val) {
            for (let key in projects) {
                if (projects[key] == val) delete projects[key];
            }
        }
        deleteByVal('tasks');

        projects.forEach(function(project) {
            //if project already exists, don't add it again
            if (document.getElementById(project)) {
                return;
            } else {
                createProject(project);
                tasksContainer.style.display = 'block';
            }
        });

    }

    return {
        addProject,
        loadProjects,
        createProject
    };

})();

//add event listener to create project when form is submitted
newProjectForm.addEventListener('submit', getProjectName);

//get project name from input and call createProject method
```

```
    function getProjectName(event) {
        event.preventDefault();
        let projectName = projectInput.value;
        // if project exists in local storage or project name is empty, don't add
it again
        if (projectName === ''|| localStorage.getItem(projectName)) {
            return;
        } else {
            project.createProject(projectName);
        }
        return projectName;
    }

    //load projects from localStorage when page loads
    project.loadProjects();
}

export default createProjects;
```

createTasks.js

```
import DOMInterface from './DOMInterface.js';
import { formatDistanceToNow} from 'date-fns'
import parseISO from 'date-fns/parseISO'

document.addEventListener('DOMContentLoaded', DOMInterface);

function createTasks() {

    // create an IIFe to create tasks

    const task = (() => {

        let taskArray = [];

        //create new task using taskInput and add it to the selected project and
add it to local storage
        const createNewTask = (projectName, taskName, priority, dueDate, checked)
=> {
            const task = document.createElement('div');
            task.classList.add('task');
            task.setAttribute('data-task', taskName);
            task.setAttribute('data-project-name', projectName);
            const taskInput = document.createElement('input');
            taskInput.setAttribute('type', 'checkbox');
            taskInput.setAttribute('id', taskName);
            taskInput.setAttribute('data-task', taskName);
            taskInput.classList.add('task-input');
            taskInput.checked = checked;

            //if task has a class delete-task dont add it to the taskArray
            const taskLabel = document.createElement('label');
            taskLabel.setAttribute('for', taskName);
            taskLabel.classList.add('task-label');
            taskLabel.setAttribute('data-content', taskName);
            let dueDateConverted = parseISO(dueDate);
```

```
            let dueDateFromNow = 'Due ' +
formatDistanceToNow(dueDateConverted,{addSuffix: true});
            taskLabel.innerText = taskName ;
            let taskLabelSpan = document.createElement('span');
            taskLabelSpan.classList.add('task-label-span');
            taskLabelSpan.innerText = ' ( ' + dueDateFromNow + ' )';
            taskLabel.appendChild(taskLabelSpan);

            task.append(taskInput, taskLabel);

            if(priority === 'high') {
               task.classList.add('high-priority');
            } else if(priority === 'medium') {
               task.classList.add('medium-priority');
            } else if(priority === 'low') {
               task.classList.add('low-priority');
            }

            tasksContainer.append(task);
            tasksContainer.style.display = 'block';

            //add task to local storage
            addTask(projectName, taskName, priority, dueDate, checked);
         }

         // add tasks local storage
         const addTask = (projectName, taskName, priority, dueDate, checked) => {

            let taskObj = {
               [projectName] : {taskName, priority, dueDate, checked}
            }
            //add task to selected project
            if(projectName) {
               taskArray.push(taskObj);
            } else {
               let taskArray = [];
               taskArray.push(taskObj);
            }

            localStorage.setItem('tasks', JSON.stringify(taskArray));
            taskInput.value = '';
            loadTasks();
         }

         //load tasks from local storage
         const loadTasks = () => {

            let tasks = localStorage.getItem('tasks');
            let tasksParsed = JSON.parse(tasks);

            let arrayOfProjectNames = [];
            let arrayOfTaskNames = [];
            let arrayOfPriorities = [];
            let arrayOfDueDates = [];
            let arrayOfChecked = [];

            //if there are no tasks in local storage return
            if(!tasksParsed) {
               return;
```

```
        } else {
            for (const [key, value] of Object.entries(tasksParsed)) {
                let taskIndex = key;
                let task = value;
                for (const [projectName, taskName] of Object.entries(task)) {

                    let nameOfProject = projectName;
                    arrayOfProjectNames.push(nameOfProject);

                    let allTaskNames = taskName;

                    let taskNames = Object.values(allTaskNames)[0];
                    let priority = Object.values(allTaskNames)[1];
                    let dueDate = Object.values(allTaskNames)[2];
                    let checked = Object.values(allTaskNames)[3];

                    arrayOfTaskNames.push(taskNames);
                    arrayOfPriorities.push(priority);
                    arrayOfDueDates.push(dueDate);
                    arrayOfChecked.push(checked);

                }
            }

            for (let i = 0; i < arrayOfProjectNames.length; i++) {
                //if task already exists, don't create it again
                if(document.querySelector(`[data-
task="${arrayOfTaskNames[i]}"]`)) {
                    return;
                } else {
                    createNewTask(arrayOfProjectNames[i], arrayOfTaskNames[i],
arrayOfPriorities[i], arrayOfDueDates[i], arrayOfChecked[i]);
                }
            }
        }
    }

    return {
        createNewTask,
        addTask,
        loadTasks,
    }

    })();


    //create new task using taskInput and add it to the selected project
    newTaskForm.addEventListener('submit', (e) => {
        e.preventDefault();
        let projectsName = document.querySelector('.active-
list').getAttribute('data-project');
        let tasksName = taskInput.value;
        let taskPriority = priorities.value;
        let dueDateValue = dueDate.value;
        let isTaskChecked = false;

        // if project exists in local storage or project name is empty, don't add
it again
        if (tasksName !== '') {
```

```
            // if project has a class of active-list then add task to that project
            if (document.querySelector('.active-list')) {
               task.createNewTask(projectsName, tasksName, taskPriority,
dueDateValue, isTaskChecked);
            }
         }

         taskInput.value = '';
      });

      task.loadTasks();

}

export default createTasks;
```

deleteProjects.js

```
import DOMInterface from './DOMInterface.js';
document.addEventListener('DOMContentLoaded', DOMInterface);


// create delete projects fxn

function deleteProject() {

    // create IIFE to delete project

    const deleteCurrentProject = (() => {

       //on pressing button delete project

       const deleteProject = () => {

       document.body.addEventListener( 'click', function ( event ) {
          if( event.target.className == 'deleteProjectBtn' ) {

             //delete project from DOM
             event.target.parentElement.remove();

             // delete project from local storage

localStorage.removeItem(event.target.parentElement.dataset.project);
             tasksContainer.style.display = 'none';
          };
       });

    }

    return {
       deleteProject,
    }

    })();

    deleteCurrentProject.deleteProject();
```

```
}

export default deleteProject;
```

deleteTasks.js

```
import DOMInterface from './DOMInterface.js';
document.addEventListener('DOMContentLoaded', DOMInterface);


function deleteTasks() {

    // create IIFE to delete project

    const deleteCurrentTask = (() => {

       //on pressing button delete project

       const deleteTask = () => {

       document.body.addEventListener( 'click', function ( event ) {
          if( event.target.id == 'deleteBtn' ) {

             //remove all tasks from DOM
             tasksContainer.innerHTML = '';

             // delete all tasks from local storage
             localStorage.removeItem('tasks');
          };
       });

    }

    return {
       deleteTask,
    }

    })();

    deleteCurrentTask.deleteTask();

}

export default deleteTasks;
```

DOMInterface.js

```
import gitImage from './images/imageee.png';
import tabIcon from './images/svg/list-tab.svg';
```

```
function DOMInterface() {

    //catching DOM
    let gitLogoDiv = document.getElementById('gitLogoDiv');

    let inputMenu = document.getElementById('inputMenu');

    let taskList = document.getElementById('taskList');

    //projects
    let projectSubmitButton = document.getElementById('submitBtn');
    let addProjectButton = document.getElementById('addProjectButton');
    let newProjectForm = document.getElementById('[newProjectForm]');
    let deleteProjectBtn = document.querySelectorAll('.deleteProjectBtn');
    let projectsContainer = document.getElementById('projectsContainer');
    let projectInput = document.getElementById('projectInput');



    //tasks
    let addTaskButton = document.getElementById('addTaskButton');
    let taskSubmitButton = document.getElementById('taskSubmitButton');
    let tasksContainer = document.getElementById('tasksContainer');
    let projectTitle = document.getElementById('projectTitle');
    let newTaskForm = document.getElementById('newTaskForm');
    let taskInput = document.getElementById('taskInput');
    let getPriority = document.getElementById('priotities');
    let dueDate = document.getElementById('dueDate');
    let getDate = new Date();
    getDate.setDate(getDate.getDate() + 1);
    dueDate.valueAsDate = getDate;
    dueDate.min = new Date().toISOString().split("T")[0];

    //delete tasks
    let deleteBtn = document.getElementById('deleteBtn');


    //git logo img
    let gitLogo = document.createElement('img');
    gitLogo.src = gitImage;
    gitLogo.classList.add('gitLogo');
    gitLogoDiv.appendChild(gitLogo);


    // DOM when page loads events
    window.addEventListener('load', () => {
       tasksContainer.style.display = 'none';
    });

    // DOM click events
    inputMenu.addEventListener('click', addMargin);
    inputMenu.addEventListener('click', hideH1);



    addTaskButton.addEventListener('click', addTask);
    addProjectButton.addEventListener('click', openForm);
    projectSubmitButton.addEventListener('click', closeForm);
    taskSubmitButton.addEventListener('click', closeTaskForm);
```

```
    //Functions for buttons and adding margin
    function openForm() {
       document.getElementById("myForm").style.display = "block";
       addProjectButton.style.display = "none";
    }

    function closeForm() {
       document.getElementById("myForm").style.display = "none";
       addProjectButton.style.display = "block";
    }


    function addMargin() {
       taskList.style.marginTop = null;
       taskList.classList.toggle('removeMarginAfterClick');
    }

    function hideH1() {
       let h1Ele = document.getElementsByTagName('h1')[0];
       h1Ele.classList.toggle('hideH1');

    }

    function addTask() {
       document.getElementById("myTasks").style.display = "block";
       addTaskButton.style.display = "none";
    }

    function closeTaskForm() {
       document.getElementById("myTasks").style.display = "none";
       addTaskButton.style.display = "block";
    }

}


export default DOMInterface;
```

index.js

```
import './style/styles.css';
import DOMInterface from './DOMInterface.js';
import createProjects from './createProjects.js';
import deleteProjects from './deleteProjects.js';
import renderProjects from './renderProjects.js';
import createTasks from './createTasks.js';
import deleteTasks from './deleteTasks.js';




document.addEventListener('DOMContentLoaded', DOMInterface);
document.addEventListener('DOMContentLoaded', createProjects);
document.addEventListener('DOMContentLoaded', deleteProjects);
```

```
document.addEventListener('DOMContentLoaded', renderProjects);
document.addEventListener('DOMContentLoaded', createTasks);
document.addEventListener('DOMContentLoaded', deleteTasks);
```

renderProjects.js

```
import DOMInterface from './DOMInterface.js';
import createProjects from './createProjects.js';

document.addEventListener('DOMContentLoaded', DOMInterface);
document.addEventListener('DOMContentLoaded', createProjects);

function renderProjects() {

    // create an IIFe to render selected project tasks

    const renderCurrentProject = (() => {

        // when a project is clicked add class to it and remove it from other
projects
        const selectedProject = () => {

            //use event delegation to select project
            document.addEventListener('click', (e) => {
                if(e.target.classList.contains('list-name')) {

                    let allProjects = document.querySelectorAll('.list-name');
                    for (let project of allProjects) {
                        if (project == e.target) {
                            project.classList.add('active-list');

                        } else {
                            project.classList.remove('active-list');
                        }

                        projectTitle.innerText = e.target.id;
                    }
                    showTasks();
                }
            });
        }


        //when i click on a show tasks that had data-project-name equal to the
project name
        const showTasks = () => {

            let project = document.querySelector('.active-
list').getAttribute('data-project');
            let tasks = document.querySelectorAll('.task');
            for (let task of tasks) {
```

```
            if (task.getAttribute('data-project-name') === project) {
                tasksContainer.style.display = 'block';
                task.style.display = 'flex';
            } else {
                task.style.display = 'none';
            }
        }
    }

    return {
        selectedProject,
        showTasks
    }

    })();

    renderCurrentProject.selectedProject();
}

export default renderProjects;
```

ВИСНОВКИ<з нової сторінки>

У випускній кваліфікаційній роботі представлено результати теоретичних і прикладних досліджень, спрямованих на розробку інформаційної технології для оцінювання показників соціально-економічного розвитку регіонів з метою підвищення ефективності управління регіональним розвитком. Результати досліджень стали основою для створення автоматизованої системи оцінювання показників соціально-економічного розвитку, яка була успішно впроваджена для оцінки Київської області.

В ході досліджень були отримані наступні висновки:

1. **Соціально-економічний моніторинг** виявився важливим засобом науково-практичного обґрунтування регіональних стратегій розвитку. Він надає органам регіонального управління повну, достовірну і оперативну інформацію про соціальні та економічні процеси в регіоні.

2. **Використання сучасних інформаційних технологій** є ключовим для забезпечення якісних даних для оцінювання. Використання актуальної та надійної моніторингової інформації дозволяє приймати адекватні управлінські рішення, відповідно до сучасної соціально-економічної ситуації.

3. **Розроблений метод визначення комплексного показника соціально-економічного розвитку** та інформаційно-логічна модель автоматизованої системи оцінювання рівня регіонального розвитку є цінними інструментами для оцінювання і аналізу регіонального розвитку.

4. **Автоматизована система оцінювання** була розроблена та успішно реалізована для оцінки показників соціально-економічного розвитку Київської області. Ця система дозволяє моделювати соціально-економічну ситуацію в регіоні та надає керівництву цінну інформацію для розвитку стратегій регіонального розвитку.

У підсумку, розроблена інформаційна технологія оцінювання показників соціально-економічного розвитку регіонів є інструментом, який сприяє ефективному управлінню регіональним розвитком та надає можливість приймати обґрунтовані рішення для поліпшення соціально-економічного стану регіону. Для подальшого розвитку цієї технології можна розглядати можливість розширення її застосування на інші регіони та врахування нових аспектів регіонального розвитку.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ<оформити відповідно до стандарту>

1. Schwalbe, K. (2020). Information Technology Project Management. Cengage Learning.
2. Phillips, J. (2020). PMP Project Management Professional Study Guide. Wiley.
3. Wysocki, R. K. (2014). Effective Project Management: Traditional, Agile, Extreme. Wiley.
4. Kerzner, H. (2017). Project Management: A Systems Approach to Planning, Scheduling, and Controlling. Wiley.
5. Marchewka, J. T. (2019). Information Technology Project Management. Wiley.
6. Gido, J., & Clements, J. P. (2018). Successful Project Management. Cengage Learning.
7. Schwalbe, K. (2019). An Introduction to Project Management. Cengage Learning.
8. Heagney, J. (2016). Fundamentals of Project Management. Amacom.
9. Project Management Institute. (2017). A Guide to the Project Management Body of Knowledge (PMBOK Guide). PMI.
10. Turner, R. (2014). Gower Handbook of Project Management. Routledge.
11. PRINCE2. (2017). Managing Successful Projects with PRINCE2. TSO.
12. Agile Alliance. (2019). Agile Project Management: Creating Innovative Products. Agile Alliance.
13. Crawford, L., & Pollack, J. (2014). Business Analysis and Leadership: Influencing Change. Routledge.
14. Thiry, M. (2016). Program Management. Routledge.

15. Schlichter, J., & Scott, A. (2017). Web Application Development with MEAN. Manning Publications.
16. Freeman, E., Robson, E., & Bates, S. (2018). Head First Design Patterns. O'Reilly Media.
17. Flanagan, S. (2015). JavaScript: The Definitive Guide. O'Reilly Media.
18. McFarland, D. (2018). JavaScript & jQuery: The Missing Manual. O'Reilly Media.
19. Zeldman, J. (2017). Designing with Web Standards. New Riders.
20. Duckett, J. (2018). HTML and CSS: Design and Build Websites. Wiley.