

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО”
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ АТОМНОЇ ТА ТЕПЛОВОЇ ЕНЕРГЕТИКИ
кафедра ЦИФРОВИХ ТЕХНОЛОГІЙ В ЕНЕРГЕТИЦІ

“До захисту допущено”
Завідувач кафедри ЦТЕ
_____ Наталія АУШЕВА

“__” _____ 2024 р.

Дипломна робота
на здобуття ступеня бакалавра
за освітньо-професійною програмою
“Цифрові технології в енергетиці”
зі спеціальності 122 “Комп’ютерні науки”

на тему: Веб-застосунок обліку особистого бюджету

Виконав:
студент IV курсу, групи ТР-03

КАРУК Артемій Павлович
(прізвище, ім’я, по батькові)

_____ (підпис)

Керівник:

Доцент каф. цифрових технологій в енергетиці, к.т.н,
Ірина МИХАЙЛОВА
(посада, вчене звання, науковий ступінь, ім’я, ПРІЗВИЩЕ)

_____ (підпис)

Рецензент: доцент каф. теплової та альтернативної енергетики,
доцент, к.т.н., Ірина ФУРТАТ
(посада, вчене звання, науковий ступінь, ім’я, ПРІЗВИЩЕ)

_____ (підпис)

Н.контроль: асистент Головакін Микита Андрійович
(посада, ім’я, ПРІЗВИЩЕ)

_____ (підпис)

Засвідчую, що у цій дипломній роботі немає
запозичень з праць інших авторів без відповідних
посилань.

Студент _____
(підпис)

Національний технічний університет України
“Київський політехнічний інститут імені Ігоря Сікорського”

Навчально-науковий інститут атомної та теплової
енергетики

Кафедра цифрових технологій в енергетиці

Рівень вищої освіти перший

(бакалаврський)

спеціальність 122 Комп'ютерні науки

освітня програма «Комп'ютерний моніторинг та геометричне
моделювання процесів та систем»

ЗАТВЕРДЖУЮ

Завідувач кафедри

Наталія АУШЕВА

(підпис)

” ” 2024р.

ЗАВДАННЯ
на дипломну роботу студенту

Каруку Артемію Павловичу

(прізвище, ім'я, по батькові)

1. Тема роботи Веб-застосунок для обліку особистого бюджету

Керівник Михайлова Ірина Юріївна, к.т.н.

(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від “15” вересня 2023р. № 2039-с

2. Термін подання студентом роботи 09.06.2023

3. Вихідні дані до роботи мова програмування Python

4. Перелік питань, які потрібно розробити _____

1) провести аналіз існуючих веб-інструментів для обліку особистого
бюджету;

- 2) обґрунтувати вибір програмних засобів реалізації;
- 3) розробити програмне забезпечення;
- 4) провести тестування розробленого програмного забезпечення
5. Орієнтований перелік ілюстративного матеріалу: мета та завдання роботи, постановка задачі, порівняння з аналогами, використані технології, діаграма прецедентів, схема бази даних, інтерфейс користувача
6. Дата видачі завдання «15» вересня 2023 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітки
1.	Затвердження теми роботи	19.09.2023 р.	Виконано
2.	Вивчення та аналіз задачі	22.04.2023 - 26.04.2023 р.	Виконано
3.	Розробка архітектури та загальної структури системи	27.04.2023 - 29.04.2023 р.	Виконано
4.	Розробка частин окремих підсистем	29.04.2023 - 4.05.2023 р.	Виконано
5.	Програмна реалізація системи	5.05.2023 - 14.05.2023 р.	Виконано
6.	Оформлення дипломної роботи	14.05.2023 - 15.05.2023 р.	Виконано
7.	Захист програмного продукту	16.05.2023	Виконано
8.	Передзахист	06.06.2023.	Виконано
9.	Подання готової роботи на кафедрі	07.06.2024	
10.	Захист	19.06.2023 р.	

Студент

(підпис)

Артемій КАРУК

(ім'я, ПРІЗВИЩЕ)

Керівник роботи

(підпис)

Ірина МИХАЙЛОВА

(ім'я, ПРІЗВИЩЕ)

АНОТАЦІЯ

Метою дипломної роботи є розробка веб-застосунку для обліку особистого бюджету, відмінністю якого є стимулювання усвідомлених витрат шляхом завчасного їх планування. Розроблений веб-застосунок надає користувачам можливість реєструватися, додавати власні дані як власноруч, так і імпортувавши csv файл, створювати місячні плани із щоденними або одноразовими витратами, додавати окремі витрати та розраховувати різницю між запланованими та фактичними витратами. Один з головних функціональних елементів застосунку – це розрахунок вільних коштів та стимулювання створення накопичень за рахунок невитрачених грошей і відображення цих розрахунків на графіках.

Застосунок написаний за допомогою інтегрованого середовища розробки PyCharm Professional 2024. Для розробки веб-застосунку використано технології: Python, HTML, CSS, JavaScript, SQL.

Записка містить 52 сторінки, 36 рисунків, 2 додатки і 9 посилань.

Ключові слова: веб-застосунок, особистий бюджет, планування, Python, SQL.

ABSTRACT

The purpose of the thesis is to develop a web application for personal budget accounting, the distinctive feature of which is to stimulate conscious spending through advance cost planning. The developed web application allows users to register, add their own data either manually or by importing a csv file, create monthly plans with daily or one-time expenses, add individual expenses, and calculate the difference between planned and actual expenses. One of the main functional elements of the application is the calculation of free funds and stimulation of savings, due to unspent money and displaying these calculations on graphs.

The application was written using the PyCharm Professional 2024 integrated development environment. The following technologies were used to develop the web application: Python, HTML, CSS, JavaScript, SQL.

The paper contains 52 pages, 36 figures, 2 appendices and 9 references.

Keywords: web application, tables, calculations, Python, SQL.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ І ТЕРМІНІВ.....	7
ВСТУП.....	8
1 ЗАДАЧА ОБЛІКУ ОСОБИСТОГО БЮДЖЕТУ	11
2 АНАЛІЗ ІСНУЮЧИХ СИСТЕМ ОБЛІКУ ОСОБИСТОГО БЮДЖЕТУ	12
3 ЗАСОБИ РОЗРОБКИ.....	16
3.1 Мова програмування Python	16
3.2 Фреймворк Flask.....	16
3.3 Фреймворк SQLAlchemy	17
3.4 Фреймворк Alembic	18
3.5 Фреймворк Flask-WTF	19
3.6 Бібліотека Bcrypt	20
3.7 Бібліотека Bootstrap	21
3.8 Система контролю версій Git.....	22
4 ОПИС ПРОГРАМНОЇ РЕАЛІЗАЦІЇ	23
4.1 Структура та архітектура системи обліку особистого бюджету.....	23
4.2 Функціональність системи обліку особистого бюджету	28
4.3 Схема бази даних	29
5 РОБОТА КОРИСТУВАЧА З ПРОГРАМНОЮ СИСТЕМОЮ	32
5.1 Опис системних вимог	32
5.2 Інсталяція	33
5.3 Робота особистого облікового запису.....	40
5.3.1 Навігація по сторінках.....	41
5.3.2 Створення місячного плану	43
5.3.3 Приклад розрахунків	47
5.3.4 Виведення результатів.....	50
5.3.5 Скидання пароля	51
ВИСНОВКИ.....	53
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ	54
Додаток А.....	55
Додаток Б.....	59

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

UX — досвід користування (англ. User Experience).

UI — інтерфейс користувача (англ. User Interface).

Back-end — серверна частина застосунку.

БД — база даних.

СУБД — система управління базами даних.

IDE — інтегроване середовище розробки (англ. Integrated Development Environment).

ООП — Об'єктно-орієнтоване програмування (англ. Object-oriented programming)

ВСТУП

У сучасному світі управління особистим бюджетом стає все більш важливою складовою успішного фінансового планування для багатьох людей. Незалежно від статусу чи доходу, ефективне керування фінансами є ключовим аспектом фінансової стабільності та забезпечення майбутньої безпеки.

Предметна область управління особистим бюджетом охоплює широкий спектр аспектів, включаючи визначення доходів, витрат, бюджетування, збереження та інвестування. Вона також охоплює аналіз фінансових показників, планування на майбутнє та прийняття фінансових рішень на основі цього аналізу.

Ця область стає все більш важливою в умовах зростання нестабільності економіки, зміни ринкових умов та особистих фінансових цілей. Для багатьох людей, навіть тих, хто має достатній рівень доходу, ефективне керування бюджетом дозволяє досягти фінансового комфорту, зменшити стрес і покращити якість життя.

Програмні засоби для управління особистим бюджетом надають користувачам можливість вести облік своїх фінансів, аналізувати їх та планувати майбутні витрати. Ці засоби можуть мати різноманітні функціональні можливості, включаючи автоматичне відстеження транзакцій, створення бюджетів, нагадування про заплановані витрати та інші.

Враховуючи значення фінансового управління для багатьох людей, розробка програмного забезпечення в цій області відіграє важливу роль у полегшенні процесу управління особистим бюджетом та покращенні фінансової грамотності користувачів.

За даними дослідження "Фінансова грамотність, фінансова інклюзія та фінансовий добробут в Україні у 2021 році", проведеного проєктом USAID спільно з Національним банком України, середній рівень фінансової грамотності населення України складає 12,3 бала, що є меншим від мінімального цільового рівня, рекомендованого Організації економічного співробітництва та розвитку (14 балів). Згідно цього дослідження на підвищення компетентності у фінансових питаннях

позитивно впливає застосування на практиці фінансових знань: контроль доходів і витрат, фінансове планування, заощадження.

Результати нещодавно проведеного дослідження в Україні, як і дані ОЕСР у розрізі інших країн, підтверджують, що пришвидшити зростання фінансової грамотності допоможе користування сучасними інформаційними технологіями. Різниця в рівні фінансової обізнаності між користувачами та не користувачами становить 2,5 бала (12,4 та 9,9 бала відповідно). [1]

З описаного вище дослідження можливо зробити висновок, що розробка застосунку є доцільною та актуальною, з погляду підвищення фінансової грамотності населення для тих користувачів, які будуть ним користуватися, оскільки є помітна різниця в рівні фінансової грамотності між тими хто користується інформаційними технологіями та тими хто не користується.

Метою даної дипломної роботи є розробка та реалізація веб-застосунку, спрямованого на облік особистого бюджету. Основна ідея полягає у створенні зручного та інтуїтивно зрозумілого інструменту, який дозволить користувачам ефективно керувати своїми фінансами, а також аналізувати їх для прийняття обґрунтованих фінансових рішень.

Попри наявність деяких програм та мобільних додатків для обліку особистого бюджету, багато з них мають обмежені функціональні можливості або не відповідають усім потребам користувачів. Таким чином, наявна проблема полягає в тому, що існуючі рішення не завжди задовольняють потреби користувачів у повному обсязі.

Робота складатиметься з наступних розділів.

1. Огляд літератури та аналіз існуючих рішень. У цьому розділі буде проведено огляд наукової літератури та існуючих веб-застосунків для обліку особистого бюджету. Буде проведений аналіз їхніх переваг та недоліків.

2. Архітектура та дизайн веб-застосунку. У даному розділі буде описана загальна архітектура розроблюваного веб-застосунку, його функціональні можливості та користувацький інтерфейс.

3. Реалізація та тестування веб-застосунку. В цьому розділі будуть описані деталі реалізації веб-застосунку, включаючи використані технології та методи програмування. Також буде проведено тестування розробленого застосунку для перевірки його працездатності та надійності.

4. Висновки та перспективи подальших досліджень. У кінцевому розділі будуть сформульовані висновки щодо результатів роботи, надані рекомендації щодо подальшого вдосконалення веб-застосунку та визначені перспективи подальших досліджень у даній області.

Ця дипломна робота має велике значення для практичного застосування здобутих знань та навичок у сфері розробки програмного забезпечення та фінансового планування. Вона сприятиме покращенню фінансової грамотності серед користувачів та допоможе їм ефективніше управляти своїми фінансами.

Для реалізації системи використані наступні технології, мови і фреймворки, такі як HTML, CSS, мова програмування Python та його фреймворк Python Flask для серверної частини та SQLAlchemy та мова SQL для обробки запитів до бази даних.

1 ЗАДАЧА ОБЛІКУ ОСОБИСТОГО БЮДЖЕТУ

Метою дипломної роботи є розробка веб-застосунку обліку особистого бюджету.

Для досягнення поставленої мети необхідно вирішити наступні завдання:

- проаналізувати існуючі програмні рішення;
- розробити архітектуру та структуру системи обліку особистого бюджету;
- розробити інтерфейс користувача системи обліку особистого бюджету;
- протестувати розроблену систему обліку особистого бюджету.

Ключові задачі, які має виконувати система обліку особистого бюджету:

- ведення обліку доходів та витрат;
- можливість вносити дані про доходи по окремих днях та місяцях;
- можливість вносити свої витрати із зазначенням їх категорій, обсягу витрат та одиниць вимірювання;
- можливість імпортувати дані з csv файлу;
- надавати можливість реєструватись та мати окремий обліковий запис для кожного користувача;
- генерація звітів та аналітичних даних про стан фінансів користувача;
- управління категоріями витрат для зручного аналізу.

Вхідна інформація – дохід та витрати користувача (внесені вручну або з csv файлу), категорії витрат, терміни та суми планованих витрат.

Вихідна інформація – звіти, статистика про фінансовий стан користувача.

Потенційні користувачі системи – це будь-яка особа, яка бажає здійснювати контроль та аналіз своїх фінансів, від молодих професіоналів до досвідчених фінансових аналітиків.

2 АНАЛІЗ ІСНУЮЧИХ СИСТЕМ ОБЛІКУ ОСОБИСТОГО БЮДЖЕТУ

Для того, аби визначити ключові аспекти того на чому необхідно сфокусувати свою увагу під час розробки програмного забезпечення важливим буде провести аналіз вже існуючих програмних систем, визначити їх ключові переваги та недоліки, й відповідно перенести вдалі рішення та покращити те, що має свої недоліки.

Перед початком розробки веб-застосунку для обліку особистого бюджету важливо вивчити та проаналізувати існуючі програмні продукти в цій області. Деякі з найпопулярніших і аналогічних програмних систем включають:

- Mint – це онлайн-сервіс для управління фінансами, який дозволяє користувачам відстежувати доходи та витрати, робити бюджети та отримувати персоналізовані поради щодо фінансового планування (рисунок 2.1);

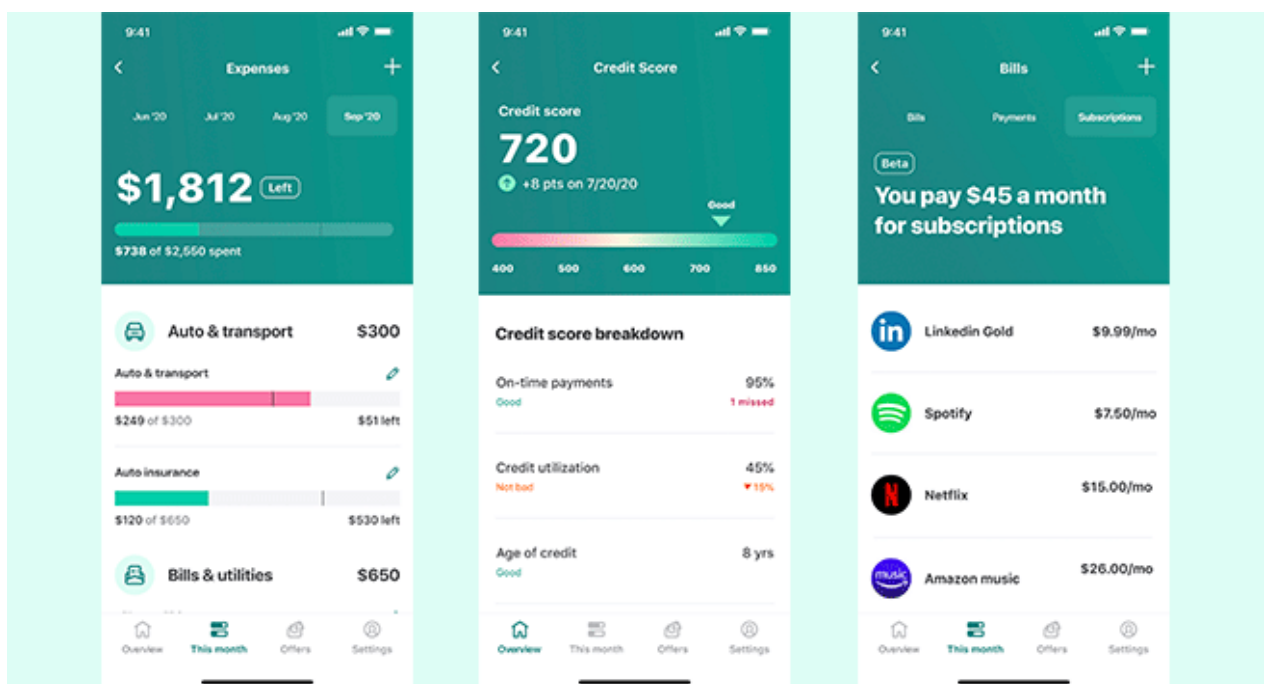


Рисунок 2.1 – Вигляд застосунку Mint

- YNAB (You Need a Budget) – це програма, яка допомагає користувачам жити за своїми можливостями, розподіляючи гроші на основі принципів бюджетування на основі «нульового балансу» (рисунк 2.2);

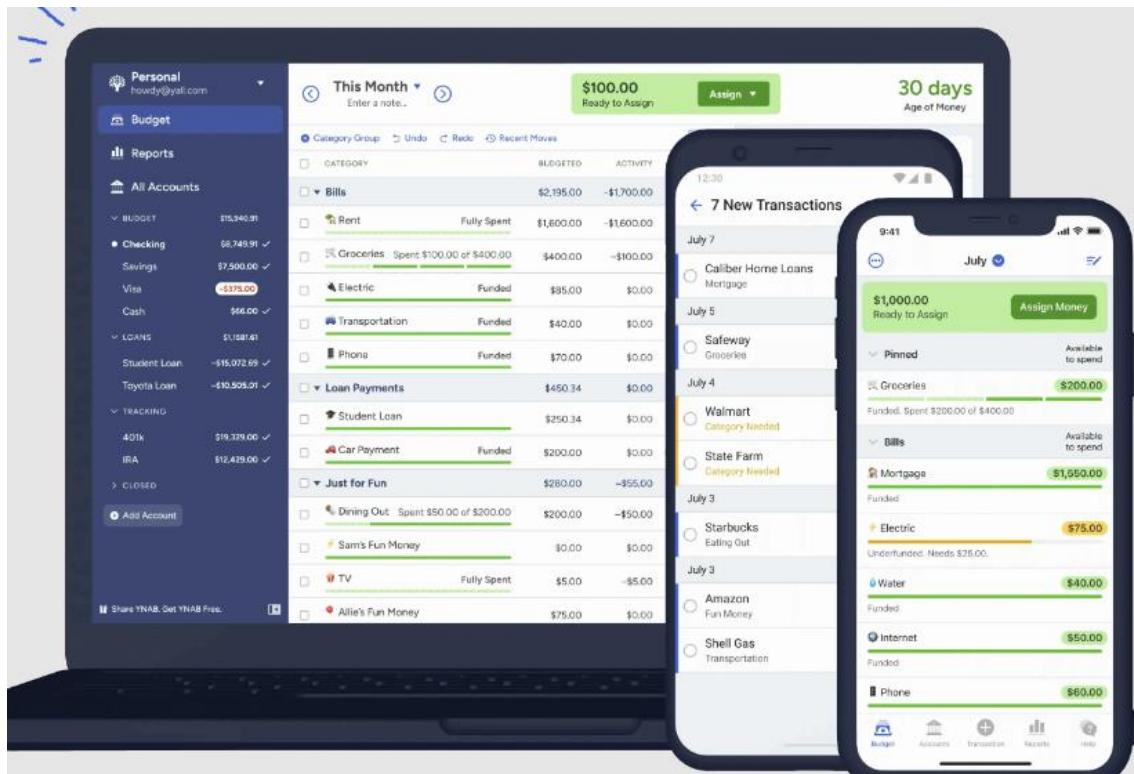


Рисунок 2.2 – Вигляд застосунку YNAB

- RocketGuard – автоматично аналізує фінанси користувача, відстежує витрати та надає рекомендації щодо економії грошей (рисунк 2.3).

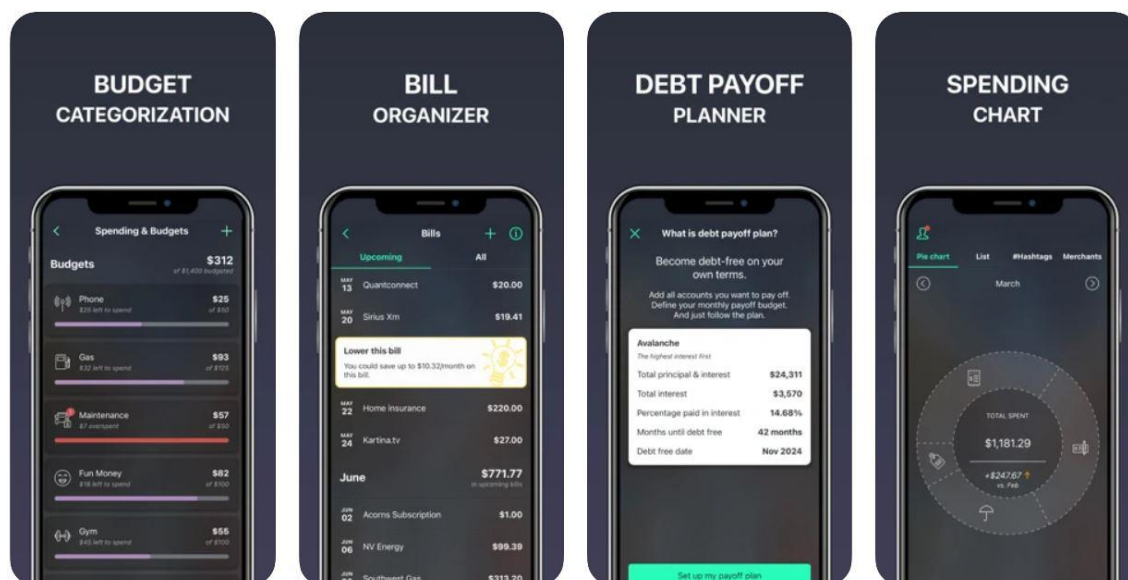


Рисунок 2.3 – Вигляд застосунку PocketGuard

Кожна з програмних систем має свої переваги та недоліки. Короткий аналіз їх функціональних можливостей та особливостей наведено у таблиці 2.1.

Таблиця 2.1 – Порівняння переваг та недоліків існуючих систем

Застосунок	Переваги	Недоліки
Mint	Пропонує детальні звіти про витрати та аналіз тенденцій.	Не має десктопного інтерфейсу. Відсутнє відстеження заощаджень.
YANAB	Зосереджений на проактивному бюджетуванні. Чистий і без відволікань інтерфейс, пропонує детальні і налаштовувані звіти.	Відсутня можливість внесення даних з таблиці або csv файлу. Відсутнє відстеження заощаджень.
Pocketguard	Простий і зручний у користуванні інтерфейс.	Обмежені можливості налаштування. Відсутнє відстеження заощаджень.

Як бачимо з таблиці 2.1, у перелічених веб-застосунках є як переваги, так і недоліки, що обов'язково будуть враховані під час розробки програмного забезпечення.

Проведений аналіз існуючих програмних систем показав, що на ринку існують різноманітні інструменти для управління особистим бюджетом з різними функціональними можливостями та підходами. Цей аналіз допоможе врахувати сильні та слабкі сторони існуючих рішень при розробці власного веб-застосунку для обліку особистого бюджету.

Ключовою відмінністю розробленого програмного продукту та його основою ідеєю, що відрізнятиме його від конкурентів буде розрахунок різниці між запланованими та фактичними витратами. Ідея та «філософія» застосунку – стимулювання користувача заощаджувати та робити накопичення.

3 ЗАСОБИ РОЗРОБКИ

В цьому розділі розглянуто використовувані засоби розробки, програмне забезпечення, мови програмування та використані бібліотеки, фреймворки.

3.1. Мова програмування Python

Як мову програмування було обрано Python. Python є високорівневою мовою програмування, що широко застосовується в різних галузях, зокрема у веб-розробці. У контексті веб-розробки Python використовується як мова для створення серверної логіки (бекенду), що взаємодіє з клієнтською стороною.

Однією з основних переваг Python для веб-розробки є його простота та лаконічність. Ця мова виділяється чистим і зрозумілим синтаксисом, який дозволяє розробникам легко створювати функціональні веб-додатки.

Для веб-розробки Python використовують різні фреймворки, такі як Flask. Їхньою особливістю є зручні інструменти для роботи з HTTP-запитами, обробки URL-адрес, керування сесіями користувачів, взаємодії з базами даних тощо.

Також однією перевагою Python для веб-розробки є існування великою кількості сторонніх бібліотек та пакетів. Це, своєю чергою, дозволяє розробникам зручно використовувати готові рішення для реалізації різних завдань [2].

3.2. Фреймворк Flask

Flask - це легковажний веб-фреймворк для Python, який дозволяє швидко і легко створювати веб-додатки. Він був розроблений з метою забезпечити простоту та гнучкість, що робить його ідеальним для невеликих та середніх проектів, а також для прототипування. Flask базується на концепції мінімалізму, що означає, що він надає

лише найнеобхідніший набір функцій, залишаючи розробникам свободу вибору додаткових компонентів та модулів відповідно до потреб проекту.

Основна функція Flask полягає в тому, щоб дозволити розробникам створювати маршрути, які відповідають на запити від користувачів, та визначати, що повинно відображатися у відповідь на ці запити. Це забезпечує легку організацію та управління веб-сторінками та API. Flask також підтримує розширюваність за рахунок використання численних сторонніх плагінів, що дозволяє легко додавати функціональність, таку як робота з базами даних, управління автентифікацією, обробка форм та інше.

Однією з головних переваг Flask є його простота і гнучкість. Він не нав'язує жорстких структур або шаблонів, дозволяючи розробникам організовувати свій код так, як вони вважають за потрібне. Це робить Flask популярним вибором для розробників, які цінують свободу і контроль над своїм кодом. Додатково, Flask має дуже детальну документацію та активну спільноту, що робить його доступним навіть для початківців у веб-розробці. [3]

Використання Flask дозволяє швидко розробляти і тестувати веб-додатки, що є великим плюсом у середовищах, де швидкість розробки та впровадження має вирішальне значення. Його легковажність і модульність дозволяють легко масштабувати додатки та інтегрувати їх з іншими сервісами та бібліотеками. Все це робить Flask потужним інструментом для сучасних веб-розробників, які шукають баланс між простотою та функціональністю.

3.3. Фреймворк SQLAlchemy

Flask-SQLAlchemy - це розширення для веб-фреймворку Flask.. Це розширення забезпечує зручний спосіб роботи з базами даних, надаючи можливість розробникам використовувати об'єктно-орієнтований підхід до взаємодії з базою даних замість написання сирого SQL-коду. Flask-SQLAlchemy робить процес управління базою

даних більш інтуїтивно зрозумілим і простим, що сприяє підвищенню продуктивності розробки.

Основне призначення Flask-SQLAlchemy - забезпечити легкий доступ до баз даних, підтримуючи всі основні функції SQLAlchemy, такі як декларативні моделі, запити, сесії та міграції. Це дозволяє розробникам визначати структуру бази даних у вигляді Python класів, що значно полегшує роботу з даними та їх маніпулювання. Крім того, Flask-SQLAlchemy автоматично керує з'єднанням з базою даних та інтегрується з іншими компонентами Flask, що робить роботу з базою даних більш безшовною і ефективною.

Однією з головних переваг Flask-SQLAlchemy є його здатність забезпечувати більш чистий і організований код. Замість того, щоб писати складні SQL-запити, розробники можуть використовувати об'єктно-орієнтований синтаксис, що підвищує читабельність і підтримуваність коду. Це особливо корисно в великих проектах, де структура бази даних може бути складною і динамічно змінюватися. Крім того, Flask-SQLAlchemy спрощує процес міграції баз даних, що є важливим аспектом при розробці додатків, які постійно еволюціонують. [3]

Flask-SQLAlchemy також сприяє більш ефективній розробці завдяки своїй гнучкості та розширюваності. Вона підтримує різні типи баз даних, включаючи SQLite, PostgreSQL, MySQL та інші, що робить її універсальним інструментом для різних проектів. Крім того, спільнота розробників активно підтримує та оновлює це розширення, забезпечуючи його стабільність і сумісність з новими версіями Flask і SQLAlchemy. Завдяки цим перевагам, Flask-SQLAlchemy є популярним вибором серед розробників, які бажають створювати ефективні та масштабовані веб-додатки.

3.4. Фреймворк Alembic

Alembic - це бібліотека для Python, яка використовується для управління версіями баз даних і виконання міграцій. Вона розроблена для роботи з SQLAlchemy і допомагає розробникам ефективно керувати змінами в структурі баз даних. Alembic

дозволяє відстежувати, застосовувати та відкочувати зміни в схемі бази даних, що особливо важливо в середовищах, де база даних активно розвивається і змінюється.

Alembic забезпечує контроль версій бази даних, дозволяючи розробникам зберігати історію змін у схемі бази даних. Це дозволяє легко відстежувати, які зміни були зроблені, і повертатися до попередніх версій, якщо це необхідно. Така функціональність особливо корисна в командних проектах, де кілька розробників можуть одночасно працювати над різними частинами бази даних. [3]

Однією з головних переваг Alembic є можливість автоматичного генерування міграцій. Це означає, що бібліотека може автоматично створювати інструкції для змін у базі даних на основі змін у моделях SQLAlchemy. Це значно зменшує обсяг ручної роботи і знижує ймовірність помилок, пов'язаних з ручним створенням міграційних файлів.

Alembic спрощує процес управління базою даних, дозволяючи розробникам зосередитися на функціональності додатку, а не на технічних деталях зміни схеми бази даних. Автоматизація міграцій і простота використання роблять Alembic зручним інструментом як для невеликих проектів, так і для великих корпоративних додатків.

Глибока інтеграція Alembic з SQLAlchemy забезпечує зручність у використанні та максимальну ефективність. Розробники, які вже використовують SQLAlchemy для роботи з базою даних, можуть легко інтегрувати Alembic у свій робочий процес. Це забезпечує узгодженість між моделями додатку та схемою бази даних.

3.5. Фреймворк Flask-WTF

Flask-WTF - це розширення для веб-фреймворку Flask, яке забезпечує інтеграцію з бібліотекою WTForms, потужним інструментом для роботи з веб-формами. Це розширення спрощує процес створення, валідації та обробки форм у Flask-додатках, надаючи розробникам зручний та інтуїтивно зрозумілий інтерфейс для роботи з

формами. Flask-WTF поєднує функціональність WTForms з можливостями Flask, що робить роботу з формами більш ефективною та організовано. [3]

Основна функція Flask-WTF полягає у спрощенні створення та валідації форм. Замість того, щоб писати HTML-код для кожної форми вручну, розробники можуть визначати форми у вигляді класів Python, що значно полегшує процес розробки. Крім того, Flask-WTF автоматично генерує HTML для форм та забезпечує валідацію даних на стороні сервера, що підвищує безпеку додатку. Це особливо корисно для забезпечення коректності даних, які вводяться користувачами.

Однією з головних переваг Flask-WTF є його здатність інтегруватися з іншими компонентами Flask, такими як механізми автентифікації та управління сесіями. Це дозволяє легко додавати функціонал обробки форм до існуючих додатків без значних змін у коді. Також Flask-WTF підтримує захист від CSRF (Cross-Site Request Forgery) атак, що додає додатковий рівень безпеки для веб-додатків.

Flask-WTF також надає розширені можливості для кастомізації та розширюваності форм. Розробники можуть легко створювати власні валідатори, поля та обробники, що робить цей інструмент надзвичайно гнучким. Крім того, детальна документація та активна спільнота роблять Flask-WTF доступним для розробників різного рівня, від початківців до досвідчених професіоналів. Завдяки своїй функціональності та простоті використання, Flask-WTF є важливим інструментом для розробки веб-додатків на базі Flask.

3.6. Бібліотека Bcrypt

Bcrypt - це бібліотека для Python, яка використовується для безпечного хешування паролів. Вона є важливим інструментом у забезпеченні безпеки веб-додатків, оскільки дозволяє надійно зберігати паролі користувачів. Хешування за допомогою Bcrypt включає використання алгоритму, який робить паролі стійкими до атак, таких як перебір можливих паролів або використання таблиць перетворень (rainbow tables).

Основна функція Bcrypt полягає в тому, щоб забезпечити сильне криптографічне хешування, що ускладнює для зломисників отримання справжніх паролів навіть у разі компрометації бази даних. На відміну від простих хеш-функцій, Bcrypt використовує соль (salt), що додається до кожного пароля перед хешуванням. Це робить кожен хеш унікальним, навіть для однакових паролів, і значно підвищує рівень безпеки.

Переваги використання Bcrypt включають його стійкість до сучасних методів атак і можливість налаштування складності хешування, що дозволяє збільшувати безпеку в міру зростання обчислювальних потужностей. Крім того, Bcrypt автоматично обробляє додавання солі та багатократне хешування, що знижує ризик помилок при імплементації безпечного зберігання паролів. Завдяки цим перевагам, Bcrypt є популярним вибором серед розробників для забезпечення надійного захисту користувацьких паролів у веб-додатках.

3.7. Бібліотека Bootstrap

Bootstrap є дуже популярною та розповсюдженою CSS бібліотекою. Вона значним чином спрощує для розробників створення адаптивних веб-сторінок. Фактично, ця бібліотека є стандартом розробки великої кількості веб-застосунків завдяки своїй зрозумілості використання та багатьох вбудованих компонентів.

В цій дипломній роботі використовувалася версія бібліотеки Bootstrap 3.

Основний файл цієї бібліотеки bootstrap.css містить стилі, що форматують основні елементи HTML розмітки сторінки, такі як: кнопки, форми, таблиці, навігаційні елементи, заголовки, підзаголовки тощо. [4]

Bootstrap 3 використовує 12-колонкову систему, яка дозволяє створювати гнучкі макети, які здатні адаптуватися до різних екранів. Певні класи дозволяють регулювати кількість колонок відповідно до екранного розміру. Це сприяє гнучкості та простоті в організації контенту.

Bootstrap 3 також представляє набір класів для форматування тексту, включаючи заголовки, абзаци, списки та інші текстові елементи. Включені також класи для вирівнювання тексту, зміни шрифту та інших текстових параметрів.

Форми в Bootstrap 3 мають спеціальні стилі для покращення їх вигляду та зручності. Включені класи для текстових полів, форм вводу, кнопок та інших формованих елементів. Це допомагає створювати горизонтальні та вертикальні форми, форми з вбудованими полями та іншими елементами.

Bootstrap 3 використовує принципи адаптивного дизайну, що дозволяє веб-ресурсам змінювати форму та вигляд відповідно до різних екранів. [5]

3.8. Система контролю версій Git

Git – це система контролю версій, що дозволяє розробникам відстежувати етапи розробки, зберігати здійснені зміни, вести розробку декільком розробникам та додавати зміни у різні гілки розробки, зберігати файли проєкту. Кожна здійснена зміна записується у систему, із зазначенням користувача, який зробив таку зміну та часу. Git працює шляхом створення так званих комітів у базі даних, що ґрунтується на принципі відгалуження та злиття гілок розробки.

Під час створення комітів використовуються криптографічні методи, що пов'язує цифрові підписи користувачів зі зробленими ними змінами.

Всі зміни обробляються локально на комп'ютері розробника, після чого завантажуються у відповідний репозиторій (сховище).

Під час розробки використовувався широко поширений сервіс контролю версій GitHub (github.com). Кожен користувач який має обліковий запис може створити репозиторій, у якому буде збережено сам проєкт, його версії, зміни тощо. Такий репозиторій може бути як закритим – тобто зміни в нього може вносити лише автор або дозволені автором користувачі, так і відкритим – в такому випадку зміни вносити може будь-який користувач.

4 ОПИС ПРОГРАМНОЇ РЕАЛІЗАЦІЇ

В цьому розділі описано програмну реалізацію розробленого веб-застосунку, що містить в собі огляд наступних пунктів:

- огляд створення сторінок сайту;
- огляд загальної логіки роботи сайту
- огляд реалізації обчислень;
- огляд процесу обліку особистих облікових записів;
- огляд бази даних.

4.1. Структура та архітектура системи обліку особистого бюджету

Цей веб-застосунок містить такі підсистеми:

- модуль обліку доходів та витрат - призначений для введення та збереження інформації про доходи та витрати користувача;
- модуль управління категоріями - дозволяє користувачам налаштовувати та редагувати категорії витрат для кращого аналізу фінансів;
- модуль обрахунків - виконує розрахунки різниці між надходженнями та витратами та облік залишку коштів.
- модуль управління обліковим записом – надає користувачеві можливість керувати власним обліковим записом, його даними, як-от пароль, логін та інші.

Для наочності та розуміння папка з проєктом поділена на логічні частини, які розглянуто далі. В корені папки знаходяться;

- `readme.md` – файл з інструкцією по запуску застосунка;
- `requirements.txt` – текстовий файл, що містить список бібліотек мови python, що необхідно встановити для коректної роботи сайту;
- `app.py` – код python, що запускає сайт;

- папка `‘.venv’` – містить файли створені віртуальним середовищем `python`, в якій міститься локальний інтерпретатор мови програмування та завантажені бібліотеки;
- папка `‘migrations’` – містить все що стосується міграцій баз даних;
- папка `‘app’` – основна папка проєкту, містить основний код сайту, базу даних та код сторінок.

Створивши статичні сторінки, виникає необхідність якимось чином їх ініціалізувати в коді та додати на локальний сервер. Робиться це у файлі `./app/main/routes.py`. Файл `«routes.py»` у `Flask` відповідає за визначення маршрутів (`routes`) у веб-застосунку. [6] Основні завдання `«routes.py»` включають:

1. визначення `url`-шляхів - визначається, які `url`-шляхи (маршрути) застосунку відповідають на які функції. наприклад, `/home` може відповідати функції `home()`, а `/about` - функції `about()`;

2. обробка запитів - кожен маршрут може мати свою функцію, яка обробляє запити, надіслані на цей маршрут. у цих функціях виконуються різноманітні завдання, такі як відображення сторінок, обробка даних форм, виконання операцій у базі даних тощо;

3. взаємодія з користувачем - файл `routes.py` відповідає за взаємодію з користувачем через веб-інтерфейс. він визначає, як користувачі можуть навігувати по сторінках застосунку та які операції вони можуть виконувати.

Наприклад, типовий вміст файлу `«routes.py»` може включати визначення маршрутів для домашньої сторінки, сторінки «про нас», реєстрації та входу користувача, а також функції для обробки даних, надісланих з веб-форм. Файл `«routes.py»` є важливою частиною структури веб-застосунку на `Flask`, оскільки він визначає основну логіку та функціональність застосунку.

Розглянемо заповнення форм користувацького вводу. Файл `«users/forms.py»` у `Flask` використовується для визначення форм, які пов'язані з функціоналом користувачів у веб-застосунку. Основні завдання `«users/forms.py»` включають:

1. визначення класів форм - в цьому файлі визначаються класи форм, які використовуються для взаємодії з користувачами. ці класи можуть містити поля для введення даних, такі як ім'я, електронна пошта, пароль тощо;

2. валідація даних - форми можуть включати правила валідації, які перевіряють коректність введених користувачем даних. наприклад, перевірка на наявність користувача з таким ім'ям користувача або електронною поштою, перевірка пароля на мінімальну довжину тощо;

3. зручний доступ до даних - використання форм дозволяє зберігати та обробляти дані, які вводяться користувачами, у зручному для подальшого використання форматі;

4. підтримка wtforms - у більшості веб-додатків на flask використовується бібліотека wtforms для роботи з формами. файл «users/forms.py» дозволяє організувати роботу з формами відповідно до стандартів, які встановлені в wtforms.

Отже, файл «users/forms.py» є важливою складовою частиною структури веб-додатка на Flask, оскільки він визначає, які дані можуть вводити користувачі у формах та як ці дані обробляються. Розглянемо це на нашому прикладі: Цей код визначає класи форм для реєстрації, входу, оновлення облікового запису, запиту на скидання пароля та скидання пароля у веб-застосунку на Flask. Основні елементи коду включають:

1. «RegistrationForm» - форма для реєстрації нового користувача. Включає поля для введення імені користувача, електронної пошти, пароля та підтвердження пароля;

2. «LoginForm»- форма для входу в систему. Містить поля для введення електронної пошти та пароля;

3. «UpdateAccountForm» - форма для оновлення інформації про обліковий запис користувача, таких як ім'я користувача та електронна пошта;

4. «RequestResetForm» - форма для запиту на скидання пароля. Містить поле для введення електронної пошти;

5. «ResetPasswordForm» - форма для скидання пароля. Включає поля для введення нового пароля та підтвердження нового пароля.

Кожна форма використовує різні валідатори, такі як «DataRequired», «Length», «Email» та «EqualTo», для перевірки коректності введених даних. Крім того, у кожній формі також визначені методи «validate_email» та «validate_username», які перевіряють, чи вже існує користувач з введеним ім'ям користувача чи електронною поштою. Ці методи використовуються для забезпечення унікальності ім'я користувача та електронної пошти при реєстрації нового користувача або оновленні інформації про обліковий запис.

Розглянемо процес реєстрації. Єдиним, що у цьому процесі є відмінним від інших форм та потребує уваги це зберігання паролів. Для надійного зберігання паролів їх необхідно записувати у базу даних у зашифрованому вигляді, аби у випадку компрометації бази даних, зловмисники не могли отримати доступ до облікових записів користувачів. Для цього використовуються хеш-функції. Пакетом, що відповідає за шифрування паролів за допомогою хеш-функцій є Flask-BCrypt. BCrypt - це функція хешування паролів. Розширення Flask-BCrypt інтегрує BCrypt з додатками Flask, забезпечуючи просту у використанні функцію хешування та перевірки паролів.

Flask-BCrypt використовує алгоритм хешування bcrypt для шифрування паролів. BCrypt - це популярна і широко використовувана функція хешування паролів, яка розроблена таким чином, щоб бути обчислювально інтенсивною і повільною, що робить її стійкою до атак грубої сили. Він використовує криптографічну хеш-функцію, засновану на шифрі Blowfish.

Основна логічна частина застосунку працює за допомогою мови програмування Python. Розглянемо приклад обробки запитів користувача на прикладі обробки форми користувацького вводу у файлі «./app/forms.py».

Цей код містить визначення кількох форм, які використовуються у веб-застосунку для управління особистим бюджетом. Він використовує Flask-WTF для створення та управління формами, а також WTForms для валідації та обробки даних.

Розглянемо як це працює:

1. імпорти - імпортуються необхідні модулі та бібліотеки для роботи з датами, формами, валідацією та користувачами;
2. форма реєстрації («registrationform»):

- поля: ім'я користувача, електронна пошта, пароль, підтвердження пароля.
- валідація: перевіряється, чи існує вже користувач з таким ім'ям або електронною поштою.

3. форма входу («loginform»):

- поля: електронна пошта, пароль, опція «запам'ятати мене».
- валідація: перевіряється наявність даних у полях.

4. форма витрат («spendingform»):

- поля: дата, назва товару, кількість, тип кількості (л, мл, кг, г, шт), сума витрат.
- валідація: перевіряється, чи всі поля заповнені коректно.

5. форма планування місяця («monthplanform»):

- поля: місяць, дохід.
- валідація: перевіряється, чи вже існує план для обраного місяця.

6. форма типу місяця («monthtypeform»):

- поля: назва типу, вибір типу суми (відсоток чи гроші), сума, параметри «за замовчуванням» та «кожен день».
- валідація: забезпечується коректність введених даних.

Кожна форма визначає набір полів з відповідними типами даних та валідаторами. При поданні форми, дані перевіряються на коректність. Якщо дані не відповідають вимогам валідації, користувачу повертається помилка. Якщо дані коректні, вони обробляються та зберігаються у базі даних.

Більш детально як це працює:

1. реєстрація - користувач заповнює форму реєстрації, вводячи ім'я, електронну пошту та пароль. якщо введена електронна пошта або ім'я користувача вже існують, виводиться помилка;

2. вхід - користувач вводить електронну пошту та пароль для входу. якщо дані коректні, користувач авторизується у системі;

3. витрати - користувач заповнює форму витрат, вводячи інформацію про покупку. дані зберігаються для подальшого аналізу;

4. планування місяця - користувач планує бюджет на місяць, вказуючи дохід. якщо для цього місяця вже існує план, виводиться помилка;

5. типи місяця - користувач створює або редагує типи місяця, вказуючи назву та суму. ці дані використовуються для детального планування бюджету.

4.2. Функціональність системи обліку особистого бюджету

Для потенційного користувача передбачено такі функціональні можливості як:

- реєстрація на сайті та ведення особистого облікового запису, без якого використання застосунку є неможливим;
- здійснення скидання паролю від облікового запису, шляхом отримання посилання для цієї дії на вказану при реєстрації електронну пошту;
- створювати місячні плани витрат та переглядати їх на графіках;
- додавати окремі витрати, що не передбачені у загальному місячному плані
- переглядати статистику за витратами у таблиці та на графіках.

Загалом, можливості користувача розглянути можливо на рисунку 4.1.

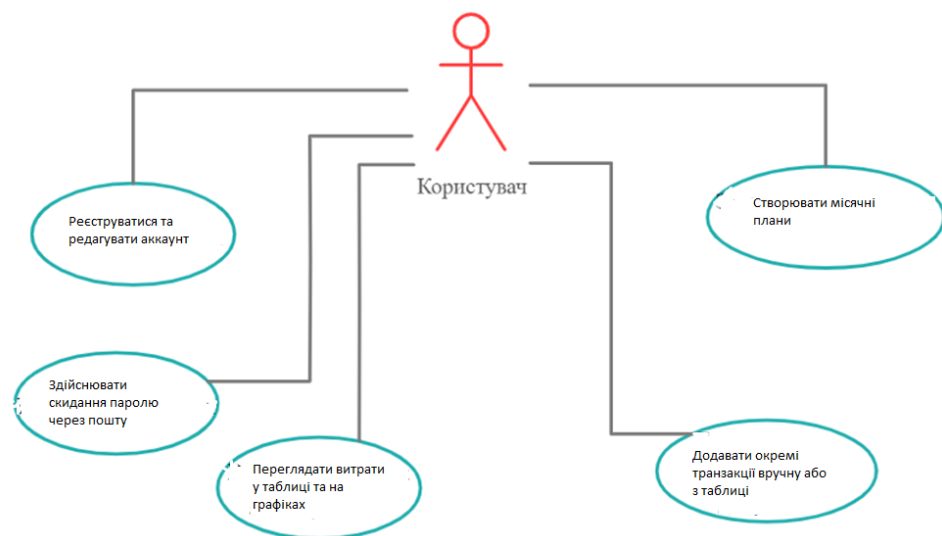


Рисунок 4.1. – Діаграма прецедентів

4.3. Схема бази даних

Розглянемо структуру бази даних. Створена база має такі таблиці (рисунок 4.2)

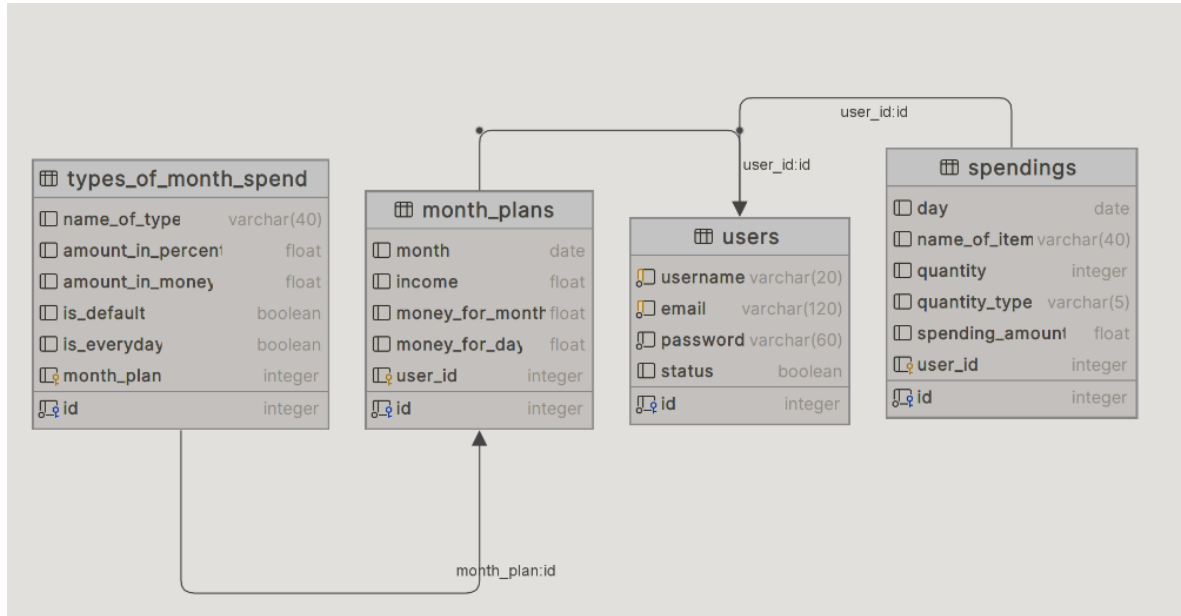


Рисунок 4.2 – Схема бази даних

1. `alembic_version` - таблиця у базі даних Flask з назвою «`alembic_version`» відповідає за збереження версії Alembic, яка використовується для керування міграціями бази даних. [7] У цій таблиці зазвичай зберігається один запис, який містить інформацію про поточну версію схеми бази даних. [8] Alembic використовує цю таблицю для визначення поточної версії та порівняння її з версіями міграцій, щоб автоматично визначити, які міграції потрібно застосувати до бази даних для оновлення схеми до поточної версії;

2. `month_plans` – таблиця, що містить місячні плани, створені користувачами. Ключем таблиці є `id` який є унікальним для кожного місячного плану. Таблиця має наступні поля:

- 1) ідентифікатор;
- 2) дату транзакції;
- 3) дохід в цю дату;

- 4) гроші на місяць;
- 5) гроші на день;
- 6) ідентифікатор користувача, якому належить цей план (рисунок 4.3).

	id	month	income	money_for_month	money_for_day	user_id
1	1	2021-09-22	35742.42	9072.83	26669.59	1
2	2	2022-04-14	1	0.2	0.8	1
3	3	2022-04-16	1500	0	1500	6
4	4	2022-04-16	100	46	54	5
5	5	2022-04-16	250	0	250	7
6	6	2024-04-22	1000	50	950	9

Рисунок 4.3 – Приклад заповнення таблиці month_plans

3. spendings - таблиця, що містить список транзакцій користувачів. Ключем є id, містить: дату, назву, кількість, одиниці вимірювання, суму, id користувача, якому належить транзакція (рисунок 4.4).

	id	day	name_of_item	quantity	quantity_type	spending_amount	user_id
1	1	2024-04-25	Молоко	1000	мл	40	9
2	2	2024-04-25	Яблука	2000	г	80	9
3	3	2024-04-01	Сік	1000	мл	50	9
4	4	2024-04-02	Хліб	1	шт	33	9
5	5	2024-04-03	Морозиво	1	шт	35	9
6	6	2024-04-05	Кава з круасаном	1	шт	60	9

Рисунок 4.4 – Приклад заповнення таблиці spendings

4. sqlite_master – У Flask, як і в будь-якому іншому веб-фреймворку, таблиця sqlite_master належить до бази даних SQLite, яка використовується за замовчуванням у режимі розробки або для простих проєктів.

Таблиця sqlite_master є системною таблицею, яка містить метадані про всі об'єкти бази даних SQLite, такі як таблиці, індекси, представлення, тригери тощо. У цій таблиці зберігаються назви, типи та інші важливі характеристики об'єктів бази даних.

5. types_of_month_spend - таблиця, що типи транзакцій користувачів. Ключем унікальний id, містить: назву витрати, процентне співвідношення, суму, прапорець чи є ця витрата щоденною, та foreign key до якого місячного плану вона відноситься (рисунок 4.5)

id	name_of_type	amount_in_percent	amount_in_money	is_default	is_everyday	month_plan
1	invest	10	3574.24	0	0	1
2	expenses	7.69	2750	0	0	1
4	Веселощі	20	20	0	1	4
5	Телефон	46	46	0	0	4
6	Молоко	10	0.1	0	0	2
7	Молоко	10	0.1	0	0	2
8	Їжа	1	10	0	1	6
9	Комуналка	5	50	0	0	6

Рисунок 4.5 – Приклад заповнення таблиці type_of_month_spend

6. users – таблиця, яка містить список всіх користувачів. Містить: id, ім'я користувача, електронну пошту, хеш пароля, та статус (zareestrovaniy чи ni) (рисунок 4.6)

id	username	email	password	status
1	9 kelok	admin@admin.com	\$2b\$12\$qSTQ1J3jj1cU.EpXzkiv4u2jT/z6rE1rckbeeo...	0

Рисунок 4.6 – приклад заповнення таблиці users

Отже, база даних сконфігурована таким чином, аби легко й швидко робити пошук даних в системі, виділяти якому користувачеві які записи належать та надійно зберігати конфіденційність системи, завдяки тому, що паролі облікових записів зберігаються у зашифрованому вигляді.

5 РОБОТА КОРИСТУВАЧА З ПРОГРАМНОЮ СИСТЕМОЮ

Після опису програмної реалізації перейдемо до взаємодії користувача із системою, що охоплює:

- опис системних вимог;
- інсталяція та запуск застосунку;
- інструкція для користувача;
- приклади результатів роботи.

5.1. Опис системних вимог

Для роботи користувача з розробленою програмною системою йому потрібні мінімальні потужності апаратного забезпечення (вимоги наведені в таблиці 5.1)

Таблиця 5.1. Вимоги до апаратного забезпечення клієнта

Пристрій	Характеристика
Процесор	Intel ® Core ™ 2 / 2 Duo / Pentium ® / Celeron ® / Xeon™ / i3 / i5 / i7 чи AMD 6 / Turion ™ / Athlon ™ / Duron ™ / Sempron ™ з тактовою частотою не нижче 1.5 GHz.
Оперативна пам'ять (RAM – Random Access Memory)	Рекомендовано не менше 2 RAM
Швидкість з'єднання з інтернет	Рекомендовано не менше 128 кб/сек

Підтримувані апаратні архітектури:

- 32-розрядна (x86);
- 64-розрядна (x64).

Нижче розглянемо вимоги до апаратного та програмного забезпечення комп'ютера, який буде виконувати функції сервера, на якому буде розгорнуто програмний застосунок (вимоги наведено в таблиці 5.2 та в таблиці 5.3 відповідно).

Таблиця 5.2. Вимоги до апаратного забезпечення сервера

Пристрій	Характеристика
Процесор	Intel ® Core [™] / i3 / i5 / i7 чи AMD K5 [™] / K6 [™] / K7 [™] з тактовою частотою не нижче 3.0 GHz.
Оперативна пам'ять (RAM – Random Access Memory)	Рекомендовано не менше 4 RAM
Швидкість з'єднання з інтернет	Рекомендовано не менше 512 кб/сек
Вільне місце на жорсткому диску	Рекомендовано 50 гб

5.2. Інсталяція

Для того, щоб інстальювати застосунок необхідно встановити наступні компоненти:

- Microsoft Visual Studio Build Tools та його компоненти
- Visual C++
- Python 3.10.2 та його бібліотеки
- PyCharm Professional

Для встановлення Microsoft Visual Studio Build Tools необхідно перейти за посиланням на сайт Microsoft <https://visualstudio.microsoft.com/downloads/> та встановити останню актуальну версію програми (рисунок 5.1).

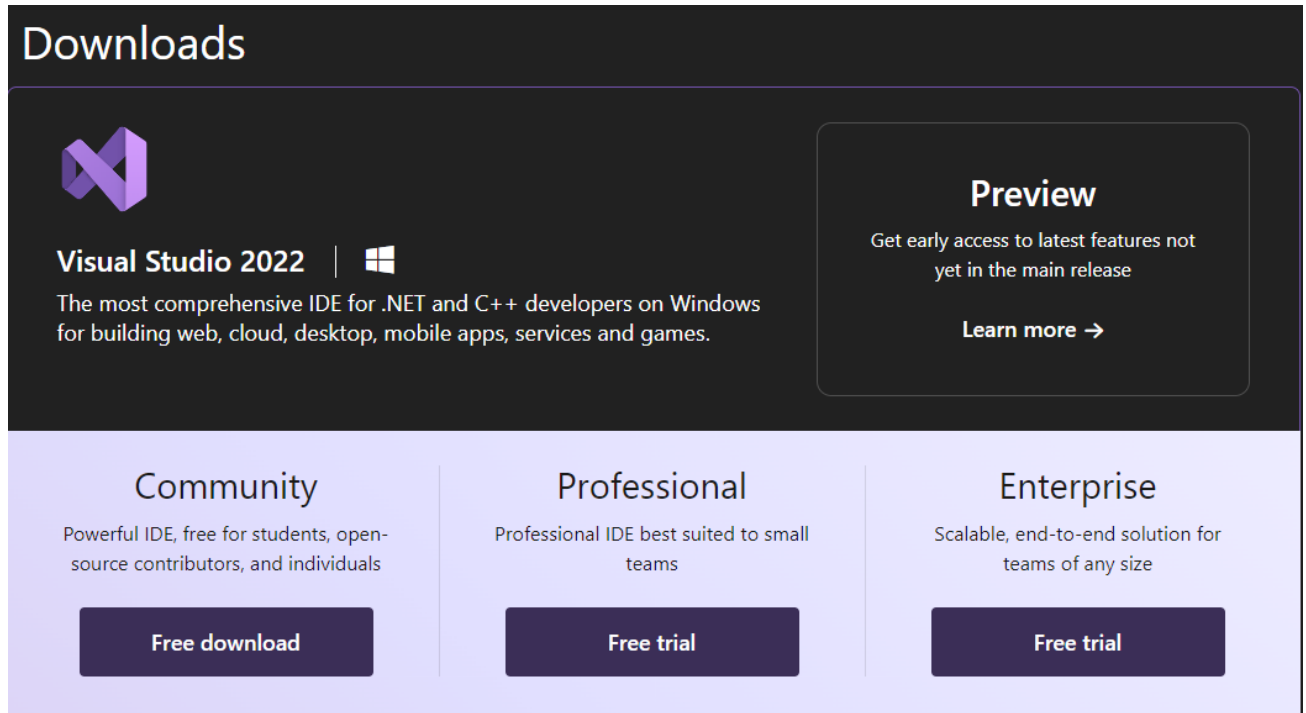


Рисунок 5.1 – Сторінка завантаження Microsoft Visual Studio

Відкривши завантажений файл потрапляємо у вікно вибору компонентів, серед яких необхідно вибрати для встановлення:

- Windows Universal C Runtime;
- MSVC v143 – VS 2022 C++ x64 build tools;
- Windows 10 SDK+;
- Windows Universal CRT SDK (рисунок 5.2).

Для встановлення наступного компоненту - Visual C++ необхідно перейти на сайт розробника <https://visualstudio.microsoft.com/visual-cpp-build-tools/> та завантажити останню актуальну версію (рисунок 5.3).

Для встановлення Python 3.10.2 перейдемо на сайт розробника <https://www.python.org/downloads/release/python-3102/> та вибрати для завантаження

відповідну версію програми. Під час встановлення обов'язково поставити галочку додати python.exe to PATH (рисунок 5.4).

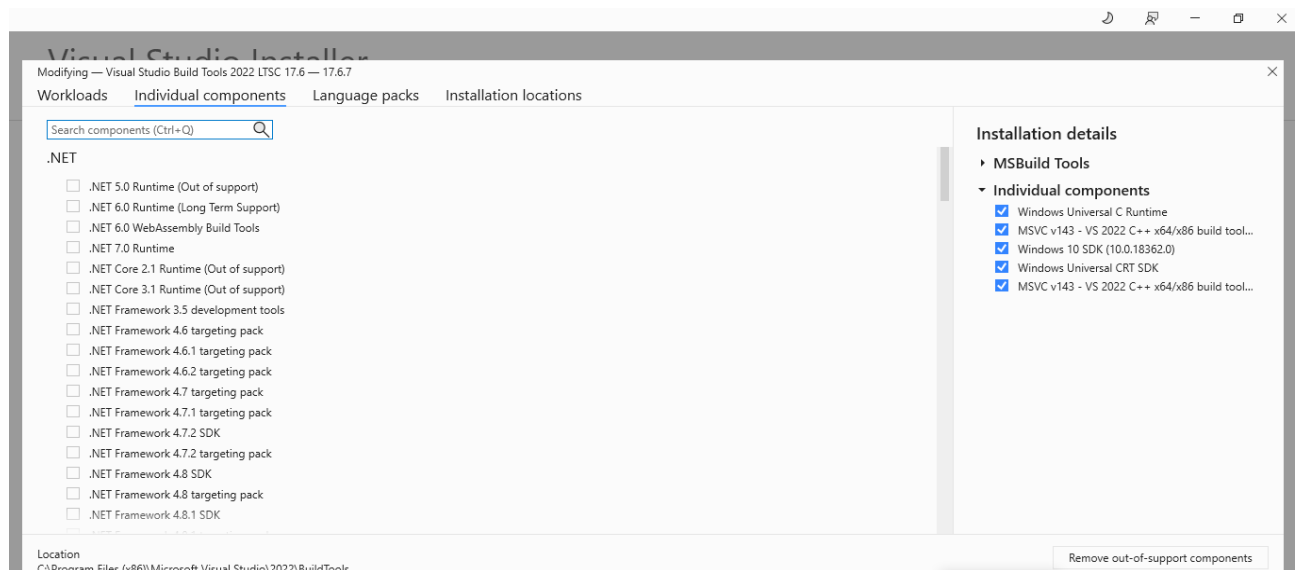
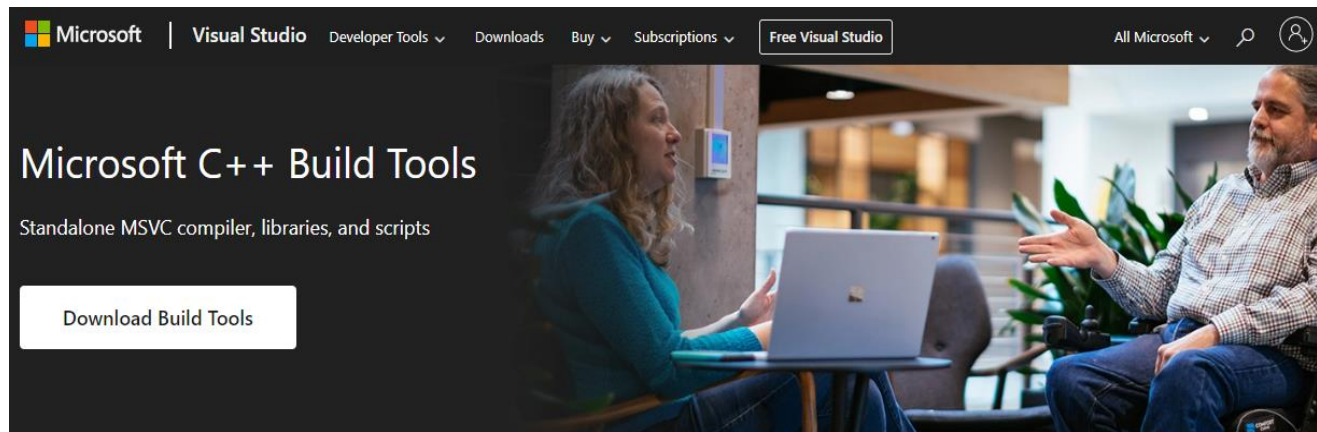


Рисунок 5.2 – Інсталяція Microsoft Visual Studio Build Tools



The Microsoft C++ Build Tools provides MSVC toolsets via a scriptable, standalone installer without Visual Studio. Recommended if you build C++ libraries and applications targeting Windows from the command-line (e.g. as part of your continuous integration workflow). Includes tools shipped in Visual Studio 2015 Update 3, Visual Studio 2017, Visual Studio 2019, and latest version of Visual Studio 2022.

Рисунок 5.3 – Сторінка завантаження MS Visual C++

Після цього необхідно встановити середовище розробки PyCharm з офіційного сайту розробника JetBrains <https://www.jetbrains.com/pycharm/> (рисунок 5.5).

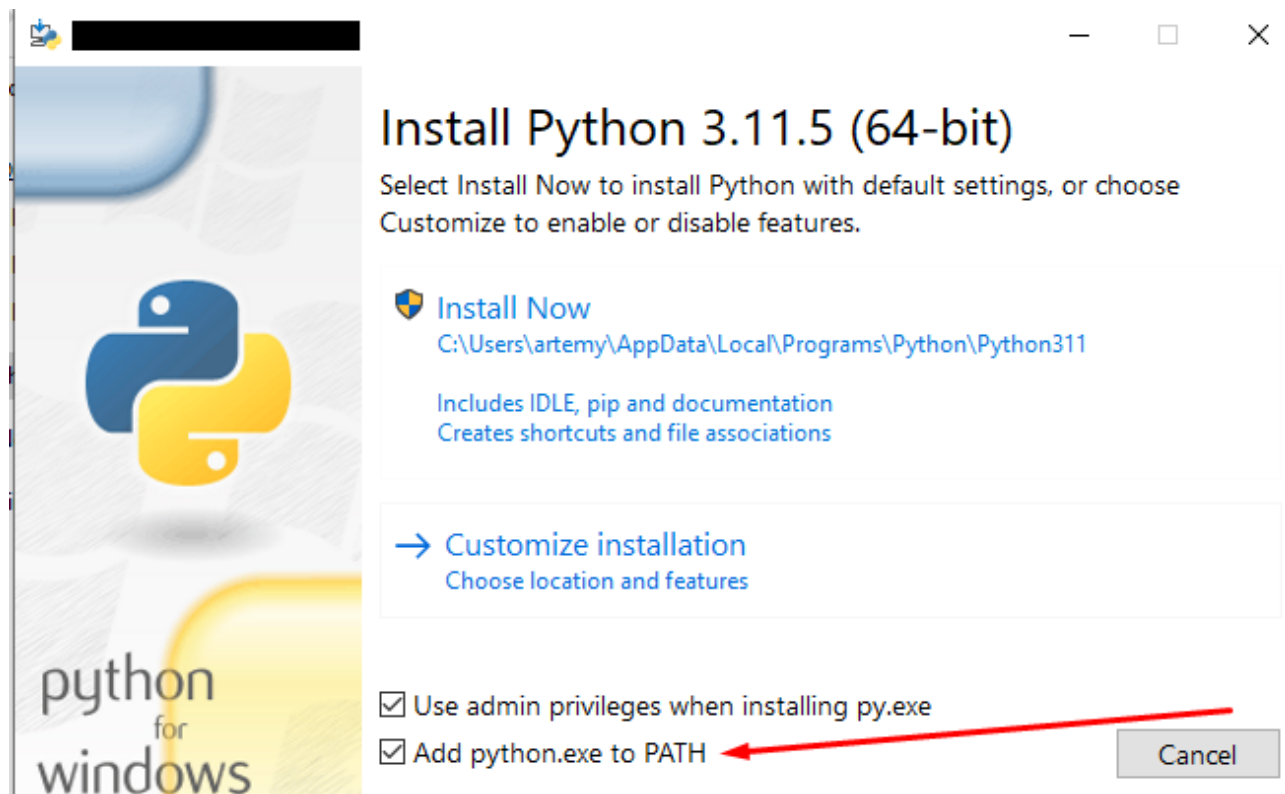


Рисунок 5.4 – Сторінка встановлення інтерпретатора мови програмування Python



Рисунок 5.5 - Сторінка завантаження PyCharm

Зрештою, необхідно завантажити безпосередньо сам веб-застосунок з GitHub репозиторію <https://github.com/karuk-artemii-tr/daytomoney>. Папку з кодом необхідно відкрити у PyCharm (рисунок 5.6).

При виконанні всіх наступних команд у консолі передбачається що всі вони будуть виконуватись безпосередньо в головній директорії з кодом програми.

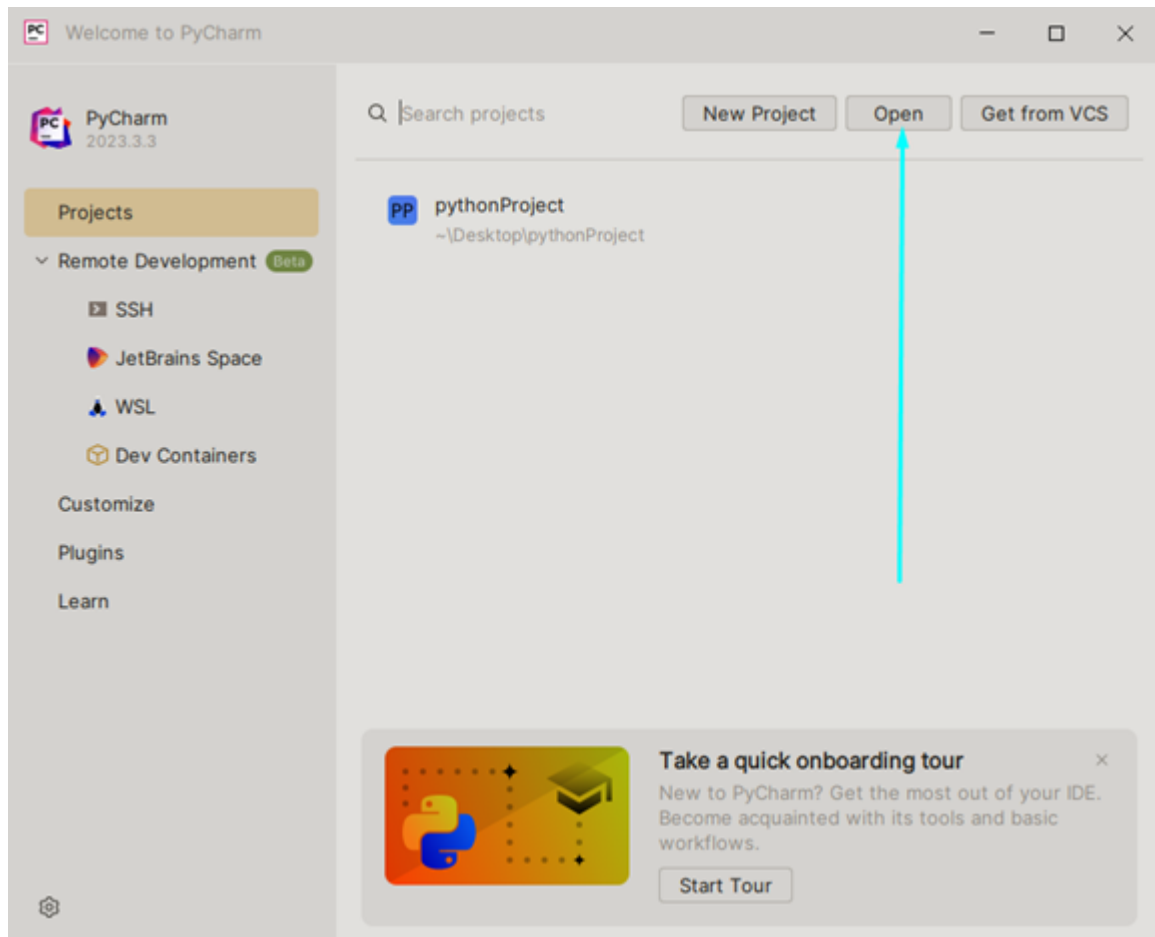


Рисунок 5.6 – Вікно PyCharm

Відкривши проєкт впевнимся що вибрано коректний інтерпретатор Python 3.10.2, після чого через консоль маємо створити віртуальне середовище, за допомогою команди:

```
python -m venv venv
```

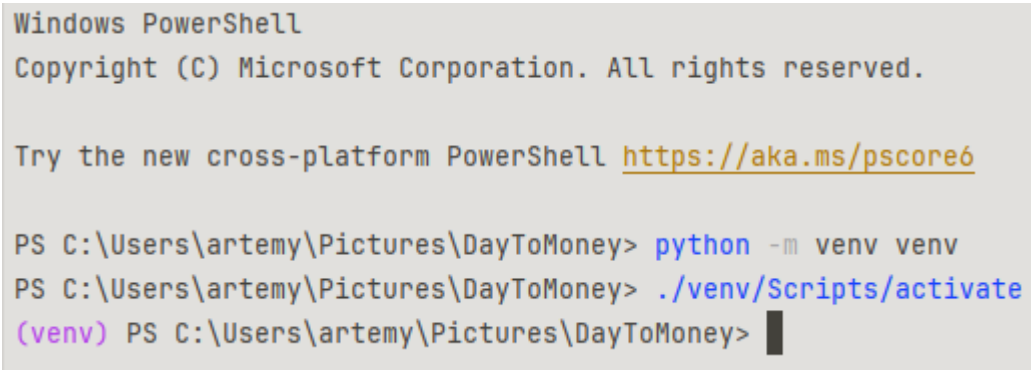
Для того, щоб зайти в створене віртуальне середовище скористаємося командою

./venv/Scripts/activate (рисунок 5.7).

Після успішного створення віртуального середовища встановимо всі необхідні для виконання програми бібліотеки, за допомогою команди:

pip install -r requirements.txt

Ця команда виконає читання списку необхідних для встановлення бібліотек та завантажить їх до директорії проєкту у віртуальне середовище (рисунок 5.8).



```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\Users\artemy\Pictures\DayToMoney> python -m venv venv
PS C:\Users\artemy\Pictures\DayToMoney> ./venv/Scripts/activate
(venv) PS C:\Users\artemy\Pictures\DayToMoney> █
```

Рисунок 5.7 – Створення віртуального середовища



```
Collecting pycparser==2.20
  Using cached pycparser-2.20-py2.py3-none-any.whl (112 kB)
Collecting python-dateutil==2.8.2
  Using cached Werkzeug-2.0.1-py3-none-any.whl (288 kB)
Collecting WTForms==2.3.3
  Using cached WTForms-2.3.3-py2.py3-none-any.whl (169 kB)
Requirement already satisfied: setuptools>=3.0 in c:\users\artemy\pictures\daytonemoney\venv\lib\site-packages (from gunicorn==20.1.0->-r requirements.txt (line 18)) (58.1.0)
Installing collected packages: MarkupSafe, colorama, Werkzeug, pycparser, Jinja2, itsdangerous, greenlet, click, SQLAlchemy, six, Mako, Flask, cffi, WTForms, pytz, python-dateutil, numpy, idna, Flask-SQLAlchemy, dnspython, blinker, bcrypt, alembic, tzdata, python-dotenv, pycpg2-binary, pycpg2, Pillow, pandas, gunicorn, Flask-WTF, Flask-Migrate, Flask-Mail, Flask-Login, Flask-Script, email-validator, certifi
Successfully installed Flask-2.0.1 Flask-Bcrypt-0.7.1 Flask-Login-0.5.0 Flask-Mail-0.9.1 Flask-Migrate-3.1.0 Flask-SQLAlchemy-2.5.1 Flask-WTF-0.15.1 Jinja2-3.1.1 Mako-1.2.0 MarkupSafe-2.1.1 Pillow-9.1.0 SQLAlchemy-1.4.20 WTForms-2.3.3 Werkzeug-2.0.1 alembic-1.7.7 bcrypt-3.2.0 blinker-1.4 certifi-2021.5.30 cffi-1.15.0 click-8.0.1 colorama-0.4.4 dnspython-2.2.1 email-validator-1.1.3 greenlet-1.1.2 gunicorn-20.1.0 idna-3.3 itsdangerous-2.0.1 numpy-1.22.3 pandas-1.4.1 psycpg2-2.9.1 psycpg2-binary-2.9.3 pycparser-2.20 python-dateutil-2.8.2 python-dotenv-0.20.0 pytz-2022.1 six-1.16.0 tzdata-2022.1
WARNING: You are using pip version 21.3.4; however, version 24.0 is available.
You should consider upgrading via the 'C:\Users\artemy\pictures\DayToMoney\venv\Scripts\python.exe -m pip install --upgrade pip' command.
(venv) PS C:\Users\artemy\pictures\DayToMoney>
```

Рисунок 5.8 – Успішне встановлення бібліотек

Встановивши необхідні бібліотеки, маємо налаштувати базу даних. [9] Зробимо це за допомогою описаних далі команд, що використовуються з Flask-Migrate -

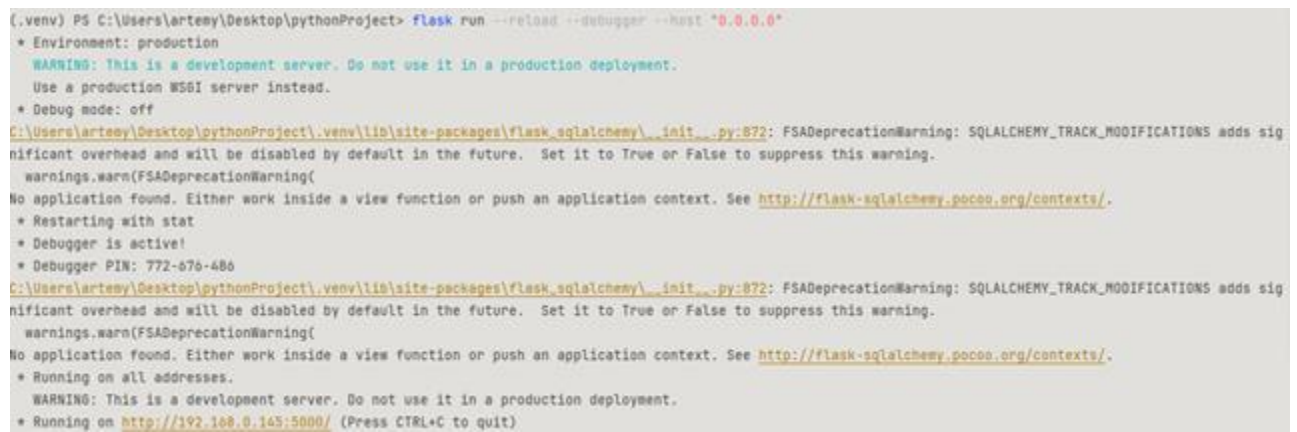
розширенням для Flask, яке дозволяє керувати міграціями баз даних у веб-застосунках Flask:

1. «flask db init» - ця команда ініціалізує структуру міграцій для бази даних. Вона створює каталог, де будуть зберігатися міграції, та інші необхідні файли;
2. «flask db migrate» - ця команда створює нову міграцію на основі поточного стану моделей даних у вашому застосунку. Вона аналізує зміни в моделях та генерує скрипт міграції, який описує необхідні зміни в базі даних;
3. «flask db upgrade» - ця команда застосовує накопичені міграції до бази даних. Вона виконує усі зміни, описані у міграціях, які ще не були застосовані до бази даних, і оновлює структуру бази даних відповідно до останньої міграції.

Нарешті, налаштувавши базу даних та встановивши всі необхідні бібліотеки, можемо запустити веб-сайт. Зробимо це за допомогою наступної команди:

flask run --reload --debugger --host «0.0.0.0»

Після чого побачимо повідомлення про успішний запуск застосунку на локальному сервері (рисунок 5.9)



```
(.venv) PS C:\Users\artemy\Desktop\pythonProject> flask run --reload --debugger --host "0.0.0.0"
* Environment: production
  WARNING: This is a development server. Do not use it in a production deployment.
  Use a production WSGI server instead.
* Debug mode: off
C:\Users\artemy\Desktop\pythonProject\.venv\lib\site-packages\flask_sqlalchemy\__init__.py:872: FSADeprecationWarning: SQLAlchemy_TRACK_MODIFICATIONS adds significant overhead and will be disabled by default in the future. Set it to True or False to suppress this warning.
  warnings.warn(FSADeprecationWarning(
No application found. Either work inside a view function or push an application context. See http://flask-sqlalchemy.pocoo.org/contexts/.
* Restarting with stat
* Debugger is active!
* Debugger PIN: 772-676-486
C:\Users\artemy\Desktop\pythonProject\.venv\lib\site-packages\flask_sqlalchemy\__init__.py:872: FSADeprecationWarning: SQLAlchemy_TRACK_MODIFICATIONS adds significant overhead and will be disabled by default in the future. Set it to True or False to suppress this warning.
  warnings.warn(FSADeprecationWarning(
No application found. Either work inside a view function or push an application context. See http://flask-sqlalchemy.pocoo.org/contexts/.
* Running on all addresses.
  WARNING: This is a development server. Do not use it in a production deployment.
* Running on http://192.168.0.145:5000/ (Press CTRL+C to quit)
```

Рисунок 5.9 – Повідомлення про запуск сайту

В цьому випадку система автоматично створила для сайту локальний сервер та присвоїла йому адресу <http://192.168.0.145:5000/>.

Комп'ютер, на якому запущено сайт, є сервером і тому за цією адресою можливо зайти на сторінку з будь-якого пристрою, що підключений до однієї локальної мережі із цим комп'ютером.

Тобто, якщо комп'ютер, на якому запущено сайт, підключений до одного маршрутизатора (роутера) зі смартфоном або іншим комп'ютером, то на сайт за вищевказаною адресою можливо буде зайти з будь-якого з перелічених пристроїв.

Відкривши це посилання у будь-якому веб-переглядачі, наприклад – Google Chrome, потрапимо на створений сайт (рисунк 5.10).

На рисунку 5.10 бачимо головну сторінку розробленого веб-застосунку.

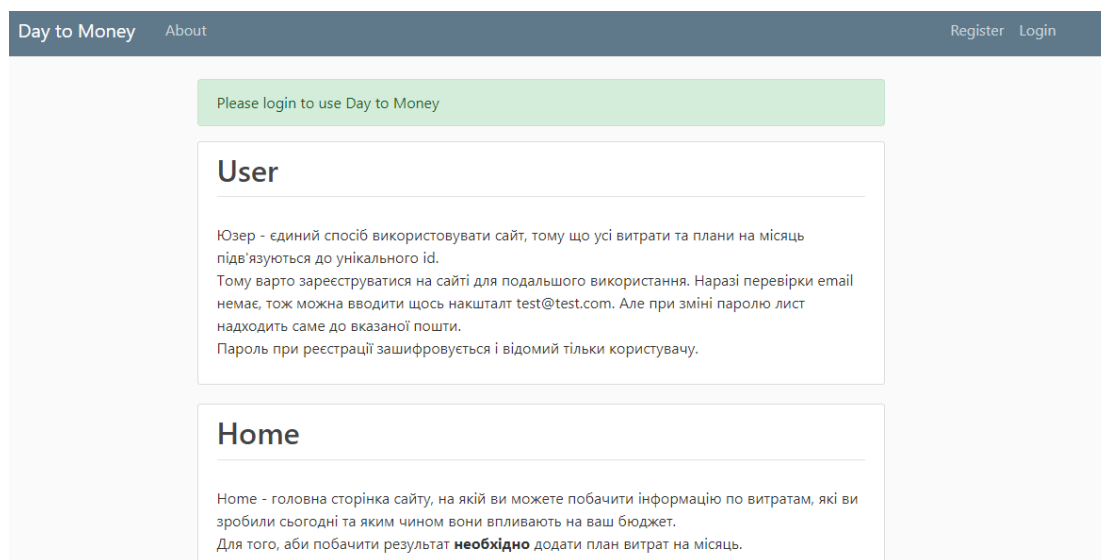


Рисунок 5.10 - Сторінка запущеного застосунку

5.3. Робота особистого облікового запису

Перш за все, будь-кому хто захоче користуватись застосунком треба буде зареєструватися.

5.3.1. Навігація по сторінках

Юзер (тобто зареєстрований користувач) - єдиний спосіб використовувати сайт, тому що усі витрати та плани на місяць підв'язуються до унікального ідентифікатора. Тому варто зареєструватися на сайті для подальшого використання. Пароль при реєстрації зашифровується і відомий тільки користувачу (рисунок 5.11).

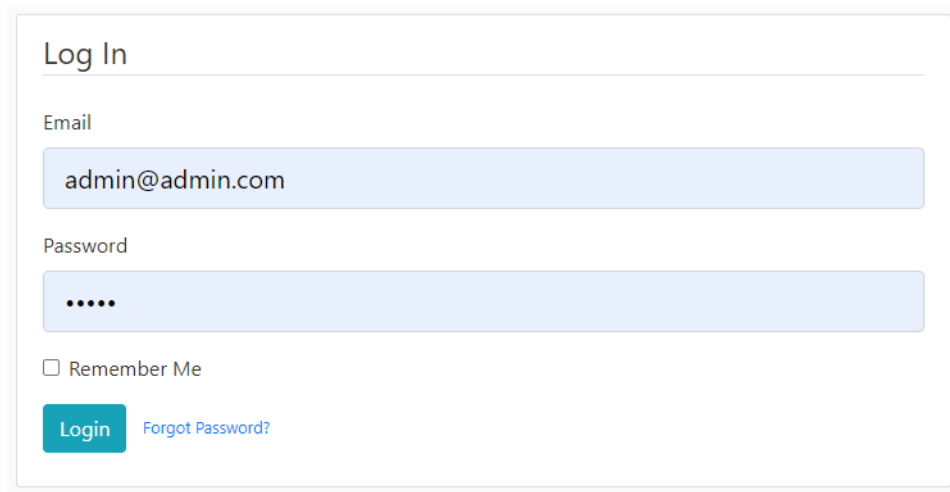


Рисунок 5.11 – Сторінка входу

Увійшовши до в обліковий запис бачимо надпис що немає жодних створених планів, тому система просить користувача створити новий план (рисунок 5.12)

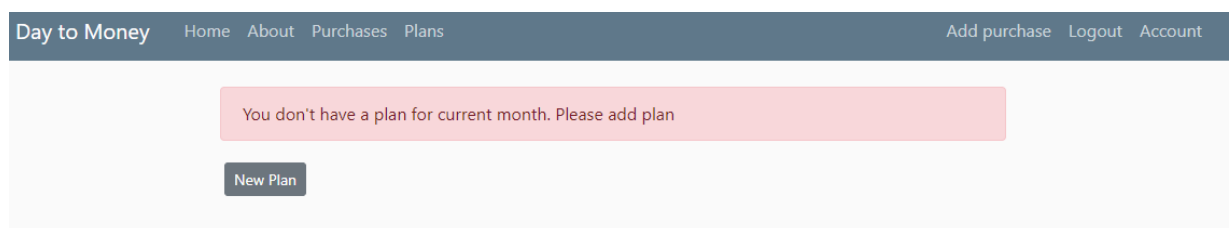


Рисунок 5.12 – Сторінка створення плану

На цьому етапі користувач може відкрити наступні вкладки:

- home;

- about;
- purchases;
- plans;
- add purchase;
- logout;
- account.

Home (домашня сторінка) - головна сторінка сайту, на якій ви можете побачити інформацію по витратам, які ви зробили сьогодні та яким чином вони впливають на ваш бюджет. Для того, аби побачити результат необхідно додати план витрат на місяць.

All plans (всі плани) - сторінка на якій користувач додає план витрат на місяць за допомогою кнопки New Plan Він обирає будь-яку дату в середині місяця для того, аби вибирає безпосередньо місяць та введіть ваш дохід або кількість грошей, яку він хоче розпланувати Користувач натискає на найменування місяця для подробиць.

Місячний план - Тут відображаються задані користувачем заплановані витрати на місяць та щоденне використання у абсолютному значенні та у відсотках до доходу. Гроші виділені на щоденні витрати враховуються при плануванні на сторінці Delta.

Користувач натискає на ім'я типу витрат для редагування. Додає типи запланованих витрат за допомогою кнопки New Type, вказує ім'я типу витрати, наприклад: комуналка, інвестування, кіно, їжа. Вказує тип запланованої витрати у відсотках від прибутку чи в абсолютному значенні. Вказує чи ця витрата запланована на щоденне використання, наприклад: їжа, кіно, транспорт. За замовчуванням, залишок незапланованих грошей враховується до щоденного використання у Delta (рисунок 5.13).

Дельта – Користувач натискає на дату для перегляду витрат за день. Result - сума витрат за день. Balance - кількість грошей виділена на щоденні витрати(Money) поділена на кількість днів, що залишилися до кінця місяця. Savings - кількість грошей, яка була збережена.

5.3.2. Створення місячного плану

Для створення нового місячного плану користувачеві треба вказати число, місяць, рік та суму першого доходу (рисунок 5.14).

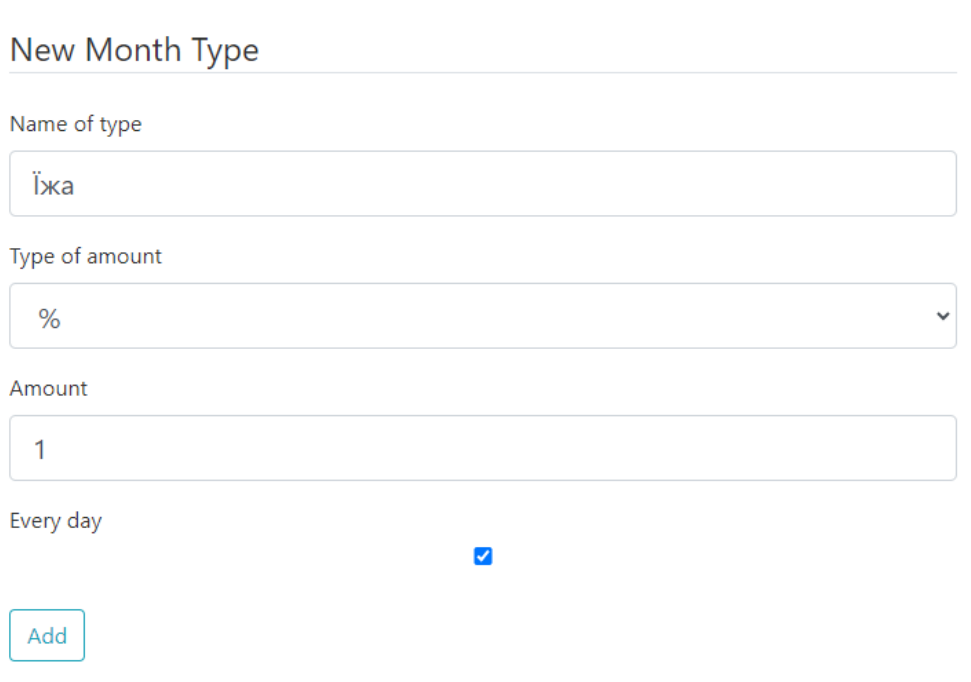
Створивши новий місячний план, бачимо його на сторінці створених планів.

На головному вікні відображається місяць, рік плану, залишок коштів та дельта (рисунок 5.15).

Створивши місячний план, користувач переходить до його заповнення (рисунок 5.16).

Після заповнення місячного плану користувач додає окремі витрати (рисунок 5.17).

На рисунку 5.18 бачимо приклад успішно доданої покупки та розрахунок кількості грошей, які було заплановано витратити, фактичні витрати та кількість зекономлених грошей відносно тої суми, яку було заплановано витратити.



New Month Type

Name of type

Їжа

Type of amount

%

Amount

1

Every day ☒

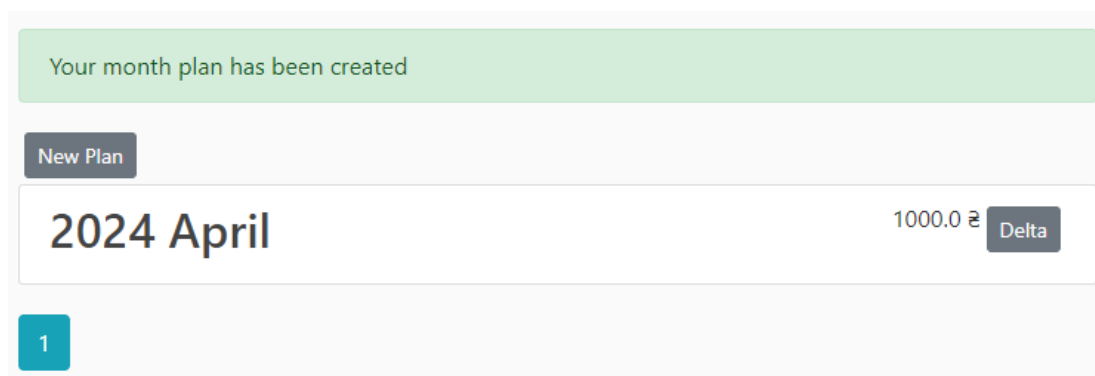
Add

Рисунок 5.13 – Сторінка додавання типу витрат



The screenshot shows a web form titled "New Month Plan". It contains two input fields: "Month" with the value "04/22/2024" and a calendar icon, and "Income" with the value "1000". Below these fields is a blue "Add" button.

Рисунок 5.14 – Створення місячного плану



The screenshot shows a confirmation screen. At the top, a green banner reads "Your month plan has been created". Below it is a "New Plan" button. The main content area displays "2024 April" on the left, "1000.0 ₴" on the right, and a "Delta" button. At the bottom left, there is a blue square button with the number "1".

Рисунок 5.15 – Створення місячного плану

Back to all plans			
Days: 30		Income: 1000.0	
Name	Percent	Daily	Amount
Їжа	1.0 %	0.33 ₴	10.0 ₴
Комуналка	5.0 %		50.0 ₴
Overall Day	1.0 %	0.33 ₴	10.0 ₴
Overall Month	5.0 %		50.0 ₴
Remains	94.0 %		940.0
New Type			

Рисунок 5.16 – Приклад заповнення місячного плану

Також передбачена можливість додавання покупок з csv файлу (рисунок 5.19).

Для того, щоб додати дані з csv файлу, натиснемо відповідну кнопку на сторінці покупок, перейдемо на сторінку додавання, оберемо потрібний файл на комп'ютері та натиснемо кнопку імпортувати (рисунок 5.20).

Для того, щоб дані з таблиці імпортувалися коректно, записи у таблиці мають відповідати певному формату. Цей формат має бути таким, що:

- перший рядок це дата у форматі чч.мм.rrrr - - - -;
- наступні рядки це транзакції - - 'назва' 'кількість' 'сума';
- пустий рядок;
- повторення циклу (рисунок 5.21).

Day

04/25/2024 

Name of item

Молоко

Quantity

1

Quantity type

л 

Spending amount

40

Add

Рисунок 5.17 – Сторінка додавання витрат (покупок)

До прикладу, в розділ покупки додано певні витрати в різні числа місяця, що зображено на рисунку 5.22.

Your spending has been created

Today: 25.04.2024

118.33 ₴ - Saved today [Add purchase](#)

158.33 ₴ - Money planned for today

-40.0 ₴ - Today's result

Edit Delete 40.0 ₴

Молоко

1000 мл

Рисунок 5.18 – Повідомлення про успішно додану покупку

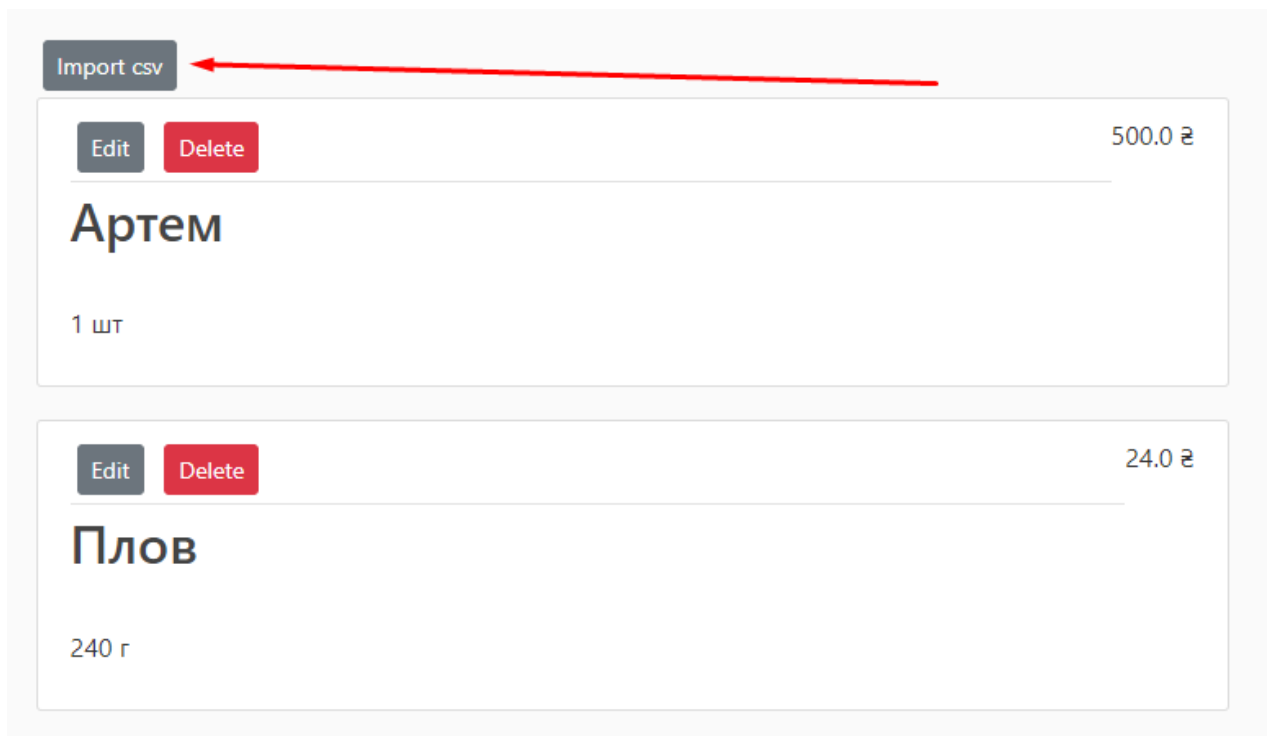


Рисунок 5.19 – Додавання даних з csv файлу

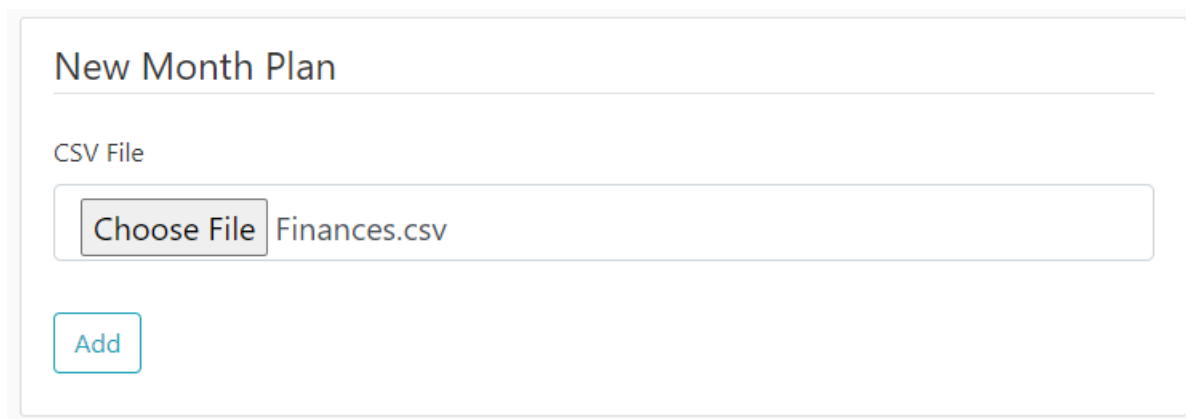


Рисунок 5.20 – Додавання даних з таблиці

5.3.3. Приклад розрахунків

Перейшовши на сторінку дельта в місячному плані бачимо розрахунок різниці між запланованими та фактичними витратами та розрахунок зекономлених грошей.

Зазначимо, що значення зекономлених коштів може бути від’ємним – це означає що витрачено було більше, ніж заплановано (рисунок 5.23)

1	day		name	quantity	price
2	16.03.2021				
3			Батон	500г	19,99
4			Плов	288г	40,03
5			Котлета	82г	17,96
6			Сахар	1кг	25,89
7			Какао	100г	30,39
8			Печенье	1шт	10,49
9			Молоко	900г	29,99
10			Сырок	1шт	9,99
11					
12	17.03.2021				
13			РозеткаП	1шт	15
14			Метро	2шт	16
15			Троллейб	1шт	8
16			Такси	1шт	67
17			Макдонал	1шт	153
18			Маме	1шт	49
19					
20	18.03.2021				
21			Иверия	1шт	50
22			Плов	240г	24
23			Котлета	88г	14

Рисунок 5.21 – Формат csv файлу

Edit Delete 60.0 ₴

Кава з круасаном

1 шт

Edit Delete 35.0 ₴

Морозиво

1 шт

Edit Delete 33.0 ₴

Хліб

1 шт

Рисунок 5.22 – Витрати місяця

[Back to month plans](#)

Start Date: 2024-04-01

Money: 950.0 ₺

End Date: 2024-04-30

Click on a date

Date	Result	Balance	Savings
01.04	-50.0 ₺	31.67 ₺	-18.33 ₺
02.04	-33.0 ₺	31.03 ₺	-19.67 ₺
03.04	-35.0 ₺	30.96 ₺	-23.0 ₺
04.04	0 ₺	30.81 ₺	8.67 ₺
05.04	-60.0 ₺	32.0 ₺	-19.67 ₺
06.04	0 ₺	30.88 ₺	12.0 ₺

Рисунок 5.23 – Розрахунок дельти

Розглянемо як отримано виведені розрахунки. В правому верхньому куті бачимо Money: 950 ₴. Money (залишок коштів) розраховується за формулою (5.1):

$$\text{Залишок коштів} = \text{Доходи} - \sum \text{обов'язкові витрати}, \quad (5.1)$$

Далі бачимо Result, Balance, Savings. Для кожного рядка розраховується наступним чином:

- Result – це сума витрат за день (витрат, зі знаком '-' або доходів, зі знаком '+');
- Balance – кількість грошей, що можна витратити за день, щоб в кінці місяця витратити всі гроші. Balance для дня місяця D розраховується за формулою (5.2):

$$\text{Balance}_D = \frac{\text{Money}}{M-D}, \quad (5.2)$$

де Balance – кількість грошей, що можна витратити на день, Money – залишок коштів, M – загальна кількість днів у місяці, D – порядковий номер поточного дня місяця;

- Savings – це заощадження (зі знаком '+') або ж наднормові витрати (зі знаком '-'), які для дня місяця D розраховуються за формулою (5.3):

$$Savings_D = Result_D + \frac{Money}{M}, \quad (5.3)$$

де Money – це сума доходів за місяць, М – кількість днів у місяці.

Для наочного розуміння динаміки витрат на сторінці окрім таблиці виводиться ще й графік, зображений на рисунку 5.24.

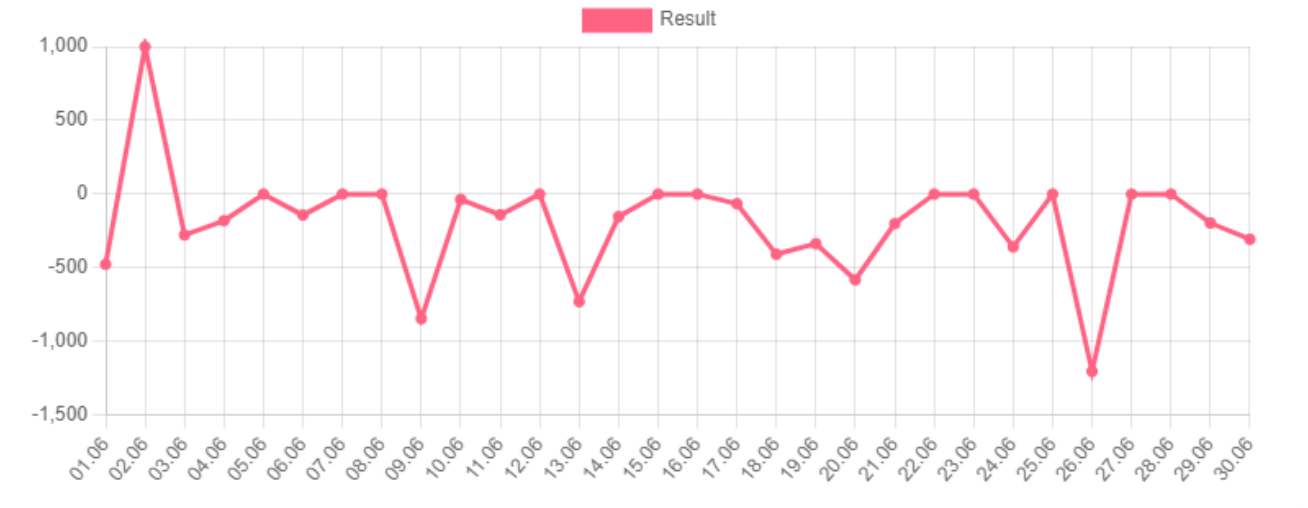


Рисунок 5.24 – Графік динаміки заощаджень

5.3.4. Виведення результатів

Для того, аби краще зрозуміти що показано на попередньому рисунку, розглянемо таблицю в якій наведено більш детальне представлення даних, із проміжними розрахунками (рисунок 5.25)

Розглянемо таблицю по стовпчиках:

1. Date – Дата;
2. Result - Сальдо (тобто результат) на кінець дня. Розраховується як сума всіх транзакцій за день (як витрат зі знаком '+', так і доходів (зі знаком '-'));
3. Dynamic delta – щоденний приріст (або спадання, зі знаком '-') dynamic mid;
4. Dynamic mid – скільки користувач може витрати, починаючи з поточного дня, щоб в кінці місяця витрати всі збереження. Розраховується як сума збережень, поділена на кількість днів місяця, що залишилась;

5. Static mid – накопичення на початок дня;
6. St delta day – сума грошей, що розподілені на непостійні витрати, поділена на кількість днів місяця, що залишились мінус витрати за день;
7. St delta over – накопичення на кінець дня.

Аналогічним чином можливо додавати інші витрати, створювати інші місячні плани, редагувати та видаляти зазначені пункти, таким чином привчаючись до систематизованого обліку, врахування особистих коштів, витрат та всього із чим це пов'язано. Зрештою, привчаючись до планування та економії власних фінансів.

Date	Result	Dynamic delta	Dynamic mid	Static mid	St delta day	St delta over
01.05	0 ₴	32.26 ₴	967.74 ₴	967.74 ₴	967.74 ₴	967.74 ₴
02.05	0 ₴	34.48 ₴	1000.0 ₴	1935.48 ₴	967.74 ₴	1935.48 ₴
03.05	0 ₴	36.95 ₴	1034.48 ₴	2903.23 ₴	967.74 ₴	2903.23 ₴
04.05	0 ₴	39.68 ₴	1071.43 ₴	3870.97 ₴	967.74 ₴	3870.97 ₴
05.05	0 ₴	42.74 ₴	1111.11 ₴	4838.71 ₴	967.74 ₴	4838.71 ₴
06.05	0 ₴	46.15 ₴	1153.85 ₴	5806.45 ₴	967.74 ₴	5806.45 ₴
07.05	0 ₴	50.0 ₴	1200.0 ₴	6774.19 ₴	967.74 ₴	6774.19 ₴
08.05	-500.0 ₴	32.61 ₴	1250.0 ₴	7741.94 ₴	467.74 ₴	7241.94 ₴
09.05	0 ₴	58.3 ₴	1282.61 ₴	8209.68 ₴	967.74 ₴	8209.68 ₴
10.05	0 ₴	63.85 ₴	1340.91 ₴	9177.42 ₴	967.74 ₴	9177.42 ₴

Рисунок 5.25 – Проміжні розрахунки

5.3.5. Скидання пароля

На рисунку 5.26 зображено сторінку керування обліковим записом користувача, де він може переглянути на яку електронну адресу зареєстрований його обліковий запис та його ім'я користувача на сайті. Передбачено можливість змінити ці два параметри, а також здійснити скидання пароля (рисунок 5.27).

The screenshot shows a web form titled 'Update Account Info'. At the top, the user's name 'kelok' is displayed, followed by their email 'admin@admin.com'. Below this, there are two input fields: 'Username' with the value 'kelok' and 'Email' with the value 'admin@admin.com'. A 'Submit' button is located below the email field. At the bottom left of the form, there is a link labeled 'Reset password'.

Рисунок 5.26 – Сторінка налаштування облікового запису

The screenshot shows a web page header for 'Day to Money' with navigation links: Home, About, Purchases, Plans, Add purchase, Logout, and Account. The main content area features a large, light gray box containing a single button labeled 'Request Password Reset'.

Рисунок 5.27 – Сторінка скидання пароля

Для того, щоб скинути пароль від облікового запису потрібно у профілі користувача натиснути відповідну кнопку та підтвердити запит на скидання пароля. Після виконання цієї дії на вказану при реєстрації електронну пошту прийде посилання, за яким треба користувачеві треба буде перейти та встановити новий пароль для його облікового запису.

ВИСНОВКИ

В ході виконання даної роботи розроблено веб-застосунок, який дозволяє користувачам ефективно керувати своїми фінансами, вести облік доходів та витрат, аналізувати фінансовий стан та планувати майбутні витрати, шляхом перелічених функціональних можливостей:

1. створення власних облікових записів;
2. можливості імпортувати дані з csv файлу або внесення вручну;
3. створення місячних планів та внесення витрат;
4. моніторингу витрат у вигляді таблиці або графіків.

Проведено детальне дослідження предметної області управління особистим бюджетом та аналіз існуючих програмних систем з цієї області. Цей аналіз дозволив зрозуміти основні проблеми та потреби користувачів у сфері фінансового управління.

За результатом аналізу існуючих програмних систем було виявлено як позитивні, так і негативні аспекти їхньої реалізації. Цей аналіз допоміг врахувати кращі практики та уникнути помилок при розробці власного веб-застосунку.

Розроблене програмне забезпечення було піддане тестуванню, яке підтвердило правильну роботу системи. Були проведені різні види тестів, включаючи модульні, інтеграційні тести, а також тести в реальних умовах.

Розроблена система може бути використана будь-якою особою, яка бажає покращити свої навички керування особистими фінансами, шляхом планування та відстеження своїх доходів і витрат.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Дослідження USAID та НБУ "Фінансова грамотність, фінансова інклюзія та фінансовий добробут в Україні у 2021 році" [Електронний ресурс] – Режим доступу:https://bank.gov.ua/admin_uploads/article/Research_Financial_Literacy_Inclusion_Welfare_2021.pdf
2. Matthes Eric, Python Accelerated Course, 2nd edition: A Practical Introduction to Project-Based Programming: A Guide. San Francisco, USA, ed. No Starch Press, 2019. 150 с.
3. Greenberg Miguel. Flask Web Development: Developing Web Applications with Python: A Guide, Sebastopol, USA, ed. O'Reilly Media, 2018. 308 с.
4. Справочник CSS [Електронний ресурс] — Режим доступу: <http://htmlbook.ru/css>
5. Офіційний сайт Bootstrap 3 [Електронний ресурс] — Режим доступу: <http://bootstrap-3.ru/index.php> Дата доступу: 01.06.2017
6. Swigart Al, Automate boring things with Python: Practical Programming for Beginners: A Guide. San Francisco, USA, ed. No Starch Press, 2015. 244 с.
7. Fiale Chris, SQL: A guide to learning the language: a manual. Berkeley, USA, ed. Peachpit Press, 2003. - 456 с.
8. Когаловский М. Р. Энциклопедия технологий баз данных / М. Р. Когаловский. — М.: Финансы и статистика, 2002. — 800 с.
9. David M. Beazley and Brian C. Jones. Python Cookbook: Recipes for mastering Python 3: A guide. Sebastopol, USA, publisher. O'Reilly Media, 2013. 303 с.

Додаток А

Реалізація роботи місячних планів

Аркушів 4

Київ 2024

Программный код

```

from calendar import monthrange

from flask import Blueprint, request, render_template, flash, url_for, redirect
from flask_login import login_required, current_user

from app import db
from app.helper.month_plan_table import plan_to_dict, daily_overall_dict
from app.helper.name_months import months_names
from app.models import Month_plans
from app.month_plan.forms import MonthPlanForm

month_plans = Blueprint('month_plans', __name__)

@month_plans.route('/planning/monthly/months', methods=['GET', 'POST'])
@login_required
def show_all_months():
    page = request.args.get('page', 1, type=int)
    month_plans =
    Month_plans.query.filter_by(user_id=current_user.id).order_by(Month_plans.month.desc(
    )).paginate(
        page=page, per_page=5)
    return render_template('month_plans/monthplanning.html',
    month_plans=month_plans, myfunction=months_names)

@month_plans.route('/planning/monthly/new', methods=['GET', 'POST'])
@login_required
def month_plan_new():
    form = MonthPlanForm()
    if form.validate_on_submit():
        monthplan = Month_plans(month=form.month.data,
                                income=form.income.data,
                                money_for_month=0,
                                # todo change 0 to actual value
                                money_for_day=form.income.data,
                                user_id=current_user.id)
        db.session.add(monthplan)
        db.session.commit()
        flash('Your month plan has been created', 'success')
        return redirect(url_for('month_plans.show_all_months'))
    calculated by plan| maybe add more if-coditions to calculate values properly?

```

```

    return render_template('month_plans/create_new_month_plan.html', title='New
Month Plan', form=form, legend='New Month Plan')

```

```

@month_plans.route('/planning/monthly/<int:month_plan_id>',
                    methods=['GET', 'POST'])
@login_required
def month_plan_table_test(month_plan_id):
    month_plan = Month_plans.query.get_or_404(month_plan_id)
    days = monthrange(month_plan.month.year, month_plan.month.month)[1]

    return render_template(
        'month_plans/month_plan.html',
        title=str(month_plan.month.month) + '.' + str(month_plan.month.year),
        month_plan=month_plan,
        legend='TEST LEGEND FOR MONTH PLAN',
        planDictionary=plan_to_dict(month_plan),
        days=days,
        monthOverall=daily_overall_dict(
            plan_to_dict(month_plan),
            month_plan.income
        ),
    )

```

```

@month_plans.route('/planning/monthly/<int:month_plan_id>/delete', methods=['GET',
'POST'])
@login_required
def delete_month_plan(month_plan_id):
    if not current_user.is_authenticated:
        return redirect(url_for('main.about'))
    month_plan = Month_plans.query.get_or_404(month_plan_id)
    db.session.delete(month_plan)
    db.session.commit()
    flash('Month plan deleted successfully', 'success')
    return redirect(url_for('month_plans.show_all_months'))

```

```

@month_plans.route('/planning/monthly/<int:month_plan_id>/edit', methods=['GET',
'POST'])
@login_required
def edit_month_plan(month_plan_id):
    if not current_user.is_authenticated:
        return redirect(url_for('main.about'))
    month_plan = Month_plans.query.get_or_404(month_plan_id)

```

```
form = MonthPlanForm()
if form.validate_on_submit():
    month_plan.month = form.month.data
    month_plan.income = form.income.data
    db.session.commit()
    flash('Month plan has been edited successfully', 'success')
    return redirect(url_for('month_plans.month_plan_table_test',
month_plan_id=month_plan.id))
elif request.method == 'GET':
    form.month.data = month_plan.month
    form.income.data = month_plan.income
    return render_template('month_plans/create_new_month_plan.html', title='Edit
Month Plan', legend='Edit Month Plan', form=form)
```

Додаток Б

Реалізація виведення та обробки витрат на сервері

Аркушів 6

Київ 2024

Программный код

```

import os
from datetime import datetime
from io import StringIO

import pandas as pd
from flask import Blueprint, request, render_template, flash, redirect, url_for
from flask_login import login_required, current_user

from app import db
from app.helper.converter import converter
from app.helper.separator import separator
from app.helper.spend_validate import validate_spending
from app.models import Spending
from app.spending.forms import SpendingForm, UploadCSVForm

spending = Blueprint('spending', __name__)

def read_and_paste_csv(df, user_id):
    day = ""
    for index, row in df.iterrows():
        # print(row)

        day_value = row['day']
        name_of_item = row['name']

        quantity_value = row['quantity']
        quantity = ""
        quantity_type = ""

        spending_amount = row['price']

        if pd.isnull(day_value):
            if pd.isnull(quantity_value):
                pass
            else:
                separated = separator(quantity_value)
                converted = converter(separated)
                converted_quantity = converted[0]
                converted_quantity_type = converted[1]
                quantity = converted_quantity
                quantity_type = converted_quantity_type

```

```

        if ',' in str(spending_amount):
            bla = (str(spending_amount).replace(",", ".").replace(' ', ''))
            spending_amount = float(bla.strip(' '))

    print(day, '|', name_of_item, '|', quantity, '|', quantity_type, '|',
spending_amount)

    # print(row)
    # print(type(day), '--', day)
    # print(type(quantity), '--', quantity)
    # print(type(quantity_type), '--', quantity_type)
    print('-----')

    test_spend = Spendings(day=day, name_of_item=name_of_item,
quantity=quantity,
                                quantity_type=quantity_type,
                                spending_amount=spending_amount,
user_id=user_id)

    db.session.add(test_spend)
    db.session.commit()

    # dto = datetime.strptime(date_str, '%d.%m.%Y').date()

    else:
        day = datetime.strptime(day_value, '%d.%m.%Y').date()

    return True

@spendings.route('/spending/all', methods=['GET', 'POST'])
@login_required
def _spendings():
    page = request.args.get('page', 1, type=int)
    day_spending =
    Spendings.query.filter_by(user_id=current_user.id).order_by(Spendings.id.desc()).paginate(page=page,

per_page=10)

    return render_template('spendings/spendings.html', day_spending=day_spending)

@spendings.route('/spending/import', methods=['GET', 'POST'])
@login_required
def spending_import_csv():

```

```

form = UploadCSVForm()
if form.validate_on_submit():
    file = form.file.data
    # print(file)
    # print(file.filename)
    file_path = os.path.join('app', 'temp_files', file.filename)
    file.save(file_path)
    df = pd.read_csv(file_path, encoding='windows-1251', sep=';')

    print(df.head())
    if not read_and_paste_csv(df, current_user.id):
        flash("CSV is not imported!", 'danger')
        return redirect(url_for('month_plans.show_all_months'))
    flash('CSV file has been successfully uploaded and read!', 'success')
    return redirect(url_for('month_plans.show_all_months'))
    return render_template('spendings/spendings_import.html', title='New Month Plan',
form=form, legend='New Month Plan')

@spendings.route('/spending/day')
@login_required
def spendings_day():
    a = request.args.get('start_day') # todo rename variable
    b = request.args.get('day') # todo rename variable
    c = datetime.strptime(f'{b}.{a[:4]}', '%d.%m.%Y').date() # todo rename variable

    day_spending =
    Spendings.query.filter_by(user_id=current_user.id).filter(Spendings.day == c).order_by(
        Spendings.id.desc()).all()
    return render_template('spendings/spendings_day.html',
day_spending=day_spending)

@spendings.route('/spending/new', methods=['GET', 'POST'])
@login_required
def new_spending():
    form = SpendingForm()
    if form.validate_on_submit():
        spending = Spendings(day=form.day.data,
                                name_of_item=form.name_of_item.data,
                                quantity=form.quantity.data,
                                quantity_type=form.quantity_type.data,

spending_amount=form.spending_amount.data,
                                user_id=current_user.id)
        validate_spendings(spending)

```

```

        db.session.add(spending)
        db.session.commit()
        flash('Your spending has been created', 'success')
        return redirect(url_for('main.home'))
    return render_template('spendings/create_new_spending.html', title='New Purchase',
form=form, legend='New Purchase')

```

```

@spendings.route('/spending/<int:spending_id>')
@login_required
def spending(spending_id):
    if not current_user.is_authenticated:
        return redirect(url_for('main.home'))
    spending = Spendings.query.get_or_404(spending_id)
    return render_template('spendings/spending.html', title=spending.name_of_item,
purchase=spending)

```

```

@spendings.route('/spending/<int:spending_id>/delete', methods=['GET', 'POST'])
@login_required
def delete_spending(spending_id):
    if not current_user.is_authenticated:
        return redirect(url_for('main.home'))
    spending = Spendings.query.get_or_404(spending_id)
    db.session.delete(spending)
    db.session.commit()
    flash('Purchase has been deleted!', 'success')
    return redirect(url_for('main.home'))

```

```

@spendings.route('/spending/<int:spending_id>/edit', methods=['GET', 'POST'])
@login_required
def edit_spending(spending_id):
    if not current_user.is_authenticated:
        return redirect(url_for('main.home'))
    spending = Spendings.query.get_or_404(spending_id)
    form = SpendingForm()
    if form.validate_on_submit():
        spending.day = form.day.data
        spending.name_of_item = form.name_of_item.data
        spending.quantity = form.quantity.data
        spending.quantity_type = form.quantity_type.data
        spending.spending_amount = form.spending_amount.data

    validate_spendings(spending)

```

```
        db.session.commit()
        flash('Purchase has been updated!', 'success')
        return redirect(url_for('spendings.spending', spending_id=spending.id))
    elif request.method == 'GET':
        form.day.data = spending.day
        form.name_of_item.data = spending.name_of_item
        form.quantity.data = spending.quantity
        form.quantity_type.data = spending.quantity_type
        form.spending_amount.data = spending.spending_amount
    return render_template('spendings/create_new_spending.html', title='Edit Purchase',
legend='Edit Purchase', form=form)
```