Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Інформаційно-аналітичних технологій та менеджменту
(повна назва)
Кафедра Інформатики
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти перший (бакалаврський)

РОЗРОБКА ЗАСТОСУКУ «МЕНЕДЖЕР ЗАДАЧ»
(тема)

Виконав:
студент 4 курсу, групи ІТІНФ-20-1

Санін В.О.
(прізвище, ініціали)

Спеціальності 122 Комп'ютерні науки
(код і повна назва спеціальності)

Тип програми освітньо-професійна

Освітня програма Інформатика
(повна назва освітньої програми)

Керівник ст. викл. Путятіна О.Є.
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри _____ Кобилін О.А.
(підпис) (прізвище, ініціали)

2024 р.

Харківський національний університет радіоелектроніки

Факультет Інформаційно-аналітичних технологій та менеджменту
(повна назва)

Кафедра Інформатики
(повна назва)

Рівень вищої освіти перший (бакалаврський)

Спеціальність 122 Комп'ютерні науки
(код і повна назва)

Тип програми освітньо-професійна

Освітня програма Інформатика
(повна назва освітньої програми)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«____»_____ 2024 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові Саніну Владиславу Олександровичу
(прізвище, ім'я, по батькові)

1. Тема роботи Розробка застосунку «Менеджер задач»

затверджена наказом університету від 20 травня 2024 року № 464 Ст

2. Термін подання студентом роботи до екзаменаційної комісії 28 травня 2024 р.

3. Вихідні дані до роботи науково-методична та науково-технічна література, матеріали конференцій, дані інтернет-мережі, фреймворк React, фреймворк Flask, мова програмування JavaScript, мова програмування Python, середовище розробки Microsoft Visual Studio Code.

4. Перелік питань, що потрібно опрацювати в роботі
1. Огляд існуючих застосунків менеджерів задач з можливістю запису фінансів.
2. Моделювання бази даних для вебсайту.
3. Проєктування архітектури застосунку.
4. Розробка алгоритму запису подій та фінансів.
5. Розробка календаря для планування часу.

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (п.5 включається до завдання за рішенням випускової кафедри) Актуальність проблеми в ефективному управлінні своїм часом та фінансами, постановка задачі, схема бази даних, відображення календаря в застосунку, головна сторінка застосунку, розділ підрахунку статистики, аналіз роботи алгоритму запису подій.

6. Консультанти розділів роботи (п.6 включається до завдання за наявності консультантів згідно з наказом, зазначеним у п.1)

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Терміни виконання етапів роботи	Примітка
1	Отримання завдання на кваліфікаційну роботу	08.04.2024	
2	Аналіз завдання, підбір літератури	08.04.24-17.04.24	
3	Аналіз літератури з досліджуваної проблеми	18.04.24-18.04.24	
4	Аналіз технічних засобів	19.04.24-22.04.24	
5	Моделювання структури застосунку	22.04.24-24.04.24	
6	Програмна реалізація	25.04.24-23.05.24	
7	Оформлення пояснювальної записки	24.05.24-26.05.24	
8	Перевірка на плагіат	29.05.24	
9	Рецензування	30.05.24	
10	Підготовка презентації та доповіді	30.05.24-02.06.24	
11	Занесення роботи в електронний архів	03.06.24	
12	Попередній захист кваліфікаційної роботи	10.06.24	

Дата видачі завдання 8 квітня 2024 р.

Студент _____
(підпис)

Керівник роботи _____ ст. викл. Путятіна О.Є.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ/ABSTRACT

Пояснювальна записка до кваліфікаційної роботи: 56 с., 1 табл., 14 рис., 5 дод., 30 джерел.

ЗАСТОСУНОК, БАЗА ДАНИХ, JAVASCRIPT, REACT, PYTHON, FLASK, REST API, SPRING BOOT, ANT DESIGN, SQLITE, ПРОФІЛЬ КОРИСТУВАЧА, ФІНАНСОВЕ ПЛАНУВАННЯ.

Об’єктом роботи є можливість створення та відстеження задач протягом місяця та відстеження фінансового планування в одному місці.

Метою роботи є розробка застосунку, який забезпечить користувачам зручний та ефективний інструмент для керування їхніми фінансами та задачами.

Проведено дослідження існуючих успішних застосунків, описано їх переваги та недоліки для користувачів. Досліджено технології, що використовуються при створенні певних застосунків. Розроблено алгоритми взаємодії користувачів із контентом і між собою, збереження інформації. Реалізовано користувацький інтерфейс для взаємодії із застосунком.

У результаті роботи здійснена програмна реалізація застосунку для відстеження задач та фінансового планування.

APPLICATION, DATABASE, JAVASCRIPT, REACT, PYTHON, FLASK, REST API, SPRING BOOT, ANT DESIGN, SQLITE, USER PROFILE, FINANCIAL PLANNING.

The object of the work is the ability to create and track tasks throughout the month and track financial planning in one place.

The purpose of the work is to develop an application that will provide users with a convenient and effective tool for managing their finances and tasks.

A study of existing successful applications was conducted, their advantages and disadvantages for users were described. The technologies used in the creation of certain applications are studied. Algorithms for user interaction with content and with each other, and for storing information are developed. The user interface for interaction with the application is implemented.

As a result of the work, the software implementation of the application for task tracking and financial planning was carried out.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

JS – JavaScript (мова програмування JavaScript)

JPA – Java Persistence API

JPQL – Java Persistence Query Language (мова запитів JPA)

SQL – Structured Query Language (мова структурованих запитів)

REST API – Representational State Transfer Application Programming Interface (інтерфейс програмування застосунків з представленням стану)

СУБД – система управління базами даних

БД – база даних

CSS – Cascading Style Sheets (каскадні таблиці стилів)

UI – User Interface (інтерфейс користувача)

PY – Python (мова програмування Python)

Flask – мікрофреймворк для вебзастосунків

ВСТУП

Сучасний ритм життя людини стає все швидшим, а потреби в ефективному управлінні своїм часом та фінансами зростають. Ефективне управління своїм часом та фінансами стає викликом для багатьох людей. Постійний стрес, безлад, навантаження та різноманітність завдань можуть призвести до втрати контролю над своїми певними задачами та фінансами.

Застосунок «Менеджер задач» це рішення сучасних проблем суспільства. Він призначений для надання користувачам ефективного та зручного інструменту для організації планів, завдань та фінансів. Цей застосунок допоможе керувати своїм розкладом протягом місяця, відстежувати свою ефективність протягом дня, вести фінансову аналітику та нагадувати про важливі події.

Актуальність роботи полягає в створенні та наданні суспільству застосунку, який повністю відповідає реальним потребам. Цей застосунок надає можливість користувачам зберігати контроль над своїм життям та завданнями, підвищує продуктивність та сприяє досягненню фінансової стабільності.

1 ТЕОРЕТИЧНИЙ ОГЛЯД ФУНКЦІОНАЛУ ТА МЕТОДІВ СТВОРЕННЯ ЗАСТОСУНКУ

1.1 Огляд предметної області

Менеджер задач – це застосунок, який об’єднує в собі можливості організації завдань та планування фінансів для користувачів. У контексті даної роботи, цей застосунок – інноваційний інструмент, що дозволяє ефективно керувати часом, планувати певні задачі протягом місяця, ввести статистику виконаних завдань. Основні аспекти цієї роботи є такими:

– функціональність, що забезпечує користувачам можливість створювати, організовувати та відстежувати завдання в їхній діяльності;

– інтеграція із засобами фінансового обліку для забезпечення відстеження витрат, планування бюджету та аналізу фінансового стану;

– забезпечення зручного та інтуїтивно зрозумілого інтерфейсу для користувачів, щоб вони могли легко використовувати застосунок без особливих навичок або навчання.

Основна мета розробки такого застосунку – це надати користувачам зручний інструмент для керування їхніми завданнями та фінансами в єдиному середовищі, що дозволить їм ефективно керувати своїм часом та грошима. Це є актуальною темою, оскільки люди постійно шукають способи оптимізації свого робочого процесу та фінансової діяльності.

1.2 Огляд фінкцій популярних менеджерів задач

Одними з найбільш популярних менеджерів задач, які зараз використовує суспільство, є TickTick, Todolist, Structured та Google Tasks. Ці застосунки надають користувачам зручний інструмент для керування

завданнями та плануванням. Вони дозволяють створювати, організовувати та відстежувати завдання, надавати їм пріоритети, дедлайни та теги. Застосунки також надають можливість структурування завдань в проекти та списки, а також інтеграцію з календарем та іншими сервісами. Вони мають різні особливості, але спільна мета – забезпечити ефективне управління часом та завданнями для користувачів.

Одним з недоліків цих застосунків є відсутність вести фінансову статистику доходів та витрат. Це може ускладнювати повний контроль над фінансовими потоками користувача, особливо в контексті їх інтеграції з менеджером задач. Відсутність фінансового планування в таких застосунках може змушувати користувачів використовувати застосунків інструменти або сервіси для цієї мети, що може призвести до розірвання їхнього робочого процесу та збільшення часу, витраченого на керування фінансами. Тому важливо розглянути можливість інтеграції фінансового планувальника в такі застосунки для забезпечення більшої функціональності та зручності для користувачів.

1.3 Аналіз складових наведених менеджерів задач

Застосунок менеджер задач, як й інші застосунки, буде складатися із серверної частини (Backend), клієнтської частини (Frontend) та бази даних.

Клієнтська частина (Frontend) є інтерфейсом користувача, який взаємодіє з застосунком. Вона відповідає за відображення даних користувачеві та обробку дій користувача [1–5]. Для розробки клієнтської частини застосунку «Менеджер задач» використовується комбінація технологій, таких як JavaScript (JS) та бібліотека React. React – це потужна бібліотека для розробки інтерфейсів користувача, яка дозволяє створювати складні та динамічні сайти з високою швидкодією. Також для стилізації та організації компонентів інтерфейсу використовується бібліотека Ant Design (Antd), яка

надає готові компоненти та стилі для швидкої розробки зручного та сучасного дизайну. Комбінація JS, React та Antd дозволяє створити зручний та естетичний інтерфейс для користувачів, який буде легко взаємодіяти з функціоналом застосунку «Менеджер задач».

Серверна частина (Backend) є програмним забезпеченням, яке обробляє запити від клієнтської частини, виконує бізнес-логіку, взаємодіє з базою даних і відправляє дані назад клієнту. Для розробки серверної частини застосунку «Менеджер задач» використовується мова програмування Python та фреймворк Flask [6–10]. Flask є легким та ефективним фреймворком для побудови вебзастосунків на Python. Використання Python та Flask дозволяє забезпечити високу швидкодію та надійність серверної частини, а також забезпечити безпеку та функціональність застосунку «Менеджер задач».

База даних є сховищем даних, що використовуються застосунком для зберігання інформації про користувачів, їх дії, вміст і т.д. Вона відіграє ключову роль у збереженні та забезпеченні доступності даних програми. Існує безліч різних типів баз даних, включаючи реляційні, SQLite3 та документоорієнтовані.

1.4 Огляд технологій для розробки менеджер задач

Розробка застосунку менеджер задач вимагає використання різноманітних технологій та інструментів, щоб забезпечити їхню ефективність, масштабованість та зручність для користувачів.

Для розробки серверної частини застосунку гарним рішенням буде використати мову програмування Python та фреймворк Flask.

Клієнтська частина може бути реалізована за допомогою мови програмування JavaScript і бібліотеки React та Antd.

У поєднанні ці технології допомагають створювати потужні та зручні застосунки, забезпечуючи надійну та швидку роботу, інтуїтивно зрозумілий

інтерфейс та багатий функціонал для користувачів, що дуже важливо для даного застосунку.

1.4.1 Огляд Python та фреймворка Flask

Мова програмування Python є однією з найбільш популярних мов програмування у світі. Вона славиться своєю простотою та лаконічністю, що робить її ідеальним вибором для широкого спектру завдань, від простих сценаріїв до складних програмних проєктів [11–16]. Деякі переваги Python включають:

– простота вивчення та читання коду. Python має чистий та лаконічний синтаксис, що дозволяє розробникам швидко засвоювати мову та створювати зрозумілий код;

– велика спільнота розробників. Python має активну та дружню спільноту, яка підтримує безліч корисних бібліотек, фреймворків та інструментів розробки;

– кросплатформенність. Програми, написані на Python, можуть працювати на різних операційних системах, що робить їх універсальними та легко переносимими;

– розширюваність. Python має велику кількість сторонніх бібліотек та модулів, що дозволяє розробникам розширювати функціональність своїх програм шляхом використання готових рішень.

Flask – це легкий та ефективний фреймворк для розробки застосунків на мові програмування Python. Він відомий своєю простотою та гнучкістю, що робить його відмінним вибором для швидкого створення застосунків. Деякі переваги Flask включають:

– простота використання. Flask має простий та інтуїтивно зрозумілий інтерфейс, що дозволяє розробникам швидко створювати застосунків без зайвих складнощів;

– гнучкість та розширюваність. Flask надає розробникам велику свободу вибору та гнучкість у реалізації власних ідей та функціональності;

– велика спільнота та підтримка. Flask має активну спільноту розробників, яка надає безліч корисних розширень та плагінів для розширення можливостей фреймворку.

Ці технології разом надають розробникам зручні та потужні інструменти для створення застосунків на Python.

1.4.2 Огляд JavaScript і бібліотеки React

JavaScript – це мова програмування, яка використовується для створення динамічних та інтерактивних сайтів. Вона володіє широким функціоналом, включаючи можливість маніпулювати вмістом сторінки, взаємодіяти з користувачем, валідувати дані та виконувати асинхронні запити до сервера без перезавантаження сторінки.

JavaScript є незамінною складовою для створення сучасних вебзастосунків, і однією із її найкращих бібліотек є React. Вона відома своєю простотою та ефективністю у роботі з компонентами, що дозволяє розробникам легко створювати складні інтерфейси. React використовує концепцію віртуальної об’єктної моделі документа (DOM) для оптимізації швидкодії та реактивної оновлення сторінок без перезавантаження. Вона також дозволяє використовувати JSX – розширення JavaScript, яке дозволяє описувати структуру компонентів у вигляді HTML коду. React має багато переваг, таких як ефективність, гнучкість і продуктивність, має активну спільноту розробників.

1.4.3 Огляд СУБД SQLite

SQLite – це легка, вбудовувана система управління базами даних (СУБД), яка забезпечує ефективне зберігання та обробку даних без необхідності встановлення окремого серверу баз даних [17–21]. Вона легка у використанні, не потребує складної конфігурації, ідеально підходить для вбудованого використання в різних типах програм, забезпечує переносність та високу продуктивність, а також підтримує багато корисних функцій, що робить її популярним вибором для різноманітних розробок.

1.5 Постановка задачі

Таким чином, актуальним завданням є розробка та реалізація застосунку менеджер задач з плануванням фінансів. Застосунок повинен забезпечувати можливість створення та редагування завданнями протягом місяця, надавати статистику виконання завдань та інформацію щодо доходів та витрат фінансів. Важливим аспектами будуть стабільність, продуктивність та простота використання системи.

Метою роботи є розробка застосунку, що дає високу продуктивність, стабільність та простоту у використанні, з метою задоволення потреб користувачів.

Для досягнення мети необхідно вирішити такі завдання:

– провести аналіз основних вимог та потреб користувачів;

– розробка та реалізація програмного коду для основних модулів застосунку, такі як календар задач, обробка виконаних задач та статистика фінансів.

2 ВИБІР ТЕХНОЛОГІЙ ТА ПРИНЦИПІВ ПРОЄКТУВАННЯ

2.1 Перелік функціоналу застосунку менеджер задач

Вебзастосунок, як і його аналоги, складається з клієнтської та серверної частини, а також використовується база даних. Все це дозволить реалізувати головні функції застосунку.

– реєстрація та авторизація користувача;
– запис та редагування задач;
– позначення запланованих днів у поточному місяці;
– запис та редагування витрат;
– запис та редагування доходів;
– слідкування за статистикою коштів за місяць;
– редагування та додавання нових категорій витрат та доходів.

2.2 Структура менеджера задач у вигляді мікросервісів

Вебзастосунок менеджер задач складається із трьох головних компонентів – база даних, серверна та клієнтська частина, що є мікросервісами. Такий підхід до розробки програми дає можливість розбиття на малі програми, автономні компоненти, який функціонує як окремий сервіс. Головні переваги використання мікросервісів:

– мікросервіси дозволяють легко масштабувати окремі компоненти системи незалежно один від одного, що полегшує адаптацію до змін обсягу навантаження і вимог користувачів;

– кожен мікросервіс може функціонувати і розвиватися незалежно, що дозволяє різним командам працювати над різними сервісами паралельно без взаємних перешкод;

– помилки в одному сервісі зазвичай не впливають на інші, що дозволяє ізолювати проблеми та полегшує виявлення, виправлення і підтримку;

– мікросервіси дозволяють використовувати різні технології та мови програмування для різних частин системи відповідно до їх потреб і вимог;

– мікросервіси дозволяють швидше впроваджувати нові функції або внесення змін без необхідності повного перезапуску всієї системи;

– розділення системи на мікросервіси спрощує керування, моніторинг і оновленнями кожного компонента окремо [22–24].

Сервер обробляє запити від клієнтів та взаємодіє з базою даних для отримання, зміни та збереження даних. Він надає API для взаємодії з клієнтською частиною програми, обробляє бізнес-логіку та керує безпекою та автентифікацією. У цій схемі сервер і клієнт можуть розглядатися як окремі мікросервіси, оскільки кожен з них може функціонувати незалежно і бути масштабованим незалежно від іншого. Клієнт – це програма, за допомогою якої користувач взаємодіє з застосунком. Клієнт надсилає запити на сервер для отримання даних або виконання певних дій, а потім відображає ці дані користувачеві у зручному форматі.

2.3 Структура база даних

2.3.1 Сутності та атрибути

В базі даних зберігається інформація про користувача, запис всіх подій, маніпуляції з грошима тощо. Для коректної роботи бази даних необхідно визначити, які сутності використовуються в даній роботі. Сутність – це об’єкт, який є одиницею даних, що має унікальну ідентифікацію та зберігається в базі даних. Сутності в базі даних перелічені у таблиці 2.1.

Таблиця 2.1 – Перелік сутностей бази даних менеджера задач

Назва сутності	**Пояснення**	**Опис сутності**
Account	Акаунт	Інформація про сторінку користувача. Дає можливість мати свій баланс
Consumption	Споживання	Запис всіх зроблених маніпуляцій доходів та витрат, які робить користувач
Consumption Types	Групи споживання	Типи можливих витрат та доходів
Events	Записи	Запис всіх завдань, які робить користувач
Sqlite Sequence	Статистика споживання	Статистика кількості повної інформації бази даних, кількість кожного типу (споживання, груп споживання, користувачів, записів користувача)
User	Користувач	Користувач застосунку

Користувач має роль, може створювати додаткові типи витрат та доходів, записувати суми витрат та доходів, переглядати статистику, робити та переглядати записи на кожен день. Для забезпечення коректного збереження даних і загалом роботи програми потрібно розробити правильну структуру таблиць.

На рисунку 2.1 зображений опис структури таблиць.

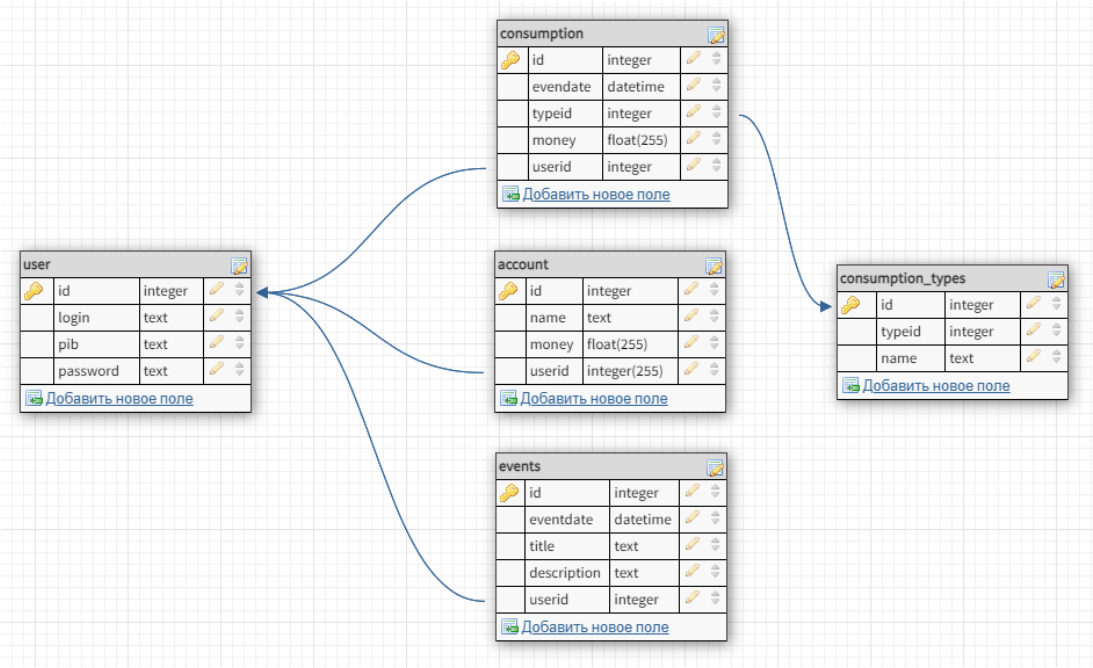

Рисунок 2.1 – Опис структури таблиць

2.3.2 Опис стовпців

Щоб користувач міг повністю використовувати застосунок у своєму житті, спочатку необхідно забезпечити можливість реєстрації та входу до свого облікового запису. Для авторизації використовуються логін та пароль, які будуть збережені у таблиці user в полях login та password. Кожен створений обліковий запис матиме унікальний ідентифікатор id.

Таблиця account зберігає інформацію про користувачів. В стопці name знаходіться гаманець кожного користувача, який прив’язаний за домопогою userid до зареєстрованих користувачів.

Після авторизації в застосунку користувач потрапляє на головну сторінку, де відображається інформація про загальний баланс. Сума коштів зберігається в полі money таблиці account. Таблиця account пов’язана з таблицею consumption, де зберігається вся інформація щодо витрат та доходів. Кожен запис у таблиці consumption має свій унікальний id та дату запису, яка

зберігається у полі eventsdate. Поле money відображає суму грошей, яка була витрачена або додана на певну дату.

Для коректного розрахунку балансу використовується додаткова таблиця consumptiontypes, яка зберігає типи надходжень або витрат коштів. Поле typeid визначає тип операції: 1 – дохід, 2 – витрата, що відповідно додає або віднімає суму з загального балансу користувача. Кожен typeid має свій унікальний id.

Користувач може записувати свої події, вибираючи певну дату в календарі. Ця інформація зберігається в таблиці events. Кожен запис у цій таблиці повинен мати певну структуру: eventsdate – дата, на яку був зроблений запис; title – заголовок задачі; description – короткий опис задачі, яку потрібно виконати.

Таблиця sqlite_sequence зберігає загальну інформацію про весь застосунок. Поле name містить назви таблиць, які використовуються в базі даних (consumption, consumptiontypes, account, events), а поле seq – загальну кількість записів у кожній таблиці.

Ця структура забезпечує зручне управління фінансами та подіями користувача, надаючи можливість реєстрації, авторизації, відстеження балансу, а також ведення обліку витрат, доходів та подій.

2.3.3 Інструмент створення бази даних

Всі таблиці та сама база даних буде створюватись за допомогою інструменту BD browser for SQLite.

BD Browser for SQLite – це інструмент для управління базами даних SQLite, який дозволяє користувачам легко переглядати, редагувати та виконувати запити до бази даних [25,26]. Цей браузер надає графічний інтерфейс, що полегшує роботу з базами даних, дозволяючи виконувати такі дії:

– можливість переглядати структуру бази даних, включаючи таблиці, стовпці та дані, що містяться в них;

– інструменти для додавання, видалення та оновлення записів у таблицях;

– інтерфейс для написання та виконання SQL запитів, з можливістю перегляду результатів запитів у зручному форматі;

– створення, видалення та модифікація таблиць та індексів;

– функції для експорту даних у різні формати (наприклад, CSV) та імпорту даних з інших джерел.

Цей інструмент є корисним для розробників, аналітиків та адміністраторів баз даних, які працюють з SQLite, оскільки він значно спрощує процес управління та аналізу баз даних.

2.4 Моделювання серверної частини

Серверна частина (Backend) – це частина програмного забезпечення, яка працює на сервері і відповідає за логіку, обробку даних, управління базами даних та взаємодію з клієнтською частиною (Frontend). Серверна частина забезпечує виконання таких основних функцій:

– приймає запити від клієнтів (наприклад, браузерів або мобільних застосунків) та відправляє відповіді з необхідними даними;

– виконує складні операції та обчислення, реалізуючи правила і логіку роботи програми;

– зберігає, змінює, видаляє і отримує дані з бази даних. Забезпечує доступність та цілісність даних;

– відповідає за авторизацію та автентифікацію користувачів, забезпечення безпеки даних та захист від несанкціонованого доступу;

– здійснює взаємодію з іншими сервісами та API, що можуть бути як внутрішніми, так і зовнішніми.

Прикладом серверної частини може бути вебсервер, що написаний на мовах програмування, таких як Python (Django, Flask), JavaScript (Node.js), Java (Spring), PHP (Laravel), Ruby (Ruby on Rails) тощо. Серверна частина тісно взаємодіє з клієнтською частиною, яка відповідає за відображення даних і взаємодію з користувачем.

2.4.1 Вибір технологій для розробки сервера

Flask – це мікрофреймворк для розробки на мові програмування Python [27,28]. Він відомий своєю легкістю, гнучкістю та простотою використання, що робить його популярним серед розробників для створення вебзастосунків та API. Основні характеристики Flask включають:

– надає базові можливості для розробки вебзастосунків, дозволяючи розробникам додавати тільки ті компоненти та функції, які їм дійсно потрібні;

– завдяки своєму модульному дизайну, Flask дозволяє розробникам вибирати додаткові бібліотеки та розширення відповідно до потреб проекту;

– має простий і зрозумілий API, що полегшує процес розробки та навчання;

– підтримує різноманітні розширення, які додають функціонал, такий як управління базами даних, автентифікація, валідація форм та інше;

– включає вбудований сервер для тестування і налагодження застосунків під час розробки.

Завдяки цим характеристикам, Flask є відмінним вибором для невеликих та середніх проектів, прототипів і застосунків, які потребують високої гнучкості.

2.4.2 Компоненти серверу

У складі архітектури серверу важливу роль відіграють різноманітні компоненти, які співпрацюють для ефективної роботи системи. Зокрема, класи сутностей, репозиторії, сервіси, контролери та інші допоміжні елементи створюють основу для функціонування застосунку.

Сутності, або об’єкти системи, є ключовими складовими, оскільки вони представляють дані, що зберігаються в базі даних. Кожна сутність має свою відповідну таблицю у базі даних та набір атрибутів, що описують її властивості. Ці сутності забезпечують цілісність даних і надають можливість взаємодії з базою даних. У конкретному випадку, серед сутностей, які складають архітектуру даного застосунку, можна відзначити такі: Account (таблиця account), Consumption (таблиця consumption), Consumption Types (таблиця consumption_types), Events (таблиця events) та Sqlite Sequence (таблиця sqlite_sequence).

Репозиторії виконують роль посередника між даними в базі даних та іншими частинами системи. Вони надають єдиний інтерфейс доступу до даних, що дозволяє інкапсулювати деталі взаємодії з базою даних, такі як створення SQL запитів або використання ORM.

Сервіси відповідають за управління бізнес-логікою та розділенням відповідальностей у системі. Вони обробляють запити, які надходять від контролерів, виконують необхідні операції з даними та повертають результати. Крім того, сервіси можуть виконувати валідацію даних та забезпечувати безпеку системи.

Усі ці компоненти разом утворюють добре організовану та ефективну архітектуру, яка дозволяє розробникам легко управляти та розвивати серверну частину застосунку.

2.4.3 Функціональність серверної частини

Серверна частина вебзастосунку виконує ключові функції, необхідні для управління фінансовими даними користувачів. Головні функції для реалізації роботи застосунку це редагування витрат, редагування доходів і відображення статистики коштів за місяць. Ці функції забезпечують інтерактивність застосунку та дозволяють користувачам ефективно контролювати свої фінанси.

Редагування витрат – важлива функція, яка дозволяє користувачам змінювати інформацію про свої витрати, включаючи суму, дату та категорію витрат.

$$NV = OV + \Delta V, \tag{2.1}$$

де NV – нові витрати;

OV – старі витрати;

ΔV – зміна витрат.

Редагування доходів означає внесення змін у вже існуючі дані про доходи. Це може включати додавання нових даних про доходи, виправлення помилок у вже існуючих записах про доходи або видалення неактуальних даних. Такі дії можуть виконуватися в адміністративному інтерфейсі програми або через спеціальні форми або поля в програмі.

$$\Delta D = D_{\text{нові}} - D_{\text{старі}}\ , \tag{2.2}$$

де ΔD – зміна доходів;

$D_{\text{нові}}$ – нові доходи;

$D_{\text{старі}}$– старі доходи.

Відображення статистики коштів за місяць означає представлення інформації про всі витрати та доходи за певний місяць у зручному форматі, який дозволяє користувачам отримати уявлення про їх фінансовий стан за цей

період. Це може включати сумарну інформацію про всі витрати та доходи за місяць, графіки чи діаграми, які візуалізують розподіл коштів за категоріями, а також можливість фільтрації за певними параметрами (наприклад, вид доходу або вид витрати). В цілому, це дає користувачам зручний спосіб відстежувати та аналізувати свої фінанси за місяць.

$$\sum \text{витрати} = \sum_{i=1}^{n} B_i, \tag{2.3}$$

$$\sum \text{доходи} = \sum_{i=1}^{n} D_i, \tag{2.4}$$

$$\text{Баланс} = \sum \text{доходи} - \sum \text{витрати}. \tag{2.5}$$

де B_i – витрати за певний день i;

D_i – доходи за конкретний день;

n – кількість днів у місяці.

Ці формули дозволяють обчислити загальну суму витрат, загальну суму доходів та загальний баланс за певний місяць на основі введених даних про витрати та доходи за кожен день місяця.

2.5 Клієнтська частина

2.5.1 Вибір технологій для створення клієнтської частини

Клієнтська частина відповідає за відображення інтерфейсу користувача та взаємодію з сервером. Для її реалізації обрано бібліотеку React, яка відома своєю простотою та ефективністю у створенні динамічних вебзастосунків. React дозволяє швидко розробляти інтерактивні інтерфейси, використовуючи компонентний підхід.

Також використовується бібліотека Antd, яка містить готові компоненти для швидкого створення стильних та функціональних інтерфейсів. Antd надає

широкий спектр готових елементів і дозволяє легко налаштовувати їх під конкретні потреби проєкту.

Таке поєднання дозволить швидко розробити сучасний та зручний інтерфейс для користувачів, забезпечуючи при цьому швидкість роботи та ефективність застосунку.

2.5.2 Структура вебсайту менеджера задач

Під час аналізу інших програм для управління завданнями було виявлено оптимальну структуру та дизайн вебсторінок, які забезпечують найкращий користувацький досвід. На основі цього були розроблені макети вебсторінок (рис. 2.2).

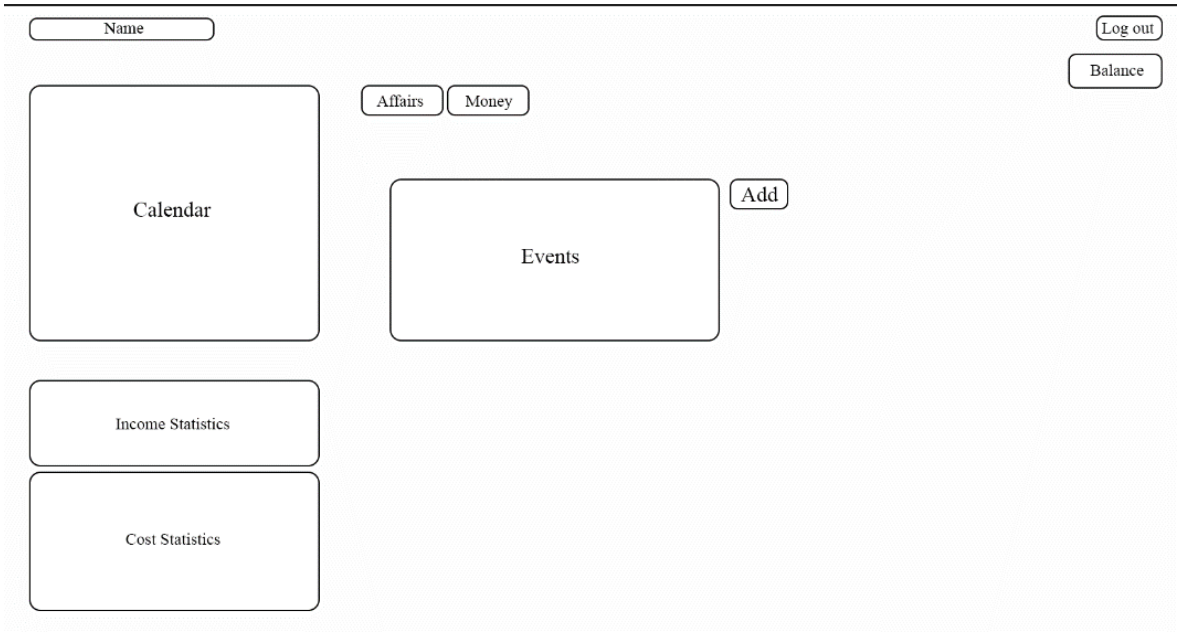

Рисунок 2.2 – Макет головної сторінки застосунку

2.5.3 Розробка інтерфейсу з допомогою React

React – це JavaScript бібліотека для створення інтерфейсів користувача, розроблена компанією Facebook. Вона дозволяє розробникам створювати динамічні та ефективні застосунки за допомогою компонентного підходу [29]. Однією з ключових особливостей React є використання віртуального DOM, що дозволяє оптимізувати роботу з реальним DOM і підвищує продуктивність застосунків. React також використовує JSX розширення синтаксису JavaScript, яке спрощує створення компонентів і розуміння структури коду. Бібліотека має широку підтримку та активну спільноту розробників, що дозволяє швидко знаходити рішення на форумах та розвивати навички роботи з React.

Ant Design (Antd) – це набір готових компонентів для створення інтерфейсу користувача вебзастосунків. Розроблений компанією Alibaba, Antd базується на React і пропонує великий вибір готових елементів, таких як кнопки, форми, таблиці, меню та інші [30]. Однією з ключових особливостей Antd є його вбудована підтримка дизайну Ant Design, яка дозволяє створювати зручний та сучасний інтерфейс користувача. Бібліотека також має гнучку систему налаштувань та можливість легкої інтеграції з іншими бібліотеками та інструментами. Ant Design активно розвивається та підтримується спільнотою розробників, що дозволяє швидко отримувати оновлення та рішення для проблем.

Для застосунку менеджер задач створюється головна сторінка записів, де відображається про події які записані в календарі, та кожен заповнений день виділяється певним кольором, статистика витрат та доходів за певний місяць та головне поле для заповнення певної інформації.

3 РЕАЛІЗАЦІЯ ВЕБЗАСТОСУНКУ МЕНЕДЖЕРА ЗАДАЧ

3.1 Середовища розробки

Для зручного створення вебзастосунку менеджера завдань було вирішено використовувати середовище розробки VS Code. У цьому редакторі було розроблено як клієнтську, так і серверну частини програми. VS Code був обраний завдяки своїй зручності та багатофункціональності, що дозволило забезпечити продуктивну робочу обстановку для розробки обох компонентів застосунку.

3.1.1 VS Code

VS Code – це інтегроване середовище розробки, яке надає широкий набір можливостей для роботи зі сховищами коду, підтримує розширення для різних мов програмування та забезпечує зручний інтерфейс для розробки різноманітних програмних проєктів. Розроблений компанією Microsoft, VS Code став популярним інструментом серед програмістів завдяки своїй ефективності та швидкості.

Відмінності VS Code полягають у його високій продуктивності, широкому спектрі можливостей та відкритій архітектурі, яка дозволяє розробникам створювати власні розширення та плагіни для підтримки різних технологій та функцій.

Цей редактор надає можливості для комфортної роботи з кодом, включаючи автодоповнення, підсвічування синтаксису, вбудовану систему контролю версій і багато інших корисних функцій, які сприяють зручній та ефективній розробці програмного забезпечення.

Інтегровані інструменти для розробки дозволяють розробникам створювати, тестувати та налагоджувати вебзастосунки безпосередньо в

середовищі VS Code, що робить його популярним вибором для розробників усього світу.

3.2 Реалізація вебзастосунку

В застосунку VS Code створено проєкт Менеджер задач із усіма необхідними залежностями.

Створено було 8 сутностей: Account, ConsumptionType, Consumption, Event, MonthData, Statistic, User, UserPro.

В класах були створенні інтерфейси маперів для забезпечення можливості перетворення об’єктів цих класів на формат JSON. Методи to_json() в класах моделей дозволяють забезпечити універсальний спосіб представлення даних у форматі, який легко зберігати, передавати та обробляти в різних частинах програми або між різними системами. Приклад одного з інтерфейсів маперів наведено в лістнигу 3.1.

Лістинг 3.1 Інтерфейс маперу класу Event:

```
def to_json(self):
    out_json = {}
    out_json["id"] = self.id
    out_json["eventdate"] = self.eventdate
    out_json["title"] = self.title
    out_json["description"] = self.description
    return out_json
```

Створено інтерфейси репозиторіїв для доступу до даних із БД: DBConsumptions, DBAccount, DBEvents, DBUser. Ці класи надають методи для взаємодії з базою даних, такі як отримання, додавання, оновлення та видалення записів. Інтерфейси репозиторіїв для доступу до даних із бази даних

були створені для забезпечення абстракції між логікою застосунка та конкретною реалізацією доступу до даних. Основна мета – забезпечити розробників можливістю взаємодії з базою даних через визначені методи, не вдаваючись у деталі реалізації бази даних.

Ці інтерфейси створені для того, щоб ізолювати логіку доступу до даних від решти застосунку. Наприклад, коли розробник працює з об’єктами класу DataBaseManager, він взаємодіє з методами цих репозиторіїв, не звертаючи уваги на те, як саме виконується доступ до бази даних або яка конкретна база даних використовується. Це дозволяє легко змінювати реалізацію доступу до даних без зміни логіки самого застосунку. Приклад такого методу наведений у лістингу 3.2.

Лістинг 3.2 Метод для отримання списку подій за певний період:

```
def getEventsByYearMonth(self, year, month, userid):
    where = 'userid = '+str(userid)+' AND eventdate BETWEEN "'+str(year)+'-
'+('0' if month < 10 else '')+str(
        month)+'-01" AND "'+str(year)+'-'+('0' if month < 10 else
')+str(month+1)+'-01"'
    res = DBEvents.getEvents(self, where)
    out_res = []
    if len(res) > 0:
        for resItem in res:
            out_data = Event(resItem['id'], resItem['eventdate'],
                    resItem['title'], resItem['description']).to_json()
            out_res.append(out_data)
    return out_res
```

У цьому методі викликається метод getEvents із класу DBEvents, який відповідає за отримання списку подій за певний період із бази даних. Результати перетворюються в об’єкти класу Event.

Таким чином, клас DataBaseManager використовує інтерфейси репозиторіїв для взаємодії з базою даних, не вдаючись у деталі реалізації самого доступу до даних.

В коді використано низку компонентів і патернів програмування для створення програми для обліку фінансів:

– класи моделей даних. Ці класи використовуються для представлення різних типів об’єктів у програмі, таких як облікові записи користувачів, витрати та події. Вони містять атрибути, які відображають структуру даних, і методи, які дозволяють працювати з цими об’єктами;

– класи для взаємодії з базою даних. Ці класи відповідають за збереження, оновлення та видалення даних у базі даних SQLite. Вони використовуються для взаємодії з таблицями бази даних та забезпечення доступу до даних з різних частин програми;

– інструменти для серіалізації об’єктів. Ці класи використовуються для перетворення об’єктів у формат JSON та навпаки. Це дозволяє зберігати та передавати дані між різними частинами програми у структурованому форматі;

– інструменти для автентифікації та авторизації. Ці класи використовуються для створення та перевірки токенів, що дозволяють користувачам авторизуватись в системі та мати доступ до своїх особистих даних;

– інші утиліти та константи. Ці класи містять різноманітні утиліти та константи, які використовуються в програмі, такі як коди статусу сервера, повідомлення про помилки тощо.

Усі ці компоненти спільно працюють для створення функціональної програми для обліку фінансів. Користувачі можуть додавати свої витрати та події, переглядати їх, а також отримувати статистику щодо своїх фінансів. Дані зберігаються у базі даних та можуть бути доступні з будь-якого пристрою, який має доступ до цієї бази даних.

3.2.1 Реалізація клієнта

За допомогою Visual Studio Code було створено вебзастосунок на базі React. У процесі розробки були створені різноманітні компоненти, які включають кнопки, календар задач, меню статистики, поле для введення завдань, а також можливість запису витрат і доходів. Кожен компонент було докладно протестовано і налаштовано, щоб забезпечити правильне функціонування.

Надалі було реалізовано різноманітні сторінки, такі як сторінки входу в систему, реєстрації, перегляду статистики та її редагування. Додатково був впроваджений функціонал для пошуку, який дозволяє користувачеві швидко знаходити необхідну інформацію.

Окрім функціональності, була приділена увага вигляду та стилю елементів. Для кожної сторінки та компонента були розроблені відповідні стилі, щоб забезпечити гармонійний та привабливий дизайн.

Всі ці етапи розробки мають на меті створення зручного та естетичного інтерфейсу для користувачів, який би забезпечував їм зручність та ефективність у виконанні їх завдань.

3.3 Демонстрація роботи застосунку менеджера задач

3.3.1 Реєстрація та вхід в застосунок

Неавторизований користувач може або зареєструватися (рис. 3.1), або увійти у свій профіль (рис. 3.2).

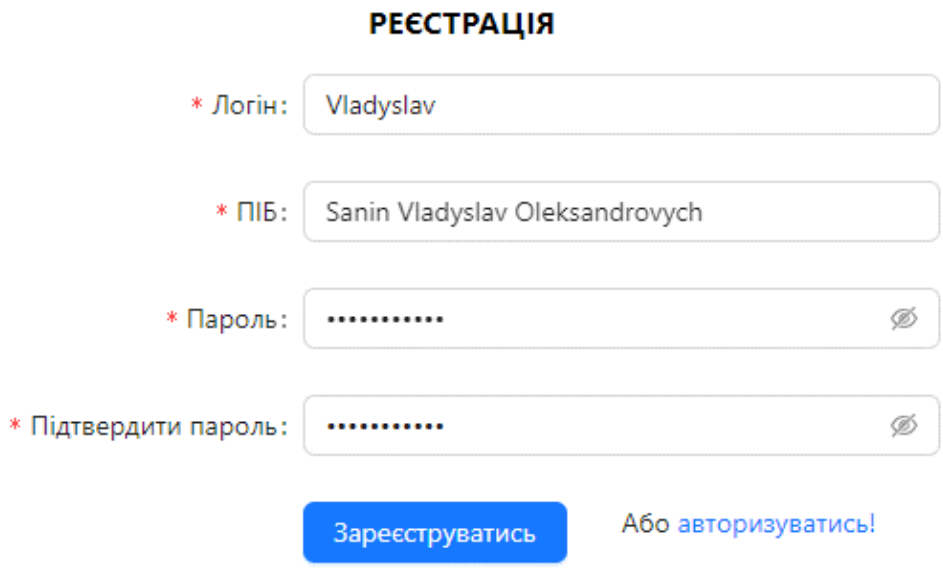

Рисунок 3.1 – Форма реєстрації

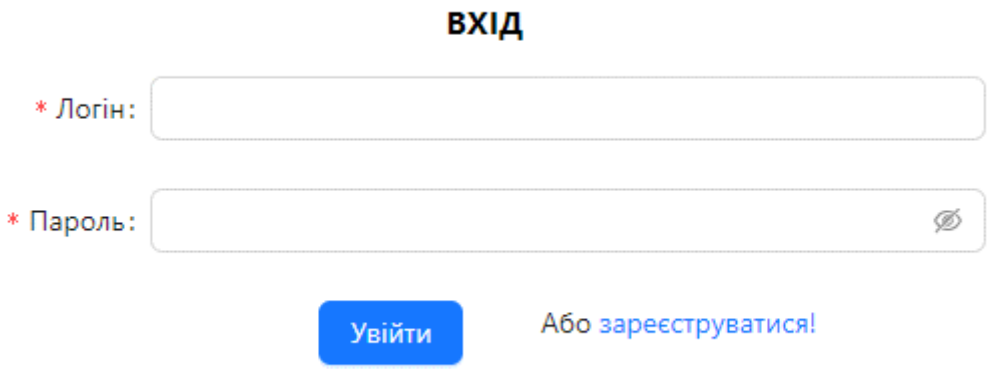

Рисунок 3.2 – Форма входу в систему

3.3.2 Головна сторінка застосунку

Після реєстрації або входу користувач переходить на головну сторінку для використання головного функціоналу (рис. 3.3). Поки користувач ще немає ніяких записів своїх задач, витрат або доходів, то календар немає виділених дат в яких повинна бути інформація.

Після успішної реєстрації або входу в систему, користувач автоматично перенаправляється на головну сторінку вебзастосунку, де може розпочати використання всіх доступних функцій (рис. 3.3). На головній сторінці можна побачити календар, який поки що не має виділених дат, оскільки користувач ще не створив жодних записів про свої завдання, витрати або доходи.

Цей підсумковий календар порожній, очікуючи на перші дані від користувача. Такий підхід дозволяє зробити процес початку використання застосунку максимально зручним та інтуїтивно зрозумілим. Після того, як користувач розпочне додавати свої записи, вони будуть відображатися на відповідних датах у календарі, що забезпечить йому зручний та структурований перегляд своєї активності.

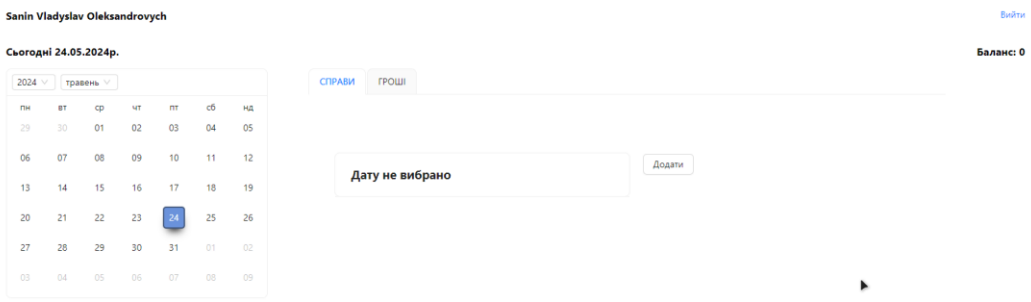

Рисунок 3.3 – Головна сторінка

Перш за все, користувач може заповнити інформацію про свій баланс, обравши розділ «Гроші». Він перейде певний розділ де обирається дата, сума балансу та обрати «Заповнення балансу» в полі «Тип дії». Для збереження інформації треба натиснути кнопку «Зберегти» (рис. 3.4).

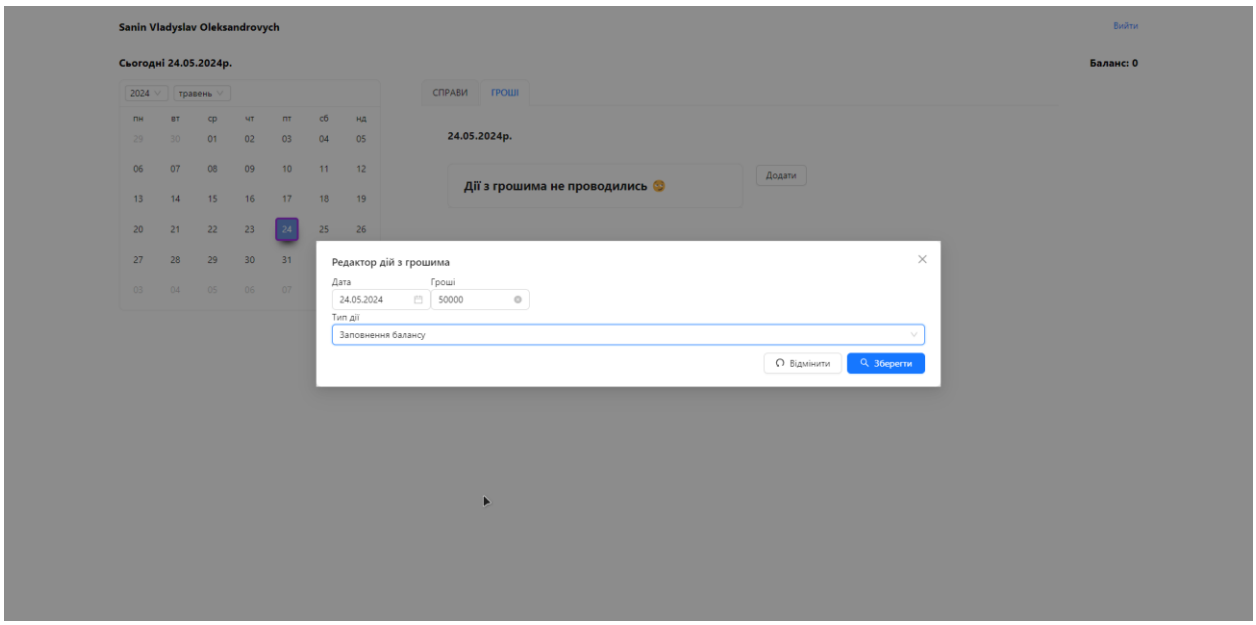

Рисунок 3.4 – Форма заповнення балансу

Після внесення нової інформації вона з’явиться на головній сторінці, а розділ «Статистика» почне заповнюватися відповідними даними (рис. 3.5).

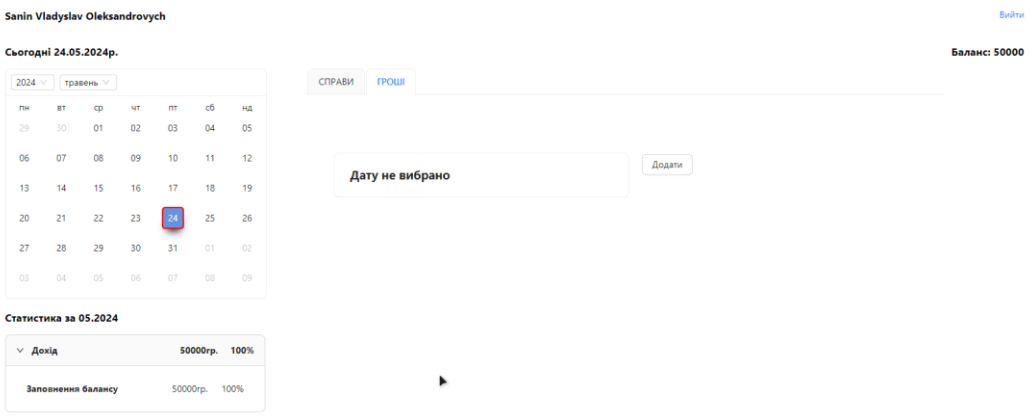

Рисунок 3.5 – Заповнення балансу

3.3.3 Дії з грошима

Якщо користувач бажає додати нову витрату або дохід, він може натиснути кнопку «Додати» (рис. 3.6). Після цього відкриється «Редактор дій з грошима», де користувач зможе вибрати потрібну дату, суму та відповідний розділ. Дати, на які додано певні записи, будуть виділені червоним кольором у календарі. Баланс користувача буде збільшуватися або зменшуватися залежно від обраного «Типу дії» (рис. 3.7).

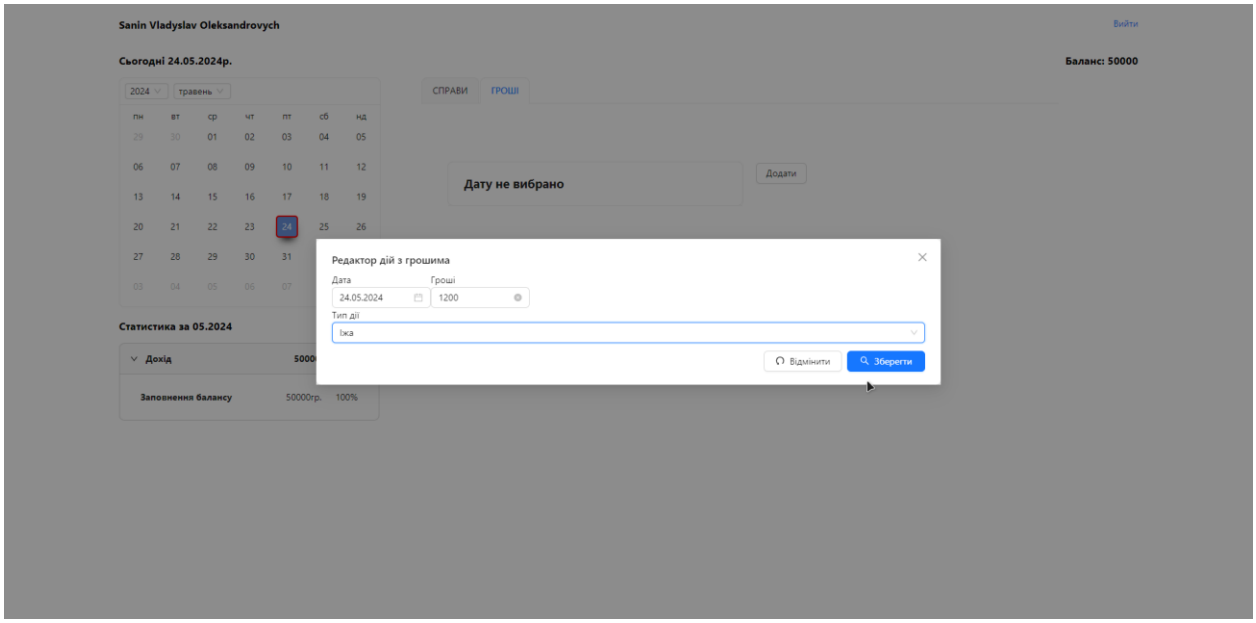

Рисунок 3.6 – Редактор дій з грошима

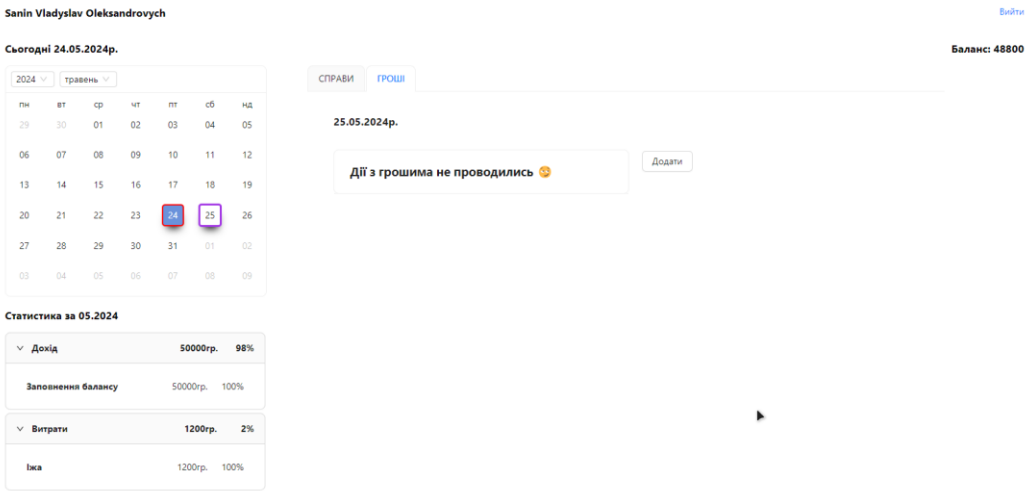

Рисунок 3.7 – Статистика дій з грошима

3.3.4 Редагування записаних дій з грошима

Кожна записана дія має можливість редагування. Якщо користувачу потрібно видалити запис, потрібно натиснути кнопку «Видалити» на потрібному записі (рис. 3.8). Також користувач може редагувати свої записи. Для цього потрібно натиснути кнопку «Редагувати», після відкриття вінка можна змінити суму та зберегти редагування (рис. 3.9).

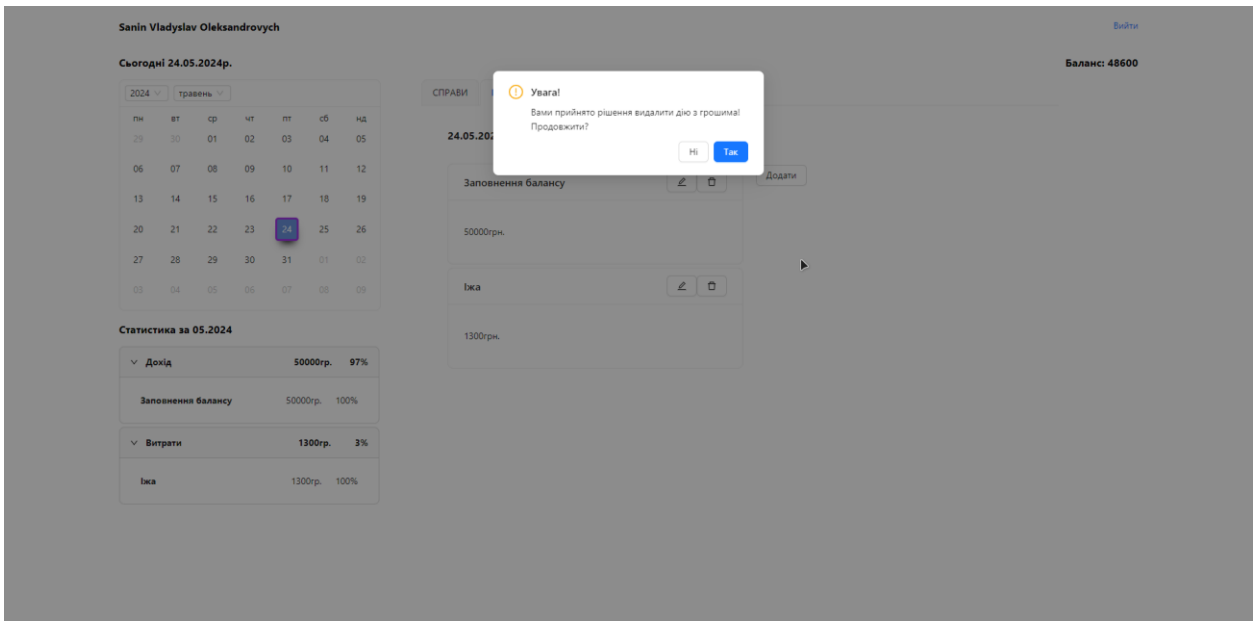

Рисунок 3.8 – Видалення запису

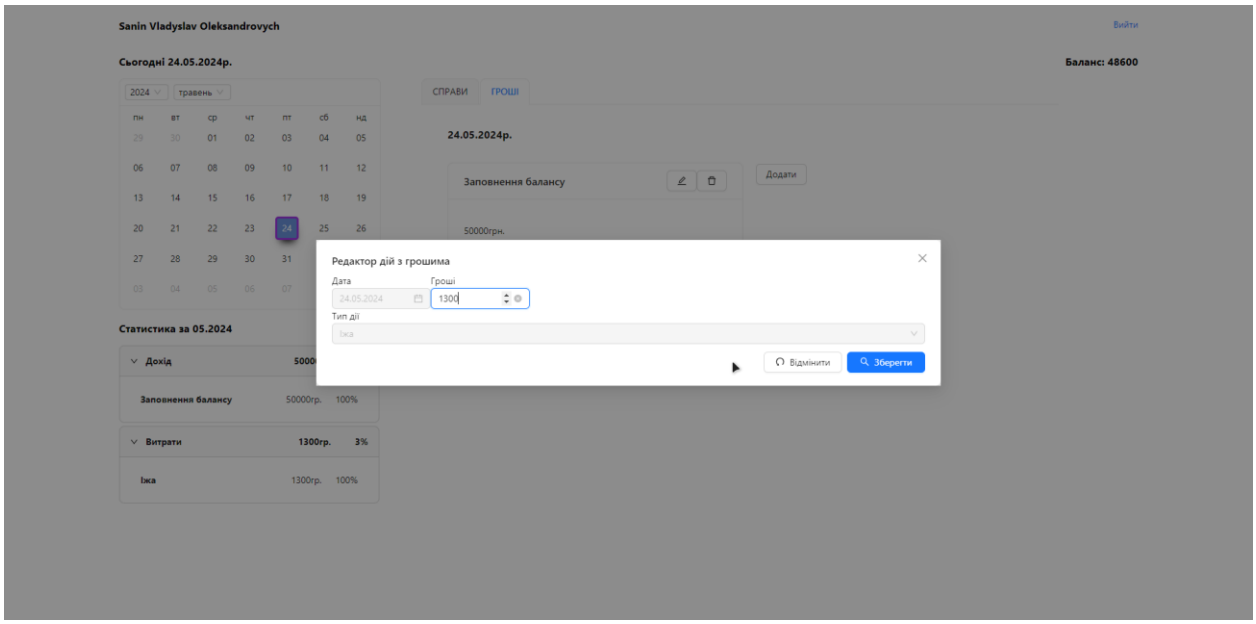

Рисунок 3.9 – Редагування запису

3.3.5 Взаємодія з задачами

Головна мета застосунку дати користувачу функціональний застосунок який допоможе збалансувати кожен свій день та стабілізувати свої витрати з доходами. Для цього потрібен зручний та інтуїтивний застосунок. Для цього

розділ «Справи» по функціоналу такий самий як і розділ «Гроші» (рис. 3.10).

Для запису своїх справ потрібно натиснути кнопку «Додати», після відкриється меню день користувач може обрати потрібний день, дати заголовок справі та написати опис, для збереження треба натиснути «Зберегти» (рис. 3.11). Дати, на які додано певні записи, будуть виділені червоним кольором у календарі.

Для видалення певної дії треба натиснути на кнопку «Видалити» біля певного запису (рис. 3.12).

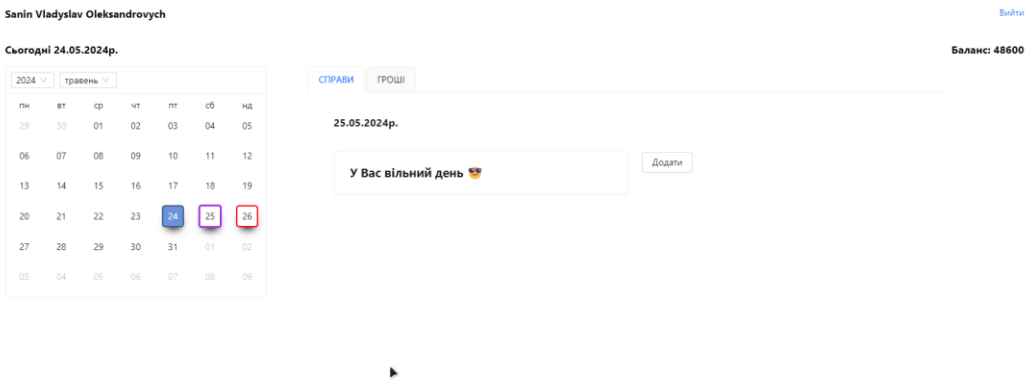

Рисунок 3.10 – Головна сторінка справ

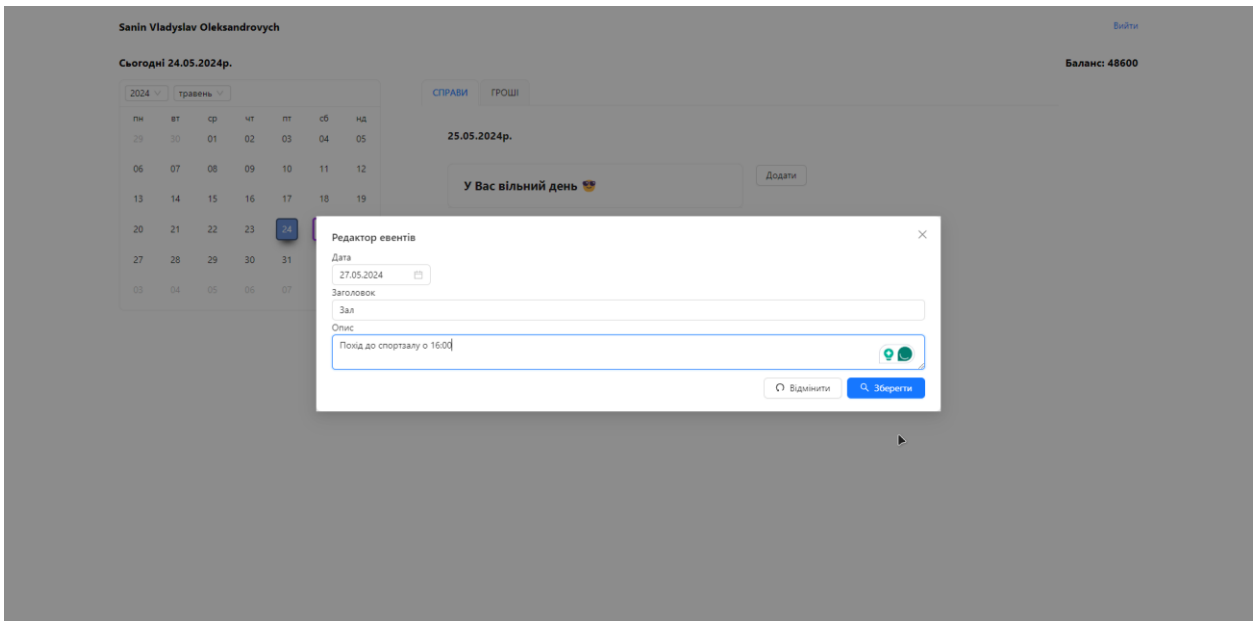

Рисунок 3.11 – Додавання справи

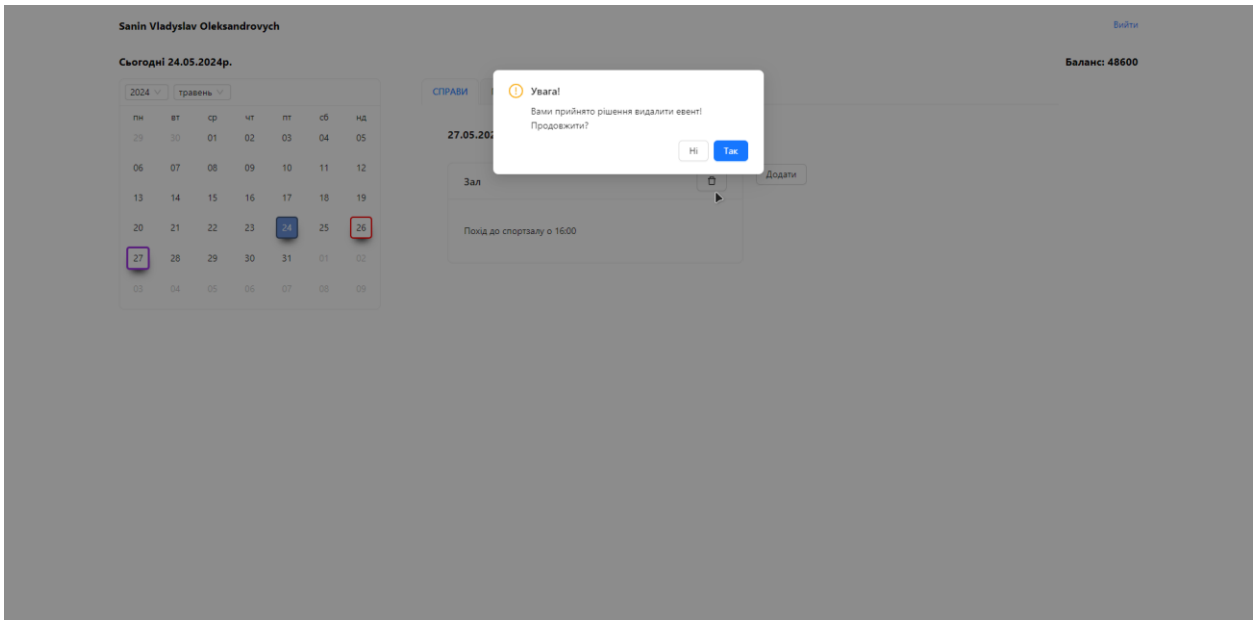

Рисунок 3.12 – Видалення певної справи

3.4 Перспективи розвитку менеджера задач

Можливі такі варіанти розвитку функціоналу застосунку:

– створення функціоналу для використання кількох гаманців на одного користувача, з можливістю швидкого перемикання між ними через кнопку «Мій гаманець»;

– додання можливості виставляти пріоритети для справ, такі як низький, середній та високий рівень;

– введення термінів виконання справ та сповіщень про наближення дедлайнів;

– розробка секундоміра для відстеження часу, витраченого на виконання конкретних справ;

– розробка функції, що дозволяє користувачам самостійно створювати додаткові типи витрат та доходів;

– додання діаграм для відображення статистики витрат та доходів за місяць;

– введення розділу «Бюджет», щоб користувачі могли ефективніше заощаджувати гроші і час на важливі речі;

– додання системи оповіщень про заплановані на сьогодні справи, завдання на завтра, а також забуті справи;

– розробка можливості заповнювати справи та грошові маніпуляції через «CheckBox», щоб позначати виконані пункти.

Отже, перспективи подальшого розвитку цього застосунку є досить великими.

ВИСНОВКИ

У рамках кваліфікаційної роботи був створений вебзастосунок менеджера задач для ефективного управління своїм часом та фінансами. В процесі розробки проєкту був проведений детальний аналіз схожих застосунків, їхній функцій та дизайну. Визначили спільні риси, переваги та недоліки, для створення більш зручного та кращого застосунку.

Освоєно та використано необхідні технології. Серверну частину було розроблено за допомогою Python та Flask Framework, клієнтську частину реалізовано з використанням мови програмування JavaScript і бібліотеки React та Ant Design. Також була створення база даних за допомогою SQLite та розроблений користувацький інтерфейс.

Цей вебзастосунок дозволяє користувачам записувати свої справи та події, записувати свої доходи та витрати, переглядати певну статистику за вибраний період часу. Після тестування було визначено для подальшого покращення та розширення функціоналу застосунка.

ПЕРЕЛІК ДЖЕРЕЖ ПОСИЛАННЯ

1. Клієнтська частина URL: https://blog.ithillel.ua/articles/building-dynamic-web-applications-using-mvc (дата звернення 18.04.2024)

2. Архітектура веб додатків URL: https://medium.com/@IvanZmerzlyi/архітектура-веб-додатків-ca4c82f75bcf (дата звернення 19.04.2024)

3. Розробка з боку Frontend сервер URL: https://dan-it.com.ua/uk/blog/rozrobka-z-boku-front-end-shho-ce-take-i-chim-vidriznjaietsja-vid-back-end/ (дата звернення 20.04.2024)

4. Frontend розробка URL: https://brainlab.com.ua/uk/blog-uk/front-end-rozrobka (дата звернення 20.04.2024)

5. Що таке Frontend сервер і як він працює URL: https://mist.home.cx.ua/ukraincyam/shho-take-frontend-server-i-yak-vin-pracyuie.html (дата звернення 21.04.2024)

6. Backend розробка URL: https://brainlab.com.ua/uk/blog-uk/back-end-rozrobka (дата звернення 24.04.2024)

7. Що таке Backend розробка URL: https://wezom.com.ua/blog/chto-takoe-back-end-razrabotka (дата звернення 24.04.2024)

8. Клієнт-серверна архітектура URL: https://javarush.com/ua/quests/lectures/ua.questservlets.level14.lecture00 (дата звернення 25.04.2024)

9. Взаємодія клієнт сервер URL: https://ua5.org/technol/2094-vzayemodiya-kliyent-server.html (дата звернення 28.04.2024)

10. Клієнт серверна архітектура URL: https://www.wikiwand.com/uk/Клієнт-серверна_архітектура (дата звернення 28.04.2024)

11. Мова програмування Python URL: https://robotdreams.cc/uk/blog/496-naypopulyarnishi-movi-programuvannya (дата звернення 30.04.2024)

12. Що таке Python URL: https://acode.com.ua/intro-python/ (дата звернення 30.04.2024)

13. Python мова програмування URL: https://freehost.com.ua/ukr/faq/wiki/chto-takoe-jazik-programmirovanija-python/ (дата звернення 02.05.2024)

14. Як почати працювати з Python URL: https://acode.com.ua/get-started-python/ (дата звернення 02.05.2024)

15. Путівник мовою програмування Python URL: https://pythonguide.rozh2sch.org.ua/ (дата звернення 02.05.2024)

16. Підручник з Python URL: https://docs.python.org/uk/3/tutorial/index.html (дата звернення 04.05.2024)

17. Введення до SQLite URL: https://wox.in.ua/threads/vvedennja-do-sqlite-kompaktna-ta-vbudovuvana-baza-danix.68/ (дата звернення 05.05.2024)

18. Введення в базу даних Sqlite URL: http://yoip.com.ua/vvedennya-v-bazu-danih-sqlite/ (дата звернення 06.05.2024)

19. Що таке Sqlite URL: https://freehost.com.ua/ukr/faq/wiki/chto-takoe-sqlite/ (дата звернення 09.05.2024)

20. Підручник з бази даних Sqlite URL: https://www.guru99.com/uk/sqlite-tutorial.html (дата звернення 09.05.2024)

21. Як користуватися sqlite database browser URL: https://striminh.zapisi.cx.ua/ukraincyam/yak-koristuvatisya-sqlite-database-browser.html (дата звернення 11.05.2024)

22. Мікросервіси URL: https://qalight.ua/baza-znaniy/shho-take-mikroservisna-arhitektura-pz/ (дата звернення 12.05.2024)

23. Мікросервісна архітектура URL: https://medium.com/@IvanZmerzlyi/microservices-architecture-461687045b3d (дата звернення 15.05.2024)

24. Для чого використовуються мікросервіси URL: https://tqm.com.ua/ua/likbez/ua-articles/micro-poslygi (дата звернення 15.05.2024)

25. Важливість інструмент конвертера в SQLite URL: https://www.datanumen.com/uk/блоги/22-найкращі-sql-server-to-sqlite-converter-tools-free-download/ (дата звернення 16.05.2024)

26. Using DB Browser for SQLite URL: https://datacarpentry.org/sql-socialsci/02-db-browser.html (дата звернення 17.05.2024)

27. Популярні фреймворки Python URL: https://foxminded.ua/freimvorky-python/ (дата звернення 17.05.2024)

28. How To Make a Web Application Using Flask in Python URL: https://www.digitalocean.com/community/tutorials/how-to-make-a-web-application-using-flask-in-python-3 (дата звернення 18.05.2024)

29. Wikipedia. React. URL: https://uk.wikipedia.org/wiki/React (дата звернення 19.05.2024)

30. Бібліотека компонентів Ant Design URL: https://www.youtube.com/watch?v=LfKgQ5mAcRs (дата звернення 19.05.2024)