ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
МІСЬКОГО ГОСПОДАРСТВА ІМЕНІ О. М. БЕКЕТОВА
Навчально-науковий інститут енергетичної, інформаційної
та транспортної інфраструктури

Кафедра комп'ютерних наук та інформаційних технологій

Пояснювальна записка
до кваліфікаційної роботи бакалавра

на тему: «Проєктування вебдодатку для контролю бюджету, трекінгу звичок та управління завданнями»

Виконала: здобувачка вищої освіти,
групи КН 2021-1
спеціальності
122 Комп'ютерні науки
(шифр і назва спеціальності)

Діана НОВІКОВА
(ім'я та прізвище)

Керівник: Наталія БРАТЕРСЬКА
(ім'я та прізвище)

Рецензент: Марина НОВОЖИЛОВА
(ім'я та прізвище)

м. Харків – 2025 рік

Харківський національний університет міського господарства імені О. М. Бекетова
(повне найменування закладу вищої освіти)
Навчально-науковий Інститут енергетичної, інформаційної та транспортної інфраструктури
Кафедра комп'ютерних наук та інформаційних технологій
Рівень вищої освіти перший (бакалаврський)
Спеціальність 122 Комп'ютерні науки
(шифр і назва)

ЗАТВЕРДЖУЮ
Завідувач кафедри КНтаІТ

Марина НОВОЖИЛОВА
«23» 06 2025 року

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

Новікової Діани Геннадіївни
(прізвище, ім'я, по батькові)

1. Тема роботи «Проєктування вебдодатку для контролю бюджету, трекінгу звичок та управління завданнями»

керівник роботи Братерська Наталія Миколаївна, асистент кафедри КН та ІТ
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджені наказом закладу вищої освіти від 09 травня 2025 р. №341-03
2. Термін подання студентом роботи 23 червня 2025 р.
3. Вихідні дані до роботи: проєктування і реалізація вебдодатку для контролю бюджету, трекінгу звичок та управління завданнями з використанням Python, Django, SQLite.
4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити): 1. Аналітична частина 2. Інформаційно-математична частина 3. Програмно-технічна частина 4. Охорона праці
5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень):

Презентація – 13 слайдів

6. Консультанти розділів роботи

Розділ	Ім'я та Прізвище, посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1	Наталія БРАТЕРСЬКА, асистент кафедри КН та ІТ	12.05.2025	20.05.2025
2	Наталія БРАТЕРСЬКА, асистент кафедри КН та ІТ	20.05.2025	28.05.2025
3	Наталія БРАТЕРСЬКА, асистент кафедри КН та ІТ	28.05.2025	05.06.2025
4	Вікторія МАЛИШЕВА, к. т. н., доцент кафедри ОП та БЖ	21.05.2025	03.06.2025

7. Дата видачі завдання 12 травня 2025 року

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Вибір теми дипломної роботи	05.05.2025	виконано
2	Затвердження тем, наукових керівників, завдань та календарного плану підготовки кваліфікаційної роботи	09.05.2025	виконано
3	Написання I розділу	20.05.2025	виконано
4	Написання II розділу	28.05.2025	виконано
5	Написання III розділу	05.06.2025	виконано
6	Написання IV розділу	09.06.2025	виконано
7	Подання дипломної роботи керівнику	10.06.2025	виконано
8	Робота по усуненню зауважень керівника, уточнення і доповнення практичного матеріалу, оформлення додатків до роботи	16.06.2025	виконано
9	Подання доопрацьованого варіанту роботи керівнику	18.06.2025	виконано
10	Захист матеріалів дипломної роботи на засіданні кафедри	20.06.2025	виконано
11	Офіційний захист матеріалів дипломної роботи на засіданні екзаменаційної комісії	24.06.2025	виконано

Студент _____ (підпис) Діана НОВІКОВА (прізвище та ініціали)

Керівник роботи _____ (підпис) Наталія БРАТЕРСЬКА (прізвище та ініціали)

АНОТАЦІЯ

Структура та обсяг роботи. Пояснювальна записка кваліфікаційної роботи бакалавра студентки групи КН 2021-1 спеціальності 122 Комп'ютерні науки Новікової Діани Геннадіївни за темою «Проєктування вебдодатку для контролю бюджету, трекінгу звичок та управління завданнями» складається з 4 розділів, містить 52 рисунки, 5 таблиць, 22 джерела.

Кваліфікаційну роботу бакалавра присвячено проєктуванню вебдодатку для самоорганізації, що включає контроль фінансових операцій, відстежування дотримання звичок та керування завданнями. Розроблено інформаційне та програмне забезпечення для розв'язання цієї задачі.

У розділі «Загальні положення» наведено огляд предметного середовища включно з процесом діяльності й функціональною моделлю та наявних аналогів, визначено задачі дослідження.

Розділ «Інформаційне та математичне забезпечення» присвячений проєктуванню бази даних та класів системи, розгляду математичного та алгоритмічного забезпечення ключових складових вебдодатку для самоорганізації і визначенню вхідних і вихідних даних.

У розділі «Програмне та технічне забезпечення» наведено опис засобів розробки, вимог до технічного та програмного забезпечення та керівництво користувача. Розглянуто процес створення бази даних та програмної реалізації.

У розділі «Охорона праці» визначено роль законодавства у забезпеченні охорони праці, розглянуто шкідливі та небезпечні виробничі фактори для IT-фахівців та запропоновано заходи для зменшення їх впливу.

Ключові слова: ВЕБДОДАТОК, КОНТРОЛЬ БЮДЖЕТУ, УПРАВЛІННЯ ЗАВДАННЯМИ, ВІДСТЕЖЕННЯ ЗВИЧОК, КОРИСТУВАЧ, БАЗА ДАНИХ, DJANGO, PYTHON.

ANNOTATION

Structure and scope of work. Explanatory note of the bachelor's qualification work of the student of the group KN 2021-1, specialty 122 Computer Science Novikova Diana Hennadiivna on the topic “Designing a web application for budget control, habit tracking and task management” consists of 4 sections, contains 52 figures, 5 tables, 22 sources.

The bachelor's thesis is devoted to the design of a web application for self-organization, including control of financial transactions, tracking of habits, and task management. Information and software for solving this problem was developed.

The “General Provisions” section provides an overview of the subject environment, including the process of activity and functional model and existing analogues, and defines the research objectives.

The section “Information and Mathematical Support” is devoted to the design of the database and system classes, consideration of the mathematical and algorithmic support of the key components of the web application for self-organization and the definition of input and output data.

The Software and Hardware section describes the development tools, hardware and software requirements, and the user manual. The process of creating a database and software implementation is described.

The Labor Protection section defines the role of legislation in ensuring labor protection, considers harmful and dangerous production factors for IT professionals, and suggests measures to reduce their impact.

Keywords: WEB APPLICATION, BUDGET CONTROL, TASK MANAGEMENT, HABIT TRACKING, USER, DATABASE, DJANGO, PYTHON.

ЗМІСТ

ВСТУП

В умовах постійного зростання темпу життя допоміжні засоби для організації особистого часу, фінансів та повсякденних справ стають важливою складовою ефективного управління власними ресурсами. Необхідність одночасно контролювати різні аспекти життя потребує використання сучасних цифрових рішень, які сприятимуть досягненню поставлених цілей і полегшуватимуть прийняття зважених рішень.

Актуальність проєктування вебдодатку для самоорганізації полягає у потребі створення єдиної системи для ефективного розподілу завдань, відстеження доходів і витрат та дотримання корисних звичок без необхідності використання різних платформ. Поєднання цих аспектів сучасного життя справлятиме позитивний вплив на рівень мотивації, продуктивності та організованості користувача. Вебдодаток для контролю бюджету, трекінгу звичок та управління завданнями стане надійним інструментом для підтримки дисципліни, досягнення особистих цілей і прийняття зважених рішень у повсякденному житті.

Метою даної роботи є створення програмного продукту для комплексного управління особистою ефективністю. Розроблений вебдодаток надасть безпечні умови для фінансового обліку, організації завдань, відстеження успішності впровадження звичок та наочного демонстрування продуктивності.

Задля досягнення поставленої мети визначено наступні завдання дослідження:

– аналіз предметної області та наявних програмних рішень;
– проєктування архітектури та функціоналу інформаційної системи;
– обґрунтування вибору інструментів розробки;
– реалізація і тестування програмного продукту;
– оцінка зручності використання та ефективності системи.

Об’єкт дослідження – веборієнтована інформаційна система для самоорганізації користувача.

Предметом дослідження є інструментальні засоби, методи проєктування та реалізації програмного забезпечення для управління особистими ресурсами.

Для дослідження обрано такі методи: аналіз наукових джерел та існуючих аналогів, моделювання функціональної структури системи, методи об’єктно-орієнтованого програмування, тестування та оцінювання зручності інтерфейсу користувача.

Результатом роботи стане вебдодаток, що поєднує такий функціонал, як керування фінансами, завданнями і звичками, в єдиному середовищі.

Отримані результати мають теоретичну цінність у контексті узагальнення підходів до інтегрованих систем самоорганізації, а також прикладну значущість для розробників, користувачів і фахівців, зацікавлених в ефективному управлінні особистими ресурсами через сучасні вебтехнології.

РОЗДІЛ 1
ЗАГАЛЬНІ ПОЛОЖЕННЯ

1.1 Опис предметного середовища

Планування особистого бюджету та завдань і формування корисних звичок є важливою складовою сучасного життя, особливо в умовах постійної зайнятості, великої кількості інформації та швидкого темпу змін. Зростаюча потреба у збереженні психологічної рівноваги, контролі над фінансами та щоденними справами спонукає дедалі більше людей користуватися цифровими інструментами для самоорганізації.

Ринок застосунків, призначених для контролю бюджету, трекінгу звичок і управління завданнями, динамічно розвивається. Він охоплює різноманітні інструменти – від простих планувальників до комплексних систем із аналітикою витрат, нагадуваннями, гейміфікацією й можливістю інтеграції з банківськими рахунками. Такі застосунки стають важливою частиною цифрової екосистеми користувача, забезпечуючи зручний інтерфейс для організації особистих та фінансових процесів.

Згідно з опитуванням, що проводилося 2024 року компанією Origin серед більш ніж 1000 представників поколінь Z та міленіалів у США, понад 42% учасників мають встановлені застосунки для бюджетування [1], що підтверджує актуальність та поширеність таких інструментів серед молоді.

За даними дослідження Global Growth Insights, понад 65% користувачів у США активно стежать за своїм розпорядком дня, а 42% компаній впровадили застосунки для трекінгу звичок у свої програми корпоративного благополуччя, підкреслюючи важливість ментального здоров’я та щоденної відповідальності [2]. Це свідчить про зростання інтересу не лише до особистої ефективності, а й до впровадження цифрових рішень у професійне та соціальне середовище, що також підтверджує актуальність розробки таких систем.

Отже, застосунки для бюджетування, планування завдань і відстеження звичок відіграють дедалі важливішу роль у підтриманні особистої організованості, психологічного добробуту та продуктивності, що обумовлює потребу у створенні нових ефективних рішень у цій сфері.

1.1.1 Опис процесу діяльності

Процес взаємодії користувача з вебдодатком розпочинається з реєстрації чи авторизації, які забезпечують ідентифікацію та захист персональних даних. Після входу в систему з’являється доступ до трьох функціональних складових: бюджету, завдань і звичок.

На сторінці бюджету користувач має можливість обирати період для перегляду та внесення фінансових записів, управляти категоріями доходів і витрат, а також контролювати баланс коштів. Автоматичний підрахунок сум забезпечує швидкий аналіз фінансового стану за обраний період. Цей модуль дозволяє систематизувати і структурувати фінансову інформацію, підвищуючи точність планування та облік особистих ресурсів.

Контроль особистого бюджету передбачає обробку персональних даних користувачів, тому одним із пріоритетів при розробці є забезпечення надійного захисту інформації. Задля зменшення ймовірності витоку конфіденційних даних, що може бути спричинено ін’єкційними атаками, міжсайтовим скриптингом, порушенням контролю доступу чи криптографічними помилками, було розглянуто сучасні підходи протидії даним загрозам [3].

Одним із найважливіших кроків гарантування безпеки даних користувачів є надійне збереження паролів, що використовуються під час реєстрації чи авторизації для доступу до вебдодатку. Це можливо завдяки хешуванню, у процесі якого пароль перетворюється у незворотний рядок фіксованої довжини. Задля збільшення рівня захисту використовується сіль – випадковий набір символів, який додається до паролю перед його хешуванням [4, с. 49]. Таким чином у базі даних зберігається хеш-код, що унеможливлює

відновлення початкового пароля навіть у разі несанкціонованого доступу до сховища.

Сторінка завдань забезпечує інструменти для створення та моніторингу виконання особистих завдань. Користувач може вносити нові записи, відмічати статус їх виконання, встановлювати дедлайни та переглядати список завершених. Автоматизація цього процесу сприяє підвищенню продуктивності, дозволяючи концентруватися на актуальних справах без ризику втрати важливої інформації.

Модуль звичок реалізовується у вигляді календаря, де користувач переглядає виконання регулярних дій, обираючи з переліку конкретних звичок. Відображення успішності дає змогу оцінити прогрес та рівень дисципліни, що сприяє формуванню стабільних корисних звичок та підвищенню якості життя.

Загальна логіка взаємодії з системою відображена на UML-діаграмі діяльності (рис. 1.1).

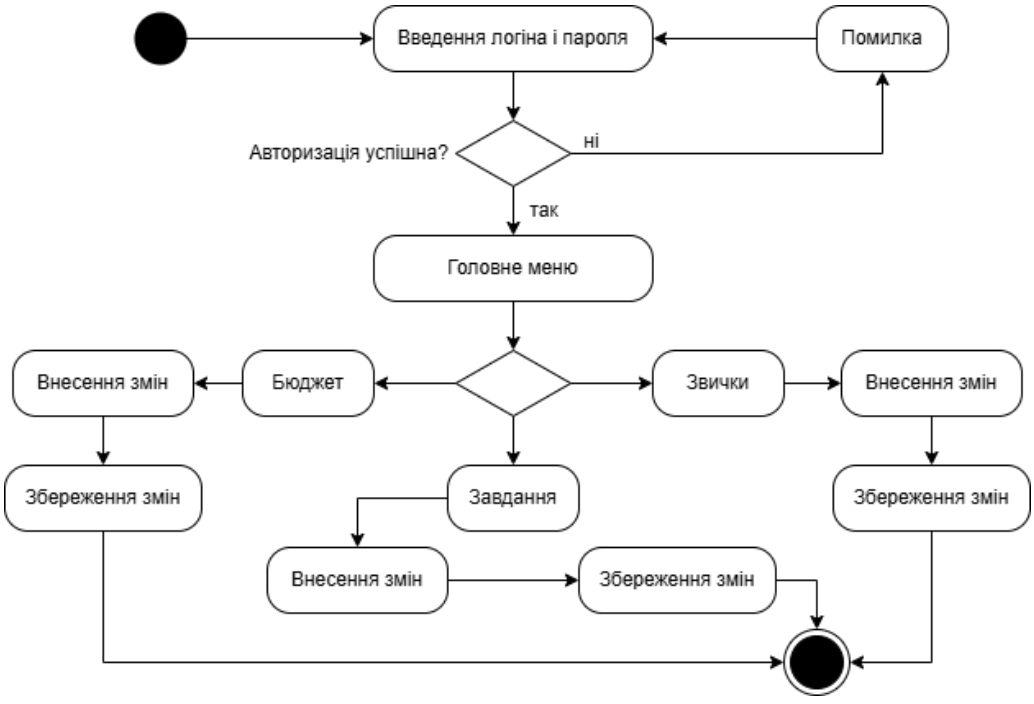


Рисунок 1.1 – UML-діаграма діяльності користувача у вебдодатку для самоорганізації

1.1.2 Опис функціональної моделі

Функціональна модель вебдодатку для самоорганізації побудована на взаємодії єдиної дійової особи – користувача – із трьома основними модулями: планування бюджету, управління завданнями та трекінг звичок. Кожен із них забезпечує користувача набором функцій для ефективного контролю різних аспектів особистого життя.

Дії, пов’язані з доступом до системи, включають реєстрацію, вхід до облікового запису та вихід із системи. Основними діями користувача після входу є контроль бюджету, відстежування звичок та управління завданнями. Це відображено на UML-діаграмі використання (рис. 1.2).

Рисунок 1.2 – UML-діаграма використання основних функцій користувача у вебдодатку для самоорганізації

Контроль бюджету включає наступні дії:

– вибір місяця і року;
– додавання доходів;
– додавання витрат;
– додавання категорій;
– управління накопиченнями;

– перегляд записів про фінансові операції.

До управління завданнями відносяться:

– створення нового завдання;
– видалення існуючого завдання;
– позначення виконаних завдань;
– перегляд активних та виконаних завдань.

Відстежування звичок включає такі дії:

– додавання нової звички;
– встановлення мети;
– вибір звички;
– позначення днів дотримання звички;
– перегляд прогресу;
– продовження виконання набутої звички.

1.2 Огляд наявних аналогів

1.2.1 Вебдодаток Notion

Notion [5] – це вебдодаток з широким функціоналом, створений для зручного управління завданнями і ведення нотаток. Його головною особливістю є гнучкість у налаштуванні сторінок і шаблонів під потреби конкретного користувача. Цей сервіс підтримує не лише роботу через браузер, він доступний у формі мобільного та десктопного застосунків для різних операційних систем.

Основні функціональні можливості Notion включають:

– створення сторінок та розділів для різних проєктів;
– списки завдань і Kanban-дошки з можливістю встановлювати статуси, терміни та пріоритети;
– бази даних і таблиці з фільтрацією й формулами для розрахунку;
– календарі в помісячному або потижневому форматі з нагадуваннями;

– шаблони спільноти, адаптовані під різні потреби (частина яких доступна на платній основі);

– спільна робота з коментарями, згадками користувачів, контролем прав доступу та історією змін;

– синхронізація між пристроями.

Розглядаючи можливість ведення особистого бюджету в Notion, варто підкреслити, що користувачу надається можливість створювати таблиці з категоріями витрат і доходів, автоматизувати підрахунки за допомогою формул та відстежувати динаміку у вигляді графіків або діаграм. Завдяки вбудованому шаблону «Monthly Budget» можна контролювати особистий бюджет за місяць, підраховуючи загальну суму (рис. 1.3).

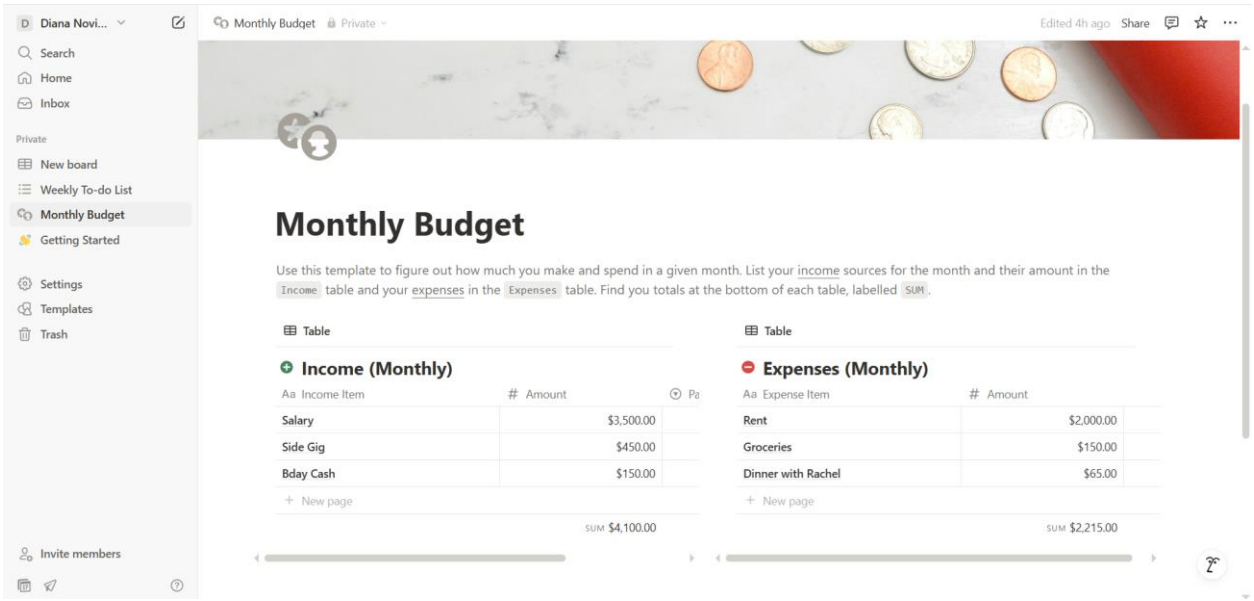


Рисунок 1.3 – Ведення особистого бюджету в Notion

Notion надає можливість відстежувати виконання завдань кількома способами, один із яких – використання Kanban-дошки, де завдання розподіляються за статусами «Не розпочато», «У процесі» і «Готово» (рис. 1.4). Платформа дозволяє включати додаткові властивості, такі як дата, людина, текст, файл тощо.

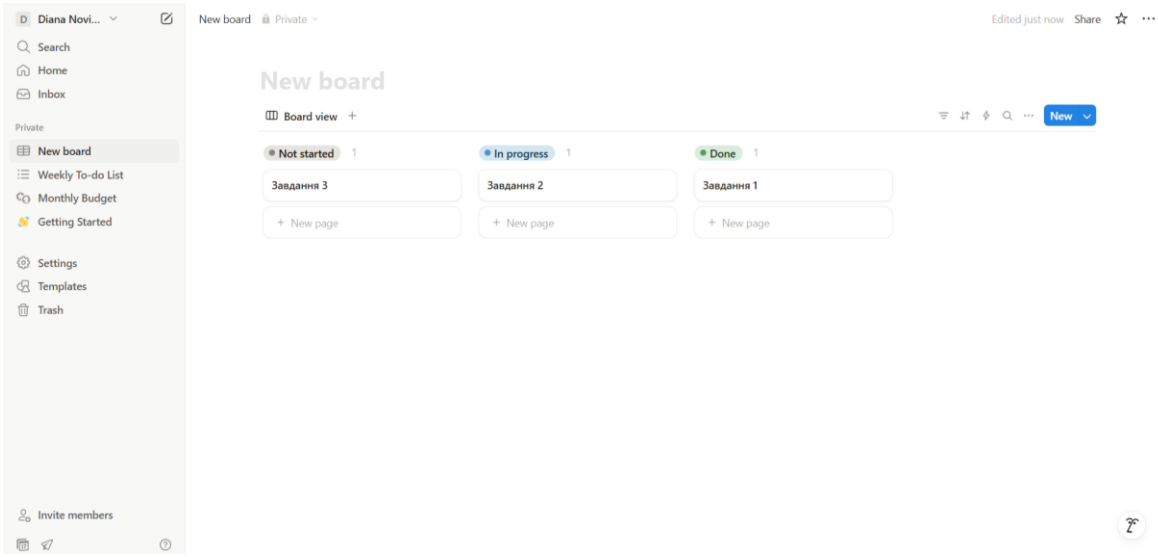


Рисунок 1.4 – Контроль виконання завдань в Notion у формі Kanban-дошки

Іншим доступним варіантом контролю виконання завдань є списки з чекбоксами, організовані за днями тижня, або ж списки, які користувач налаштовує самостійно. Крім того, у Notion є всі необхідні інструменти для створення трекерів звичок з відмітками «виконано» й «пропущено» та автоматичним підрахуванням прогресу.

Отже, даний вебдодаток має ряд переваг завдяки широкому функціоналу. Користувачі можуть створювати власні сторінки, індивідуально налаштовуючи робочий простір, або використовувати шаблони спільноти. Notion пропонує зручні інструменти, з якими реалізація контролю бюджету, завдань та звичок поєднується в єдиній системі. Попри це, на платформі відсутнє готове рішення для відстеження дотримання звичок та обліку фінансів, що враховувало б вибір місяця та року й заповнення таблиці кожного дня. Тому для таких потреб користувачам необхідно створювати власні планери або обирати відповідні шаблони з-поміж наявних у спільноті.

1.2.2 Вебдодаток TickTick

TickTick [6] – вебдодаток для ефективного управління завданнями, планування часу та формування корисних звичок. Він відомий своєю простотою у використанні, інтуїтивно зрозумілим інтерфейсом та широкими можливостями для організації особистої продуктивності. TickTick доступний

не лише у вигляді вебдодатку, а і як мобільний і десктопний застосунок для різних операційних систем та розширення для браузера.

Вебдодаток TickTick має такі корисні функції:

– створення списків завдань із чекбоксами, які можна групувати за проєктами чи категоріями;

– календарі з можливістю перегляду за днем, тижнем або місяцем;

– нагадування з налаштуванням часу та повторів;

– встановлення пріоритетів і термінів виконання для завдань;

– таймер Pomodoro для ефективного тайм-менеджменту;

– матриця Ейзенхауера для розподілу завдань за пріоритетом;

– трекери звичок зі статистикою прогресу;

– спільна робота з можливістю ділитися списками та завданнями;

– синхронізація між пристроями.

Безкоштовна версія вебдодатку пропонує зручне планування завдань у вигляді Kanban-дошки і списку з додаванням чекбоксів, термінів виконання, тегів та прапорців для позначення важливості (рис. 1.5). Окремо присутня фільтрація за датою чи, наприклад, пріоритетом, що значно спрощує відстеження задач.

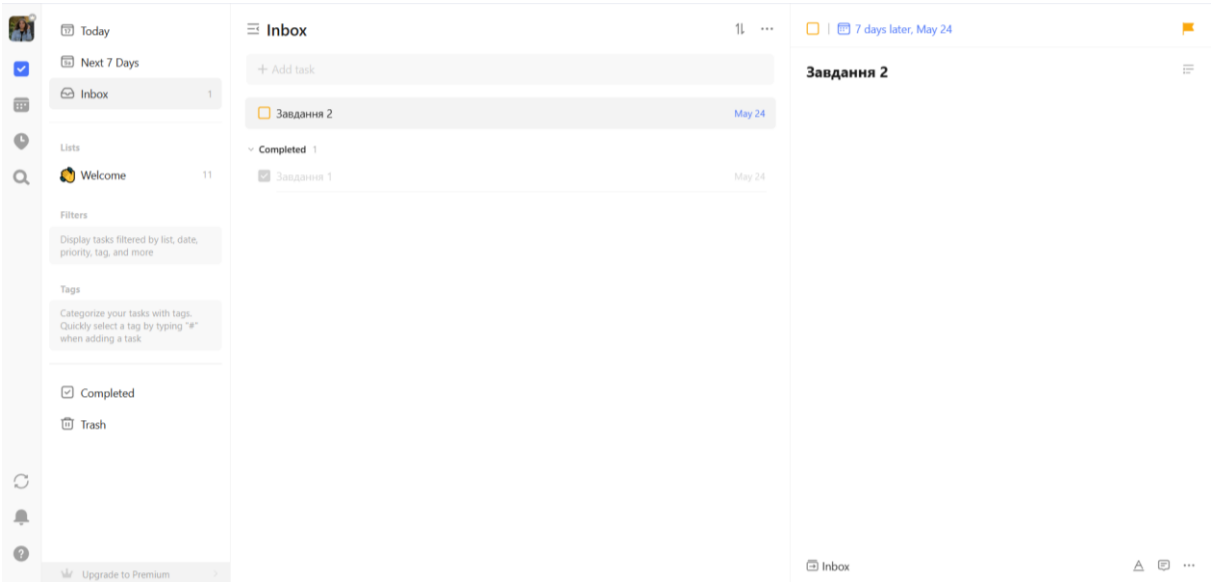


Рисунок 1.5 – Список завдань у TickTick

Трекер звичок в TickTick поданий у формі поточного тижня, де відмічено дні, в які ту чи іншу звичку було дотримано. Також у вебдодатку можна побачити загальний аналіз за місяць, що демонструє успішне впровадження нового способу життя (рис. 1.6).

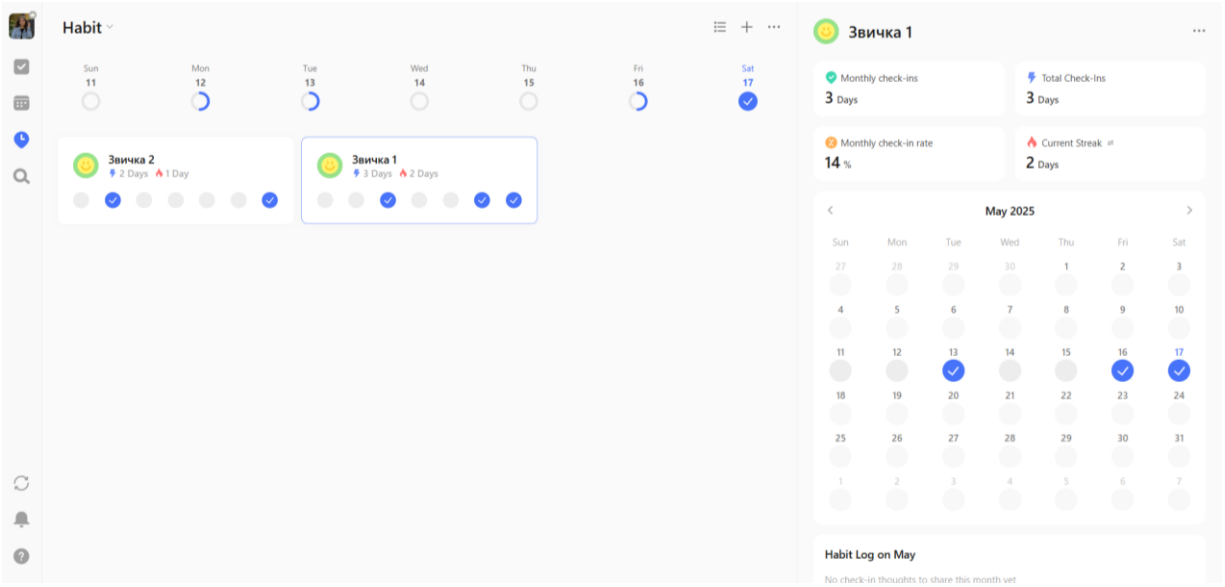


Рисунок 1.6 – Трекер звичок у TickTick

Таким чином, TickTick надає прості та ефективні засоби для відстежування звичок та виконання завдань. Наявність фільтрації за різними категоріями, додаткові способи розподілу задач і статистика продуктивності спрощують роботу, підвищують рівень мотивації та допомагають зосередитися. Оскільки TickTick орієнтується на завдання, розклад та звички, контроль бюджету у вебдодатку відсутній, що зумовлює необхідність пошуку альтернативних рішень.

Завдяки аналізу таких аналогів, як Notion та TickTick, можна зробити висновок, що кожен із цих вебдодатків пропонує зручні інструменти для планування, але не має повноцінного вбудованого механізму для щоденного обліку особистого бюджету та цілісного поєднання всіх трьох функцій – фінансів, завдань і звичок – в одній системі. Порівняння вебдодатків за ключовими функціональними критеріями подано в табл. 1.1.

Таблиця 1.1 – Порівняння функціональних можливостей вебдодатків-аналогів

№ з/п	Критерій	Notion	TickTick
1	Планування бюджету	Є необхідні інструменти для самостійного створення та шаблони	Немає
2	Трекер звичок	Є необхідні інструменти для самостійного створення та шаблони	Календар та аналітика для різних звичок
3	Управління завданнями	Списки, Kanban-дошки та інші інструменти з додатковими налаштуваннями	Списки і Kanban-дошки з додатковими налаштуваннями
4	Простота налаштування	Самостійне налаштування з використанням доступних інструментів	Мінімум налаштувань
5	Візуалізація даних	Графіки, діаграми тощо	Статистика за звичками
6	Зберігання даних	Хмарне, експорт/імпорт	Хмарне, експорт/імпорт

Аналіз переваг та недоліків наявних аналогів та розгляд їх функціональних можливостей дає змогу сформулювати вимоги до проєктованого вебдодатку з врахуванням особливостей складових самоорганізації: керування завданнями, контролю бюджету та відстежування звичок. Їх поєднання в цілісній системі зі зрозумілим інтерфейсом та збереженням даних користувача про обліковий запис, фінансові операції, актуальні задачі та історію виконання звичок з визначеною метою стане основою для подальшої розробки.

1.3 Постановка задачі

Ціллю створення вебдодатку є підвищення особистої ефективності користувача шляхом об’єднання в одному цифровому інструменті трьох важливих складових самоорганізації: планування бюджету, керування завданнями та формування корисних звичок. Такий підхід дозволяє уникнути фрагментації даних у кількох різних застосунках, забезпечуючи зручність, централізованість і цілісність особистого планування.

Основними завданнями для досягнення визначеної мети є:

– розробити функціональні модулі для ведення та аналізу особистого бюджету, включаючи управління накопиченнями;

– забезпечити гнучкий інтерфейс для створення, моніторингу та управління завданнями з урахуванням термінів і їх стану;

– реалізувати модуль трекінгу звичок зі встановленням цілей на місяць;

– забезпечити захист паролів користувачів;

– розробити зручний і зрозумілий інтерфейс користувача, що поєднує всі функції у єдину систему.

Для реалізації вебдодатку буде використано мову програмування Python і фреймворк Django, що зумовлено наявністю широких можливостей для швидкої розробки, вбудованих інструментів для роботи з базами даних, формами та автентифікацією. Крім того, це забезпечить надійний захист паролів корситувачів із застосування сучасних методів хешування. База даних SQLite дозволить зберігати дані та виконувати запити без необхідності налаштування окремого серверного середовища. HTML і CSS стануть засобами для створення інтерфейсу, що забезпечить зрозумілу навігацію.

Результатом стане вебдодаток, який дозволить користувачу ефективно планувати особистий бюджет, керувати завданнями та формувати корисні звички в одному середовищі. Це забезпечить підвищення продуктивності та організованості, зменшить час на перемикання між різними додатками та сприятиме досягненню поставлених цілей завдяки зручному інтерфейсу та гнучкому функціоналу для аналізу і контролю особистих фінансів і щоденних справ.

Висновки до розділу

У першому розділі було обґрунтовано актуальність розробки вебдодатку, який поєднує три ключові напрями особистої ефективності: планування бюджету, керування завданнями та формування корисних звичок.

Такий підхід дозволяє вирішити проблему фрагментації особистих даних у різних сервісах, зменшити навантаження на користувача та сприяти підвищенню продуктивності завдяки єдиному середовищу управління особистими ресурсами.

Було проаналізовано наявні аналоги у сфері самоорганізації та розглянуто їх функціональні можливості, що дозволило виявити відсутність одночасного поєднання контролю фінансових операцій, завдань та звичок. Це, зі свого боку, дозволило визначити напрямок для подальшої розробки.

У результаті було сформульовано мету та основні завдання проєкту, що передбачають реалізацію функціональних модулів для роботи з фінансами, завданнями та звичками, а також гарантування безпеки даних і зручного інтерфейсу. Обрані методи реалізації дозволяють створити надійний та зручний вебдодаток, що є основою для ефективного виконання наступних етапів проєктування і реалізації системи.

РОЗДІЛ 2
ІНФОРМАЦІЙНЕ ТА МАТЕМАТИЧНЕ ЗАБЕЗПЕЧЕННЯ

2.1 Аналіз предметної області

Система самоорганізації охоплює кілька важливих складових контролю особистого часу та фінансів: управління бюджетом, формування звичок та відстежування виконання завдань.

У межах предметної області можна виокремити ряд об’єктів дослідження, властивості яких продемонстровано в табл. 2.1.

Таблиця 2.1 – Об’єкти дослідження

№ з/п	Об’єкт дослідження	Опис	Основні атрибути
1	Користувач	Зареєстрований користувач системи, який має обліковий запис і взаємодіє з додатком	id, логін, пароль (у вигляді хешу), активність, дата реєстрації, останній вхід
2	Завдання	Особисті завдання користувача, які потрібно виконати	id, назва, опис, термін виконання, стан виконання, користувач
3	Звичка	Звички, які формує користувач та відстежує їх дотримання	id, назва, дата початку, дата досягнення поставленої мети, цільова мета (кількість днів), користувач
4	День звички	Інформація про виконання або пропуск звички в конкретний день	id, дата, стан виконання, звичка
5	Бюджетна операція	Записи доходів або витрат користувача, що відображають фінансові операції	id, сума, дата, тип операції (дохід чи витрата), категорія, користувач
6	Категорія бюджету	Категорії доходів або витрат для класифікації фінансових операцій	id, назва категорії, користувач
7	Накопичення	Записи про накопичені користувачем кошти	id, назва, сума накопичення, дата, користувач

2.1.1 Вхідні дані

Користуючись вебдодатком, користувач залишає вхідні дані, які можна віднести до кількох ключових категорій:

– обліковий запис;

– модель бюджету;

– модуль звичок;

– модуль завдань.

Під час реєстрації облікового запису користувач вводить логін і пароль, які система запам’ятовує та в подальшому, у процесі авторизації, перевіряє їх відповідність.

Керування бюджетом включає інформацію про рух коштів. Зокрема, після вибору місяця і року заповнюються такі поля, як категорія фінансової операції, її сума, дата здійснення та тип – витрата чи дохід. Також користувач має можливість створювати нові категорії бюджету та залишати деталі про накопичення (дату, суму і назву заощадження).

Трекінг звичок розпочинається зі створення нової звички, де необхідно вказати її назву та цільову мету – кількість днів, в які вона повинна бути виконана. З цього моменту щодня є можливість відмічати успішне дотримання за поточною датою або ж залишати пропуск.

Для створення нового завдання користувачу необхідно вказати його назву, кінцевий термін виконання та за бажанням залишити опис, також наявна можливість відмітити стан як «Виконано» відразу або згодом.

Обмеження на вхідні дані включають:

– під час реєстрації можна використовувати лише унікальний логін, який не повторюється в системі, довжиною не більше 150 символів;

– пароль повинен відповідати мінімальним вимогам безпеки, зокрема містити не менше 8 символів та не бути повністю числовим;

– назва завдання не може містити більше 200 символів;

– назва категорії бюджету не може містити більше 100 символів;

– сума витрат, доходів чи накопичень може містити до 10 цифр, 2 з яких – після коми;

– назва звички може містити не більше 100 символів;

– кількість днів, відведених для набуття звички, повинна бути цілим додатним числом, за замовчуванням дорівнює 100.

2.1.2 Вихідні дані

На основі вхідних даних, які залишає користувач, в системі формуються вихідні дані за трьома ключовими аспектами вебдодатку: бюджетом, завданнями та звичками.

Вихідні дані в бюджеті включають список фінансових операцій, складений з огляду на залишену користувачем інформацію про витрати та доходи за різними категоріями за обраний період. Схожим чином відображаються й деталі про накопичення, які містять назву, дату й суму операцій. Для обох складових бюджету формується загальний підсумок по витратам, доходам та заощадженням. Для користувача відображаються актуальний баланс за обраний місяць, сума доходів, витрат та загальний підсумок по накопиченням.

На основі залишених деталей про завдання користувача в системі формується список з відображенням активних та завершених задач. У разі прострочення терміну дата змінить колір на червоний.

Створені звички відображаються у вигляді таблиці з назвою, датою початку, прогресом у вигляді кількості успішно дотриманих днів із цільової мети та позначки виконання для поточної доби. Також присутня можливість перегляду історії звички для відстеження її дотримання за місяць. У разі досягнення цільової мети звичка переміщується до набутих, де додатково відображається дата фінішу та пропонується продовжити її виконання, зазначивши нову кількість запланованих днів дотримання.

Таким чином, вихідні дані охоплюють інформацію про фінансові операції, статус і прогрес виконання завдань, а також відстеження формування звичок.

2.2 Проєктування системи

Проєктування системи самоорганізації включає кілька важливих етапів, зокрема проєктування бази даних, яка зберігатиме дані користувачів, та класів,

які буде використано для програмної реалізації вебдодатку для контролю бюджету, трекінгу звичок та управління завданнями.

2.2.1 Проєктування бази даних

Для збереження даних користувачів вебдодатку для самоорганізації, було обрано систему керування базами даних SQLite [7], що пов’язано з рядом переваг: вона однофайлова, безкоштовна, кросплатформена, легко інтегрується та не потребує адміністрування [8, с. 277]. Таким чином, усі таблиці з даними про користувачів, бюджет, завдання та звички будуть зібрані в одному файлі, що спрощує перенесення, резервне копіювання та розгортання проєкту на різних пристроях.

База даних налічує 7 таблиць, взаємозв’язки яких відображено на рис. 2.1.

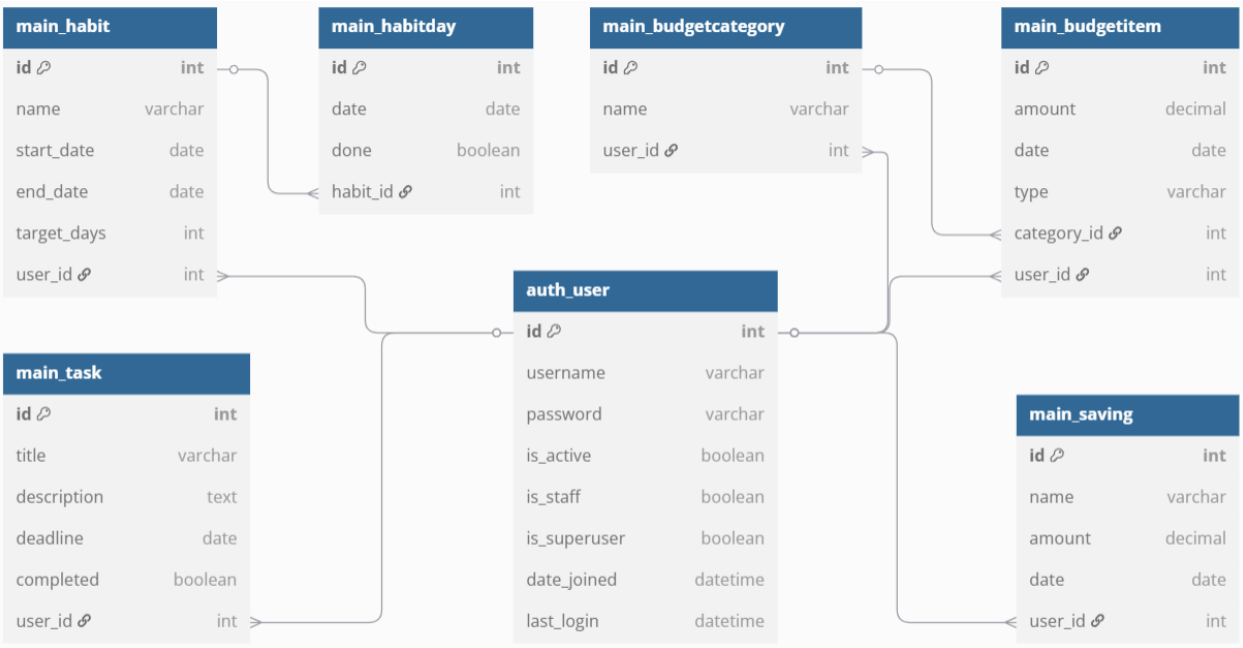


Рисунок 2.1 – ER-діаграма структури бази даних

Призначенням auth_user є збереження даних про зареєстрованого користувача, що включає ряд полів:

– id – унікальний ідентифікатор користувача;

– username – унікальне ім’я користувача для входу в систему;

– password – хеш пароля користувача;

– is_active – позначка активності облікового запису;

– is_staff – позначка працівника, який може заходити в адміністративну панель;

– is_superuser – позначка адміністратора;

– date_joined – дата створення облікового запису;

– last_login – дата і час останнього входу користувача.

Усі таблиці, за винятком main_habitday, поєднані з auth_user зв’язком типу «один-до-багатьох», оскільки кожна з них може містити кілька записів для одного користувача.

Завдання зберігаються до таблиці main_task, полями якої є:

– id – унікальний ідентифікатор завдання;

– title – назва завдання;

– description – опис змісту завдання;

– deadline – строк виконання;

– completed – логічне поле, що вказує на статус виконання;

– user_id – зовнішній ключ, який пов’язує завдання з конкретним користувачем у таблиці auth_user.

Для реалізації модуля звичок є необхідними дві таблиці, одна з яких – main_habit. Вона зберігає всю основну інформацію про створені користувачем звички, за що відповідають наступні поля:

– id – унікальний ідентифікатор звички;

– name – назва звички;

– start_date – дата початку формування звички;

– end_date – дата завершення, що свідчить про набуття звички;

– target_days – кількість днів, що є цільовою метою для набуття звички;

– user_id – зовнішній ключ, що вказує на користувача, якому належить звичка (зв’язок із таблицею auth_user).

Таблиця main_habitday зберігає дані про дотримання звички, де:

– id – унікальний ідентифікатор для позначення запису дня звички;

– date – дата, в яку відбулося дотримання звички;

– done – булевий показник, який свідчить про успішне дотримання звички;

– habit_id – зовнішній ключ, що вказує на відповідну звичку в таблиці main_habit.

Таблиці main_habit і main_habitday поєднані зв’язком типу «один-до-багатьох», оскільки одній звичці може відповідати багато записів днів її дотримання.

Ключовою для контролю бюджету є таблиця main_budgetitem, яка містить поля:

– id – унікальний ідентифікатор запису про зміни в бюджеті;

– amount – сума доходу або витрати;

– date – дата операції;

– type – тип операції (дохід чи витрата);

– category_id – ідентифікатор категорії бюджету;

– user_id – ідентифікатор користувача, якому належить запис.

Таблиця main_budgetcategory зберігає всі категорії доходів та витрат:

– id – унікальний ідентифікатор категорії;

– name – назва категорії;

– user_id – ідентифікатор користувача, який створив категорію.

Таблиці main_budgetcategory і main_budgetitem поєднує зв’язок типу «один-до-багатьох», тому що на одну категорію може припадати кілька доходів чи витрат.

Попри те, що накопичення безпосередньо пов’язані з фінансами користувача, дані про них зберігаються в окремій таблиці під назвою main_saving, що дає змогу відстежувати їх окремо від доходів та витрат. Поля є наступними:

– id – унікальний ідентифікатор запису про накопичення;

– name – назва накопичення;

– amount – сума нарахованих до накопичень коштів;

– date – дата здійснення запису;

– user_id – ідентифікатор користувача, якому належить накопичення.

Така база даних забезпечить централізоване зберігання та доступ до всіх ключових даних користувача, а структура зв’язків між таблицями – ефективне управління бюджетом, звичками та завданнями.

2.2.2 Проєктування об’єктно-орієнтованої моделі

У проєктуванні об’єктно-орієнтованої моделі системи самоорганізації важливо розглянути класи основні класи, що відповідають за зберігання та обробку даних користувача.

Основними класами системи є User, Task, Habit, HabitDay, BudgetItem, BudgetCategory та Saving, структуру яких продемонстровано на рис. 2.2.

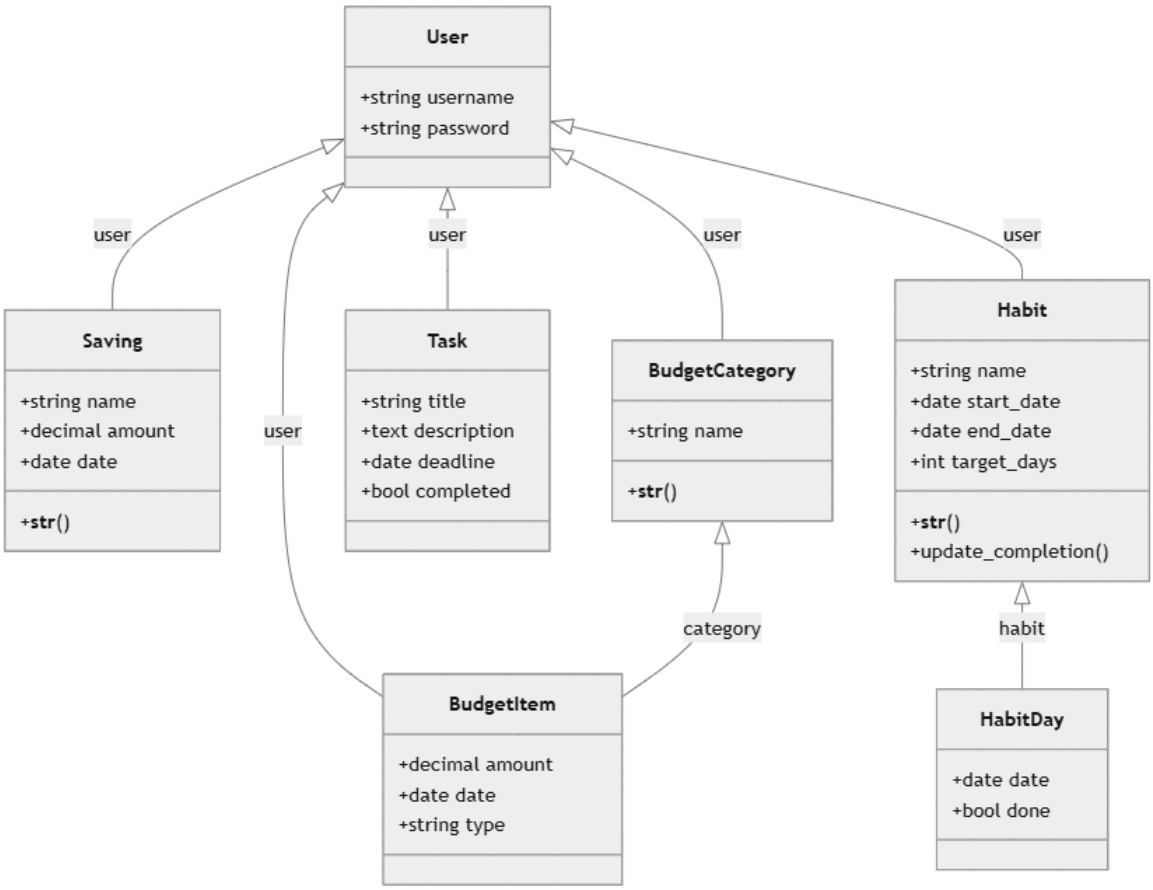


Рисунок 2.2 – UML-діаграма класів, що відображає структуру основних об’єктів системи

Клас User відповідає за збереження інформації користувача системи. Зокрема, його ключовими атрибутами є логін (username) та пароль (password), які забезпечують доступ до вебдодатку і використовуються під час автентифікації та авторизації.

Клас Task створений для завдань користувача, його атрибутами є назва (title), опис (description), термін (deadline) та статус виконання (completed). Він дозволяє зберігати інформацію про активні та завершені завдання, а також відслідковувати прострочення термінів.

Клас Saving призначений для збереження даних про накопичення користувача, його атрибути включають назву (name), суму (amount) і дату операції (date). Крім того, у класі реалізовано метод __str__(), який повертає текстове представлення об’єкта у вигляді «назва – сума», що спрощує відображення інформації в інтерфейсі.

Клас BudgetItem відображає записи бюджету користувача. Атрибутами цього класу є сума (amount), дата (date) і тип операції (type) – дохід чи витрата. BudgetItem пов’язаний з класом BudgetCategory через зовнішній ключ, що дозволяє класифікувати фінансові операції за категоріями.

Клас BudgetCategory зберігає створені користувачем категорії доходів і витрат, він має атрибут назва категорії (name) та використовує метод __str__(), який повертає назву категорії для зручності відображення.

Клас Habit відповідає за звички користувача, він налічує такі атрибути, як назва звички (name), дата початку (start_date), дата завершення (end_date), а також цільову мету для набуття звички у вигляді кількості днів дотримання (target_days). У класі використовуються методи __str__(), який повертає назву звички, та update_completion(), що оновлює дату завершення звички у разі досягнення цільового показника.

З Habit пов’язаний клас HabitDay, що відповідає за відстеження дотримання звички за днями. Даний клас має два атрибути: дату (date) та позначку про виконання (done).

Усі перелічені класи, за винятком HabitDay, мають зв’язок з User, оскільки кожен об’єкт цих класів належить конкретному користувачу системи.

У системі також присутні класи форм, які створюються для введення та валідації даних користувача про бюджет, завдання та звички, вони наведені на рис. 2.3.

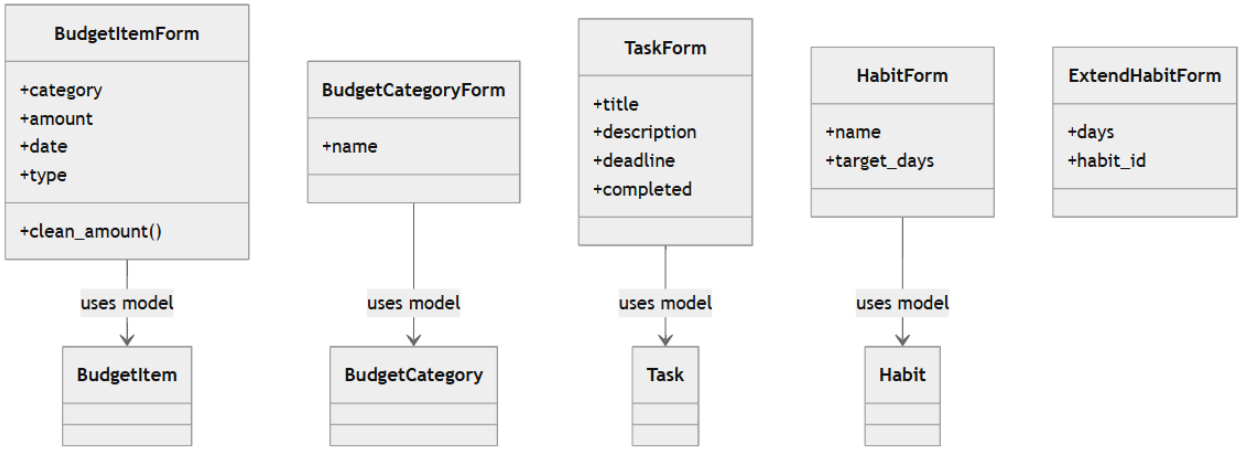


Рисунок 2.3 – UML-діаграма класів для форм введення та валідації даних у системі

Клас BudgetItemForm відповідає за введення та валідацію даних про фінансові операції користувача. Він містить поля для вибору категорії (category), дати (date), типу операції (type) та зазначення її суми (amount). Для забезпечення коректності даних у формі за допомогою методу clean_amount() реалізовано перевірку на те, що сума не може бути від’ємною. Цей клас має прямий зв’язок із моделлю BudgetItem, оскільки використовує її атрибути для відображення та обробки відповідних полів форми.

Клас BudgetCategoryForm призначений для створення категорій доходів та витрат, назва яких є його атрибутом (name). Цей клас формується на основі BudgetCategory, завдяки чому отримує можливість працювати з даними категорій бюджету, що належать користувачу.

Клас TaskForm створений для введення завдань користувача, він містить поля для назви (title), опису (description), кінцевого терміну (deadline) та статусу виконання (completed). Для зручності введення дати передбачено

календарний віджет. Зв’язок класу TaskForm з Task дає можливість автоматично створювати відповідні поля на основі атрибутів моделі, що забезпечує узгодженість між формою та збереженням даних у базі

Клас HabitForm відповідає за створення нових звичок, він включає поля назва (name) та мета у вигляді кількості днів (target_days). Цей клас пов’язаний з Habit, що дає змогу відображати інформацію про звички користувача та зберігати нові або змінені дані в базі.

Клас ExtendHabitForm призначений для збільшення кількості днів для набуття звички у разі досягнення першочергової мети. Він містить поле кількість днів (days) для встановлення нової цілі та приховане поле з ідентифікатором звички (habit_id).

2.3 Математичне та алгоритмічне забезпечення

2.3.1 Математичне забезпечення

Обробка даних користувачів у системі самоорганізації включає кілька математичних операцій, необхідних для фіксування актуального стану бюджету.

Для кожного користувача підсумок за доходами та витратам за обраний місяць формується як сума всіх операцій за одним із типів. Таким чином, математичну модель для підрахунку підсумку за доходами можна відобразити у вигляді (2.1):

$$P(u,m) = \sum_{i \in I_P(u,m)} i \qquad (2.1)$$

де $P(u,m)$ – сума доходів користувача u за місяць m;

$I_P(u,m)$ – множина всіх операцій доходів користувача u за місяць m;

i – сума операції;

$i \geq 0$.

Підрахунок суми за витратами можна показати за допомогою формули (2.2):

$$E(u,m) = \sum_{i \in I_E(u,m)} i \tag{2.2}$$

де $E(u,m)$ – сума витрат користувача u за місяць m;

$I_E(u,m)$ – множина всіх операцій витрат користувача u за місяць m.

Для обчислення балансу користувача враховуються витрати та доходи за поточний місяць, а також баланс за попередній місяць, це можна відобразити у вигляді формули (2.3):

$$B(u,m) = B(u,m-1) + P(u,m) - E(u,m) \tag{2.3}$$

де $B(u,m)$ – баланс користувача u за місяць m;

$1 \leq m \leq 12$.

Накопичення користувача містять записи про поповнення та виведення коштів з рахунку, а загальна сума є результатом додавання всіх операцій, пов’язаних з заощадженнями за поточний місяць, та врахуванням накопичень за попередній період. Математично це можна відобразити в такому вигляді (2.4):

$$S(u,m) = S(u,m-1) + \sum_{a \in A(u,m)} a \tag{2.4}$$

де $S(u,m)$ – сума накопичень користувача u за місяць m;

$A(u,m)$ – множина всіх операцій накопичень користувача u за місяць m;

a – сума операції;

$a \geq 0$.

2.3.2 Алгоритмічне забезпечення

Для доступу до вебдодатку користувач повинен увійти в систему, якщо обліковий запис уже наявний, або ж здійснити реєстрацію і створити новий. Це передбачає введення логіну та паролю, який за допомогою криптографічної хеш-функції перетворюється на хеш – незворотне значення фіксованої довжини – та зберігається в базі даних у зашифрованому вигляді. З метою підвищення безпеки перед застосуванням алгоритму до паролю додається сіль.

Отже, вхід користувача в систему можливий тільки за наявності облікового запису. Під час авторизації перевірка супроводжується порівнянням хешу паролю, обчисленого на основі введених даних, з тим, що збережено в базі даних, завдяки чому забезпечується захист від несанкціонованого доступу.

Алгоритм авторизації та реєстрації користувача відображено на схемі в нотації BPMN (рис. 2.4).

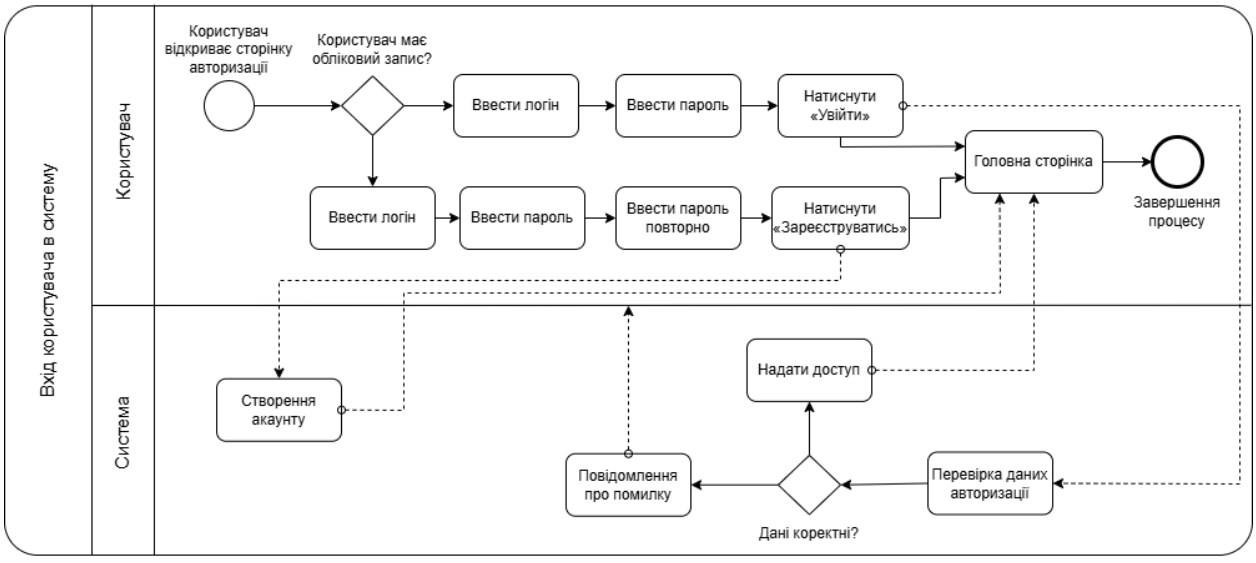


Рисунок 2.4 – BPMN-схема алгоритму входу користувача в систему

Керування бюджетом включає взаємодію з витратами та доходами, зокрема створення нових записів, в яких вказуються сума, дата, категорія й тип, та перегляд наявної інформації за обраний місяць. Крім того, користувач має можливість додати нову категорію фінансових операцій.

Алгоритм взаємодії користувача з доходами та витратами показано на рис. 2.5.

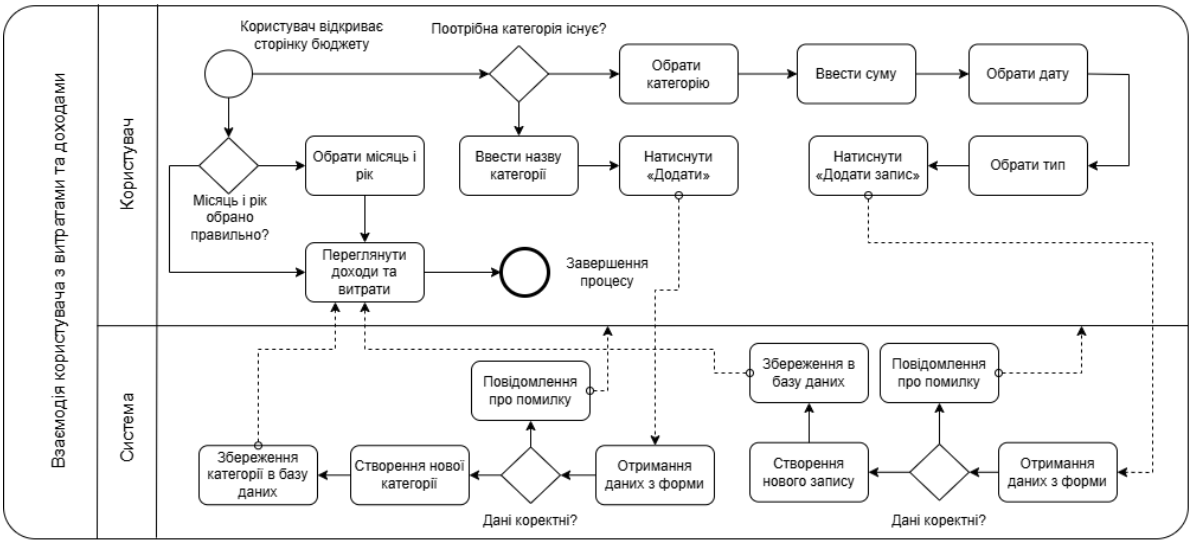


Рисунок 2.5 – BPMN-схема алгоритму взаємодії з витратами чи доходами

Керування накопиченнями також є складовою бюджету, що регулюється користувачем окремо. Зокрема, для створення нового запису про заощадження необхідно ввести назву, суму, обрати дату та вид операції: додавання чи виведення коштів. Алгоритм взаємодії користувача з накопиченнями продемонстровано на рис. 2.6.

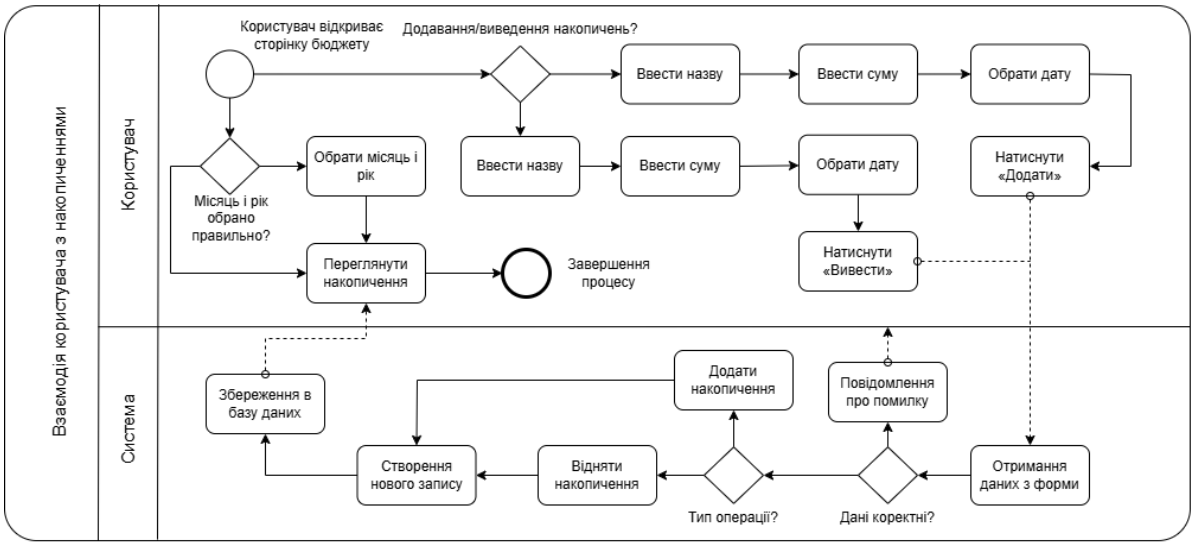


Рисунок 2.6 – BPMN-схема алгоритму взаємодії з накопиченнями

Одним із аспектів самоорганізації є трекінг власних звичок, які користувач може створювати, вказуючи назву та мету – кількість днів для набуття. Це відкриває можливості для регулярного відмічання статусу виконання та відстеження прогресу у вигляді історії за місяць. Набуті звички можна не лише переглядати, а й продовжувати їх дотримання, вказуючи наступну ціль.

Алгоритм взаємодії користувача зі звичками наведено на рис. 2.7.

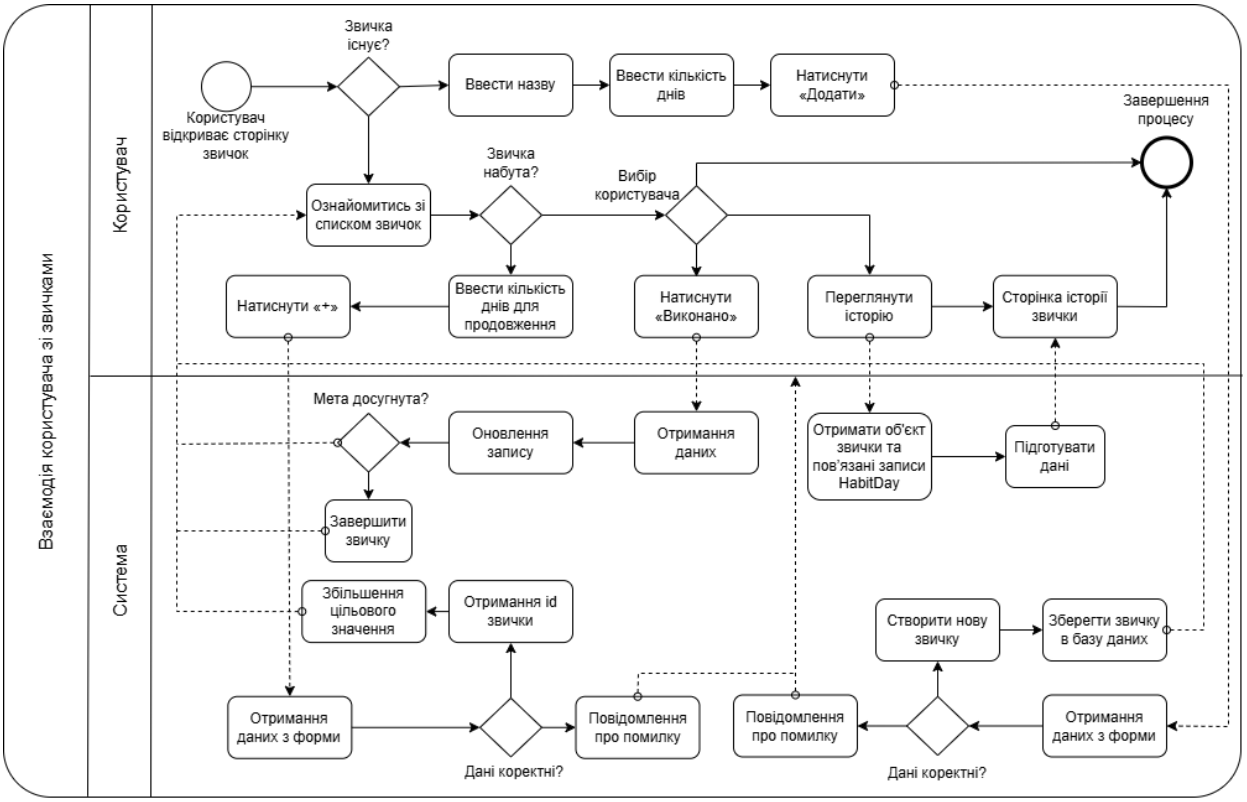


Рисунок 2.7 – BPMN-схема алгоритму взаємодії зі звичками

У процесі створення нових завдань користувач має можливість вказати назву, опис, кінцевий термін і їх стан. Також взаємодія з цією складовою вебдодатку включає відслідковування виконаних та активних задач, які можна видалити або позначити як виконані.

Алгоритм взаємодії користувача з завданнями відображено на рис. 2.8.

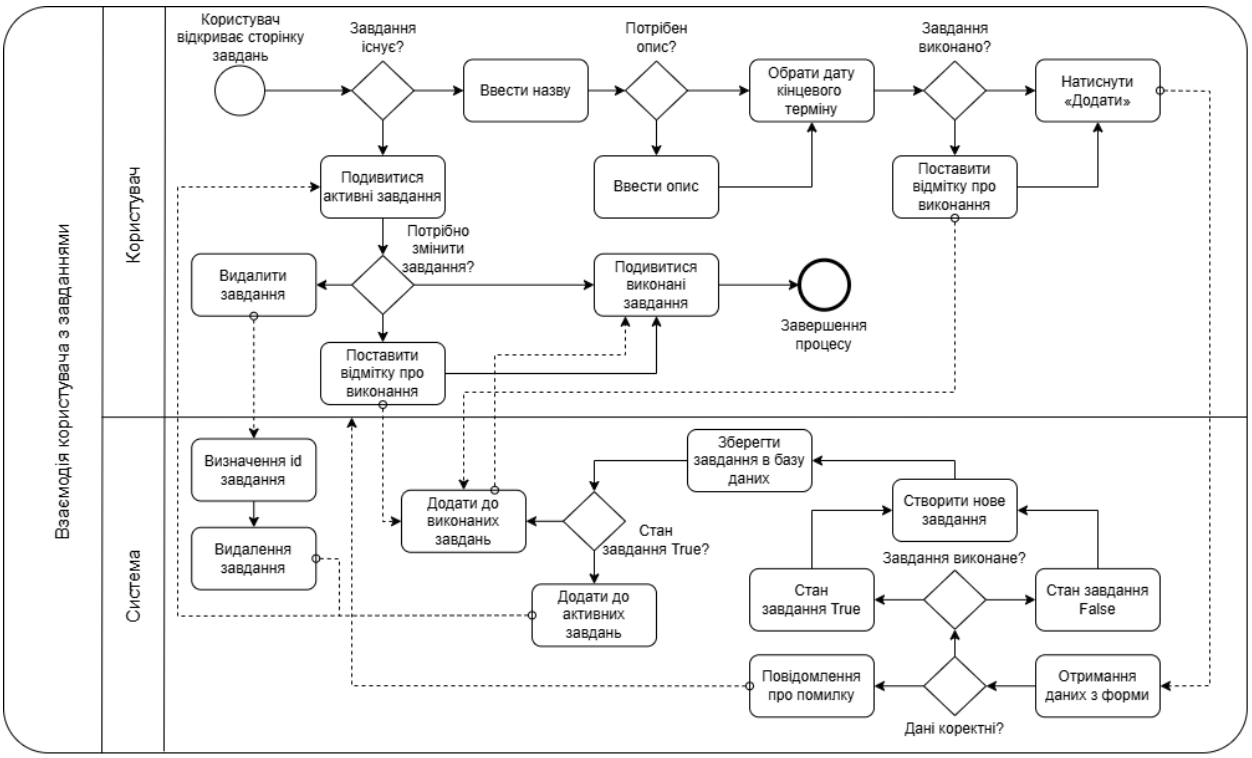


Рисунок 2.8 – BPMN-схема алгоритму взаємодії з завданнями

Таким чином, алгоритмічне забезпечення охоплює найважливіші складові взаємодії користувача з вебдодатком для самоорганізації. Перш за все, відбувається процес реєстрації, якщо обліковий запис ще не існує, в інакшому випадку проходить авторизація. Цей крок відкриває доступ до основних функціональних можливостей:

- контроль доходів та витрат;
- керування накопиченнями;
- створення та відстеження дотримання звичок;
- управління завданнями.

Висновки до розділу

У рамках розділу було проведено аналіз предметної області, що охоплює виокремлення об’єктів дослідження, а також розгляд вхідних та вихідних даних і наявних обмежень. Було з’ясовано, що ключовими сутностями є:

користувач, завдання, звичка, день звички, бюджетна операція, категорія бюджету та накопичення – кожна з яких має власні атрибути. Вхідні дані являють собою інформацію, залишену користувач під час взаємодії з вебдодатком у таких процесах, як реєстрація, створення записів у бюджеті, накопиченнях, завданнях та звичках. Вихідними даними є оброблена інформація, яка представлена у вигляді списків про фінансові операції та активні чи виконані завдання, історій звичок.

У ході проєктування системи було розглянуто застосування бази даних SQLite для створення таблиць і їх полів, які відображено разом зі зв’язками. Також було досліджено ключові класи об’єктів та форм введення й валідації, перелічено атрибути, методи та їх взаємодію.

Математичне забезпечення включало розгляд ключових формул для розрахунку складових бюджету: витрат, доходів, балансу й накопичень. У ході дослідження алгоритмічного забезпечення було продемонстровано схеми в нотації BPMN для відображення взаємодії користувача з вебдодатком. Зокрема, розглянуто процес входу в систему, що передбачає реєстрацію та авторизацію, керування звичками та завданнями, створення записів доходів, витрат та накопичень.

Розглянуті кроки стануть основою для подальшої програмно реалізації вебдодатку для самоорганізації.

РОЗДІЛ 3
ПРОГРАМНЕ ТА ТЕХНІЧНЕ ЗАБЕЗПЕЧЕННЯ

3.1 Засоби розробки

Для створення вебдодатку для самоорганізації обрано мову програмування Python [9], що пов’язано з простими синтаксисом й читабельністю. Вона налічує велику кількість бібліотек, що відкриває можливості для створення стабільного backend-застосунку.

У проєкті використовуються фреймворк для веброзробки Django [10], який забезпечує реалізацію всієї серверної логіки, організацію моделей, форм, маршрутизацію та обробку запитів користувача. З його допомогою створено механізми автентифікації та взаємодії з базою даних.

Для збереження даних користувачів застосовується вбудована реляційна база даних SQLite. Вона використовується в Django за замовчуванням і в проєкті відіграє важливу роль у збереженні інформації про облікові записи, завдання, звички та їх історію і записи бюджету й накопичень.

Для створення шаблонів інтерфейсу користувача застосовується HTML, а оформлення зовнішнього вигляду реалізовується за допомогою CSS. Використання цих засобів дозволяє створити зрозумілий, зручний для взаємодії та привабливий інтерфейс.

Програмне забезпечення для створення вебдодатку для контролю бюджету, трекінгу звичок та управління завданнями включає Visual Studio Code [11], що є основним середовищем розробки. Такий вибір обґрунтовано підтримкою розширень, зокрема для Django, підсвіткою синтаксису й автодоповненням коду.

Взаємодія з базою даних SQLite відбувається в програмі DB Browser [12], яка надає можливість перегляду всіх таблиць та редагування їх вмісту.

3.2 Вимоги до технічного та програмного забезпечення

Вимоги до технічного забезпечення:

– ПК або ноутбук з операційною системою Windows/Linux/macOS;
– оперативна пам’ять (ОЗП) не менше 4 ГБ;
– процесор не нижче Intel Core i3 або AMD Ryzen 3.

Середовищем розробки є Visual Studio Code, який охоплює роботу з програмним кодом та підтримує розширення для фреймворку Django. Для коректної роботи системи розробки необхідно мати встановлену мову програмування Python версії не нижче 3.10, а також фреймворк Django версії 4.x або вище.

Також вимоги до програмного забезпечення для можливості запуску вебдодатку включають наявність операційної системи Windows, Linux чи macOS та браузера Google Chrome чи, наприклад, Firefox.

Проєкт працює локально завдяки вбудованому серверу розробки Django, що не потребує хостингу. Таким чином, допуск до вебдодатку забезпечується через тимчасовий локальний вебсервер, який запускається командою в терміналі, і відкривається у браузері за адресою.

Процес запуску проєкту відображено на рис. 3.1.

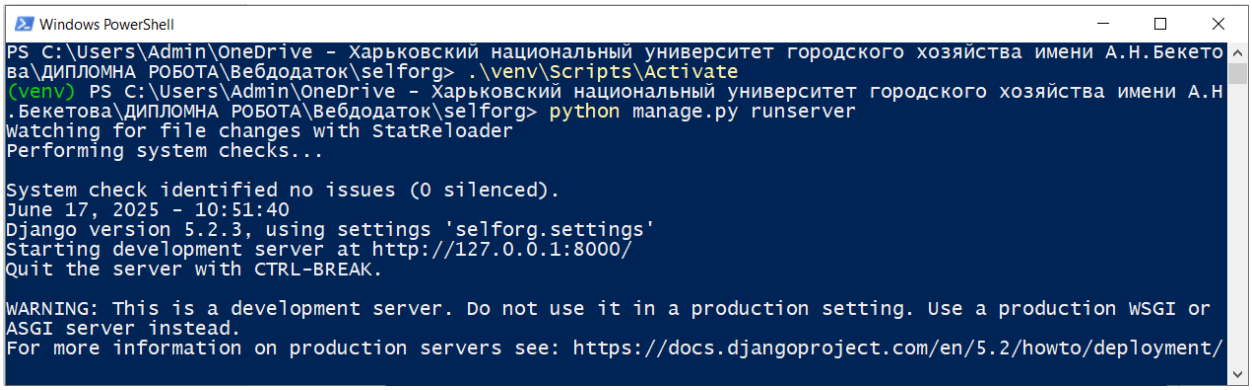


Рисунок 3.1 – Процес розгортання тимчасового вебсервера

3.3 Опис програмної реалізації

Проєкт реалізовано у вигляді Django-додатку з чіткою модульною структурою. Його директорія містить необхідні компоненти для локального запуску вебдодатку та його коректної роботи, зокрема:

– db.sqlite3 – файл бази даних SQLite, що зберігає всю інформацію про користувачів, завдання, бюджет і звички;

– manage.py – головний скрипт керування Django-проєктом;

– main – папка, що містить ключові файли логіки проєкту, зокрема, для роботи з формами введення даних, запитами користувачів, маршрутизацією та описом моделей бази даних;

– selforg – конфігураційний пакет проєкту;

– static – папка, в якій зберігається фонове зображення;

– templates – папка, яка містить HTML-шаблони для інтерфейсу користувача, що включають сторінки реєстрації, авторизації, бюджету, завдань, звичок та історії їх дотримання;

– venv – використовується для запуску вебдодатку.

Процес програмної реалізації можна поділити на два ключових етапи: створення бази даних та реалізація логіки вебдодатку.

3.3.1 Створення бази даних

Створення бази даних відбувається після застосування команди міграції та оголошення моделей у файлі models.py, згідно з якими формуються відповідні SQL-таблиці. Вони описуються у вигляді класів з використанням методів Django ORM (Object-Relational Mapping), що дозволяють визначити структуру таблиць та типи полів. Зокрема, одними з таких методів є save() для збереження об’єктів до бази даних та delete() для їх видалення.

User є вбудованим класом фреймворку Django, він відповідає за управління обліковими записами користувачів, їх автентифікацію та

авторизацію. У проєкті він застосовується шляхом імпорту через пакет django.contrib.auth.models.

Складова бюджету у вебдодатку реалізовується за допомогою таких класів:

– Saving, що містить поля для таблиці з накопиченнями, які включають id користувача, назву заощадження, суму і дату операції;

– BudgetCategory, який містить поля для назви категорії фінансової операції та ідентифікатора користувача;

– BudgetItem, за допомогою якого в таблиці записів бюджету створюються поля id користувача, категорія витрати чи доходу, сума, дата і тип операції.

Програмну реалізацію класів, пов'язаних із бюджетом, відображено на рис. 3.2.

```python
class Saving(models.Model):
    user = models.ForeignKey(User, on_delete=models.CASCADE)
    name = models.CharField(max_length=100)
    amount = models.DecimalField(max_digits=10, decimal_places=2)
    date = models.DateField(default=date.today)

    def __str__(self):
        return f"{self.name} - {self.amount}"

class BudgetCategory(models.Model):
    name = models.CharField(max_length=100)
    user = models.ForeignKey(User, on_delete=models.CASCADE)

    def __str__(self):
        return self.name

class BudgetItem(models.Model):
    CATEGORY_TYPE = [('income', 'Дохід'), ('expense', 'Витрата')]
    user = models.ForeignKey(User, on_delete=models.CASCADE)
    category = models.ForeignKey(BudgetCategory, on_delete=models.CASCADE)
    amount = models.DecimalField(max_digits=10, decimal_places=2)
    date = models.DateField()
    type = models.CharField(choices=CATEGORY_TYPE, max_length=10)
```

Рисунок 3.2 – Структура моделей бази даних для бюджету

Клас Task створено для складової завдань у вебдодатку, він містить такі поля, як ідентифікатор користувача, назва та опис задачі, кінцевий термін та стан виконання (рис. 3.3).

```python
class Task(models.Model):
    user = models.ForeignKey(User, on_delete=models.CASCADE)
    title = models.CharField(max_length=200)
    description = models.TextField(blank=True)
    deadline = models.DateField()
    completed = models.BooleanField(default=False)
```

Рисунок 3.3 – Структура моделі бази даних для завдань

Для створення складової звичок застосовуються такі класи:

– Habit, що містить такі поля, як id користувача, назва звички, дата початку та завершення її дотримання, кількість днів, яка представляє мету, а також наявний метод для фіксування дати досягнення визначеної цілі;

– HabitDay, який відповідає за створення таблиці історії дотримання звички, він містить поля з назвою, датою та станом виконання.

Програмну реалізацію класів звичок наведено на рис. 3.4.

```python
class Habit(models.Model):
    user = models.ForeignKey(User, on_delete=models.CASCADE)
    name = models.CharField(max_length=100)
    start_date = models.DateField(auto_now_add=True)
    end_date = models.DateField(null=True, blank=True)
    target_days = models.PositiveIntegerField(default=100)

    def __str__(self):
        return self.name
    def update_completion(self):

        completed_days = self.habitday_set.filter(done=True).count()
        if completed_days >= self.target_days:
            latest = self.habitday_set.filter(done=True).order_by('-date').first()
            if latest:
                self.end_date = latest.date
                self.save()

class HabitDay(models.Model):
    habit = models.ForeignKey(Habit, on_delete=models.CASCADE)
    date = models.DateField()
    done = models.BooleanField(default=False)
```

Рисунок 3.4 – Структура моделей бази даних для звичок

Підключення до бази даних відбувається автоматично через налаштування у файлі settings.py, якийсь міститься в папці selforg. На рис. 3.5

продемонстровано частину коду, відповідальну за даний процес, із вказуванням шляху до db.sqlite3.

```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.sqlite3',
        'NAME': BASE_DIR / 'db.sqlite3',
    }
}
```

Рисунок 3.5 – Налаштування підключення до бази даних

Створену структуру бази даних включно з таблицями, полями та їхніми типами можна переглянути в програмі DB Browser (рис. 3.6). Також вона надає можливість редагувати наявні записи та створювати нові.

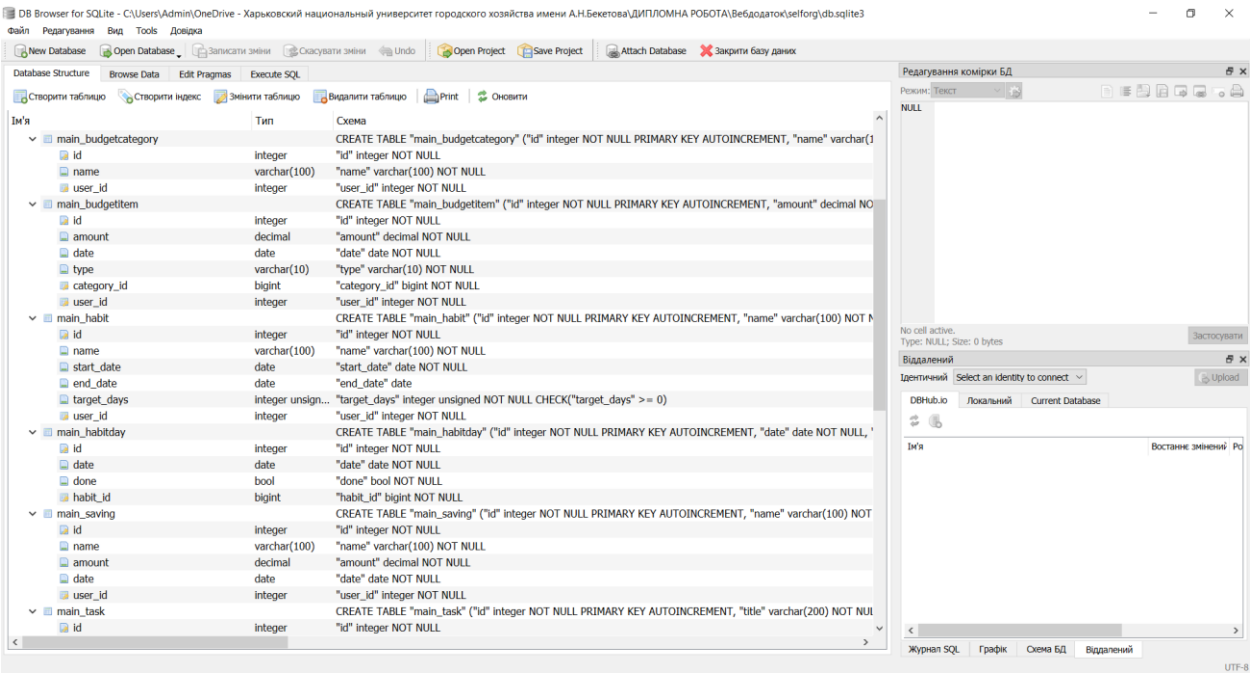


Рисунок 3.6 – Структура бази даних, відображена в програмі DB Browser

3.3.2 Реалізація логіки вебдодатку

Базова сторінка base.html виконує роль шаблону, який використовується для формування єдиного зовнішнього вигляду всіх сторінок вебдодатку. Вона

містить навігаційне меню з кнопкою входу або виходу та посиланнями на основні розділи: «Головна», «Бюджет», «Завдання» та «Звички».

Однією з переваг використання фреймворку Django є підвищення рівня безпеки, оскільки завдяки автоматичному генеруванню параметризованих SQL-запитів значно знижується ймовірність виникнення вразливостей, пов’язаних із SQL-ін’єкціями [13, с. 64].

До паролів користувачів застосовуються вимоги для підвищення безпеки, які включають мінімальну довжину, заборону надто простих або типових паролів, а також перевірку на подібність до імені користувача, що відображається на формі реєстрації (рис. 3.7)

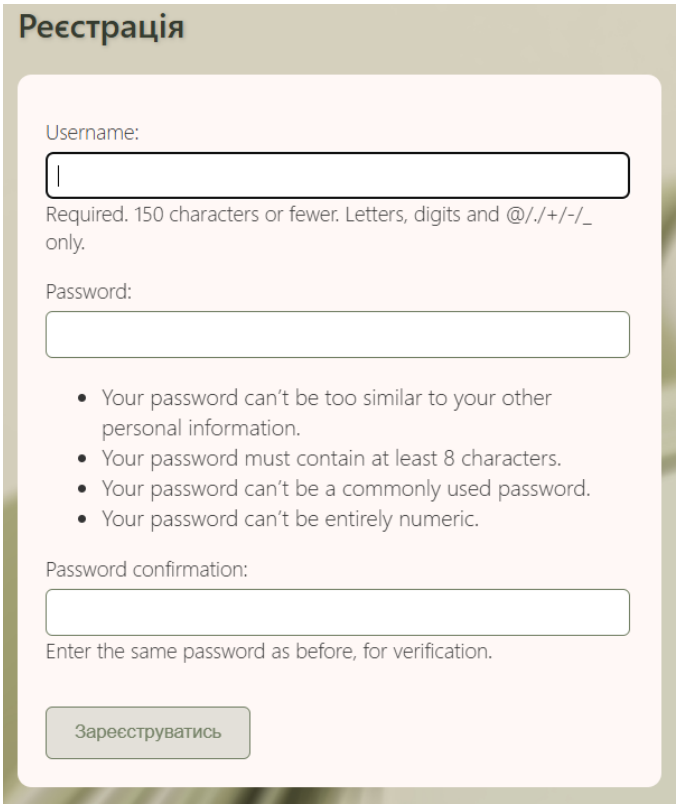


Рисунок 3.7 – Форма реєстрації користувача

Така перевірка паролів встановлюється у файлі settings.py за допомогою налаштування AUTH_PASSWORD_VALIDATORS, де визначено перелік вбудованих валідаторів (рис. 3.8).

```
AUTH_PASSWORD_VALIDATORS = [
    {
        'NAME': 'django.contrib.auth.password_validation.UserAttributeSimilarityValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.MinimumLengthValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.CommonPasswordValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.NumericPasswordValidator',
    },
]
```

Рисунок 3.8 – Налаштування валідаторів паролів

Крім того, паролі користувачів зберігаються не в оригінальному вигляді, а з застосуванням хеш-функції з додаванням солі, що зменшує небезпеку викрадення паролів у разі несанкціонованого доступу до бази даних. Це реалізовується автоматично в межах вбудованої системи автентифікації Django.

Таким чином, вхідними даними у процесі реєстрації є ім’я користувача, пароль та його підтвердження, а вихідними – створений обліковий запис у базі даних із захищеним паролем. Форма реєстрації реалізовується за допомогою UserCreationForm, що надається фреймворком Django, вона автоматично обробляє вхідні дані, перевіряє валідність та формує запис у базі даних. Обробку запиту на реєстрацію здійснює функція register(request), що міститься у views.py, вона передбачає автоматичний вхід користувача і перенаправлення на головну сторінку у разі коректності введених даних (рис. 3.9).

```
def register(request):
    if request.method == 'POST':
        form = UserCreationForm(request.POST)
        if form.is_valid():
            user = form.save()
            login(request, user)  # автологін після реєстрації
            return redirect('index')
    else:
        form = UserCreationForm()
    return render(request, 'register.html', {'form': form})
```

Рисунок 3.9 – Програмна реалізація обробки реєстрації користувача

Шаблон сторінки реєстрації містить елементи форми, які генеруються автоматично на основі визначеної форми у views-функції. Захист від

міжсайтової підробки запитів забезпечується через тег, який додає до форми прихований токен безпеки. Схожим чином реалізовується й форма для входу в систему, яка також містить захист CSRF та автоматично відображає необхідні поля для автентифікації користувача (рис. 3.10).

```
<h2>Вхід</h2>
<form method="post">
  {% csrf_token %}
  {{ form.as_p }}
  <button type="submit">Увійти</button>
</form>
<p>Немає акаунту? <a href="{% url 'register' %}"><i>Зареєструватись</i></a></p>
```

Рисунок 3.10 – Програмна реалізація форми входу користувача

Головна сторінка містить коротку інформацію про складові вебдодатку, вона реалізована у файлі index.html за допомогою HTML-структури з блоками, що містять посилання на основні функціональні розділи: «Фінанси», «Звички» та «Завдання».

Створення нової звички відбувається через форму HabitForm, де вхідними даними є назва звички та кількість днів запланованого виконання, вихідними – створений об’єкт моделі Habit, який автоматично прив’язується до облікового запису користувача та зберігається у базі даних. Клас ExtendHabitForm використовується для продовження виконання набутої звички, де у якості вхідних даних виступає кількість днів, а вихідних – оновлений запис моделі, в якому змінено мету та скинуто дату завершення. Дані класи форм, що зберігаються в forms.py, відображені на рис. 3.11.

```
class HabitForm(forms.ModelForm):
    target_days = forms.IntegerField(min_value=1, label="Кількість днів")

    class Meta:
        model = Habit
        fields = ['name', 'target_days']

class ExtendHabitForm(forms.Form):
    days = forms.IntegerField(min_value=1, label="Додати днів")
    habit_id = forms.IntegerField(widget=forms.HiddenInput())
```

Рисунок 3.11 – Класи форм HabitForm і ExtendHabitForm

Структура сторінки звичок реалізована у файлі habits.html з використанням механізму шаблонів Django, що дозволяє динамічно виводити інформацію на основі бази даних. Це включає елементи для відображення активних і завершених звичок, кнопки для відміток щоденного виконання, форму для створення нової звички та для її продовження, а також посилання на перегляд історії. При взаємодії користувача з цими елементами викликаються відповідні функції у views.py, які обробляють запити та оновлюють дані. Згадані елементи відображено на рис. 3.12.

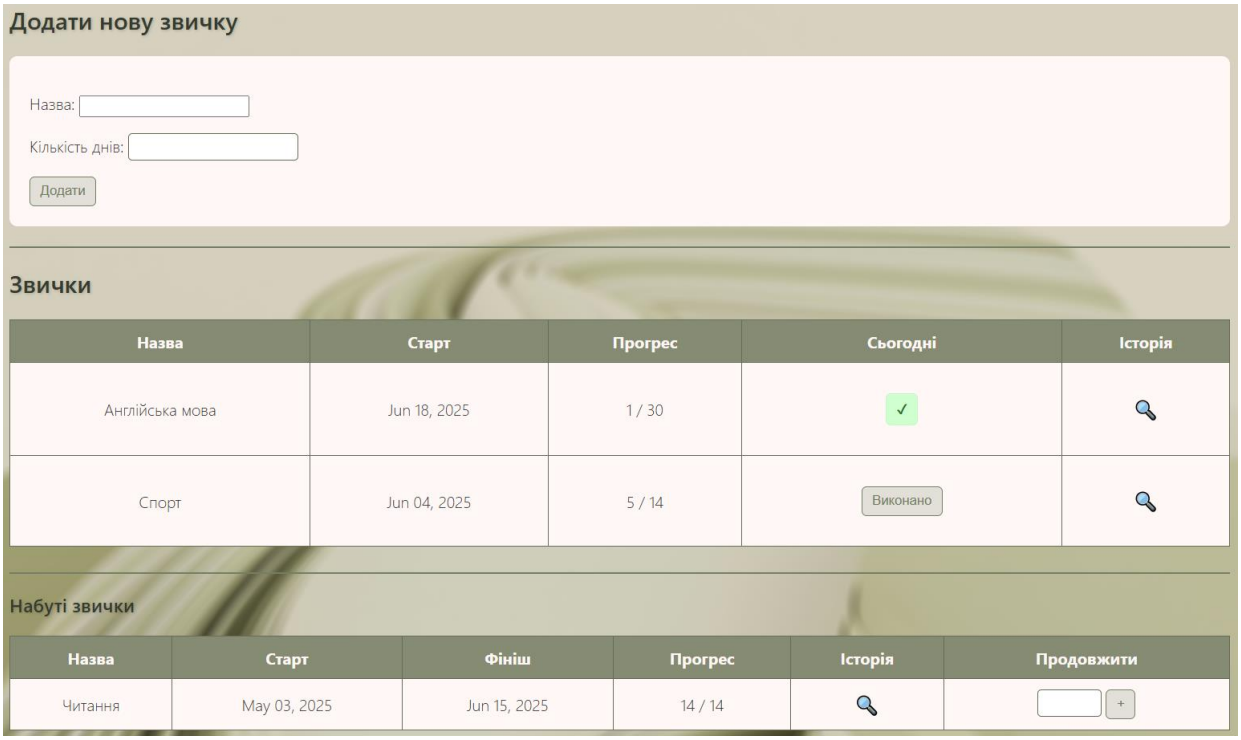


Рисунок 3.12 – Інтерфейс сторінки звичок із формами додавання та продовження й відмітками виконання

Для роботи зі звичками створено кілька функцій, одна з яких – habits_view(request). Її завдання полягає у відображенні всіх активних і завершених звичок користувача, обробці введених даних із форм, а також оновленні прогресу дотримання звички. Функція підтримує створення нової звички, продовження вже завершеної, а також щоденну відмітку про її виконання.

Одним із ключових елементів даної функції є обробка POST-запиту з позначкою про виконання звички поточного дня (рис. 3.13).

```
# перевіряється, чи була надіслана форма з кнопкою "виконано сьогодні"
if 'done_today' in request.POST:
    habit_id = int(request.POST['done_today'])
    habit = get_object_or_404(Habit, id=habit_id, user=request.user)

    # відмітити день як виконаний
    HabitDay.objects.update_or_create(
        habit=habit, date=today, defaults={'done': True}
    )

    # перевірити прогрес
    progress = HabitDay.objects.filter(
        habit=habit,
        date__gte=habit.start_date, # враховуються лише дні після старту
        done=True
    ).count()

    # встановити день завершення звички
    if progress >= habit.target_days and habit.end_date is None:
        latest = HabitDay.objects.filter(habit=habit, done=True).order_by('-date').first()
        if latest:
            habit.end_date = latest.date
            habit.save()
        success_habit_id = habit_id
```

Рисунок 3.13 – Програмна реалізація відмітки дня виконання звички

Функція toggle_habit_day(request) створена для перемикання стану виконання звички на конкретну дату, що передбачає встановлення позначки у разі пропуску дотримання, та її зняття в протилежному випадку. Обробка здійснюється через POST-запит, а результат повертається у форматі JSON для оновлення інтерфейсу без перезавантаження сторінки (рис. 3.14).

```
def toggle_habit_day(request):
    habit_id = request.POST.get('habit_id')
    date_str = request.POST.get('date')
    try:
        habit = Habit.objects.get(id=habit_id, user=request.user)
        day, created = HabitDay.objects.get_or_create(habit=habit, date=date_str)
        day.done = not day.done
        day.save()
        return JsonResponse({'success': True, 'done': day.done})
    except:
        return JsonResponse({'success': False})
```

Рисунок 3.14 – Програмна реалізація функції toggle_habit_day(request)

Функція habits_history_view(request) призначена для побудови календаря дотримання звички, який відображає прогрес користувача. У межах функції для кожного дня місяця формується відповідна інформація про виконання звички, дані готуються у структурі calendar_data, яка містить інформацію про кількість порожніх клітинок на початку тижня, дні місяця та позначки виконання. Цей фрагмент коду представлено на рис. 3.15.

```
done_map = {hd.date: hd.done for hd in habit.habitday_set.all()}
# кількість днів у поточному місяці
days_in_month = calendar.monthrange(current_year, current_month)[1]
days = []
# список усіх днів місяця
for day in range(1, days_in_month + 1):
    d = date(current_year, current_month, day)
    if habit.start_date <= d <= end_or_today:
        days.append({"date": d, "done": done_map.get(d, False)})
    else:
        days.append({"date": d, "done": None})

# кількість порожніх клітинок на початку календаря
empty_cells = (date(current_year, current_month, 1).weekday() + 7) % 7

calendar_data = {
    (current_year, current_month): {
        "days": days,
        "empty_cells": range(empty_cells),
    }
}
```

Рисунок 3.15 – Формування даних для відображення календаря звички

Відображення історії дотримання обраної звички у вигляді календаря з відмітками за місяць реалізовано у шаблоні habits_history.html (рис. 3.16).

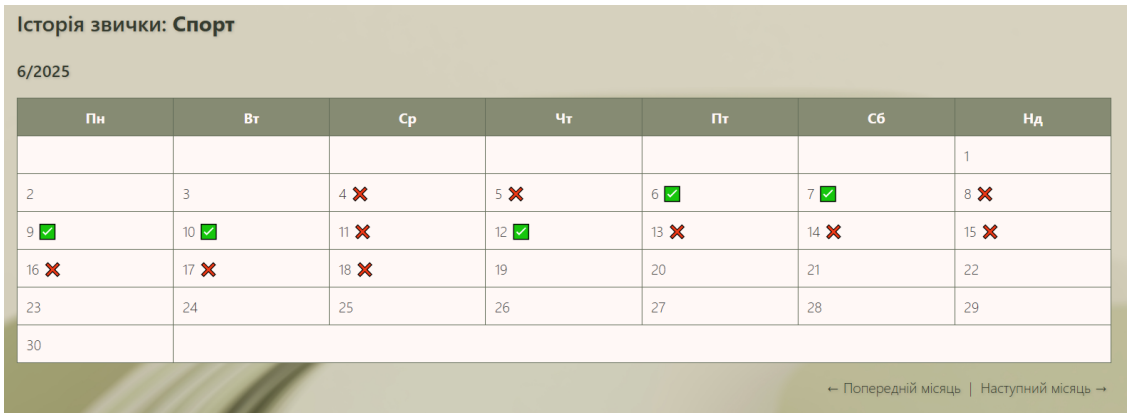


Рисунок 3.16 – Історія звички у вигляді календаря з відмітками

Для реалізації можливості поновлення дотримання набутої звички створено функцію restore_habit_view(request, habit_id), яка скидає дату завершення та додає нову кількість днів до цільового значення (рис. 3.17).

```
def restore_habit_view(request, habit_id):
    habit = get_object_or_404(Habit, id=habit_id, user=request.user)
    habit.target_days += 100
    habit.end_date = None
    habit.save()
    return redirect('habits_view')
```

Рисунок 3.17 – Програмна реалізація функції restore_habit_view(request, habit_id)

Складова бюджету включає можливість додавання нового запису про доходи чи витрати, створення категорії та роботу з накопиченнями. Для роботи з фінансами створено два класи форм: BudgetItemForm, для якого у якості вхідних даних виступають категорія, сума, дата і тип операції, та BudgetCategoryForm, що приймає назву категорії (рис. 3.18). Отримані дані передаються у відповідні моделі та зберігаються в базі даних.

```
class BudgetItemForm(forms.ModelForm):
    class Meta:
        model = BudgetItem
        fields = ['category', 'amount', 'date', 'type']
        widgets = {
            'date': forms.DateInput(attrs={'type': 'date'}),
        }
        labels = {
            'category': 'Категорія',
            'amount': 'Сума',
            'date': 'Дата',
            'type': 'Тип',
        }
    def clean_amount(self):
        amount = self.cleaned_data['amount']
        if amount < 0:
            raise forms.ValidationError("Сума не може бути від'ємною.")
        return amount

class BudgetCategoryForm(forms.ModelForm):
    class Meta:
        model = BudgetCategory
        fields = ['name']
        labels = {
            'name': 'Назва категорії',
        }
```

Рисунок 3.18 – Класи форм BudgetItemForm і BudgetCategoryForm

На сторінці бюджету, яка реалізована у budget.html, розміщено форми для додавання нових записів (рис. 3.19) та таблиці доходів, витрат та накопичень з врахуванням актуального балансу.

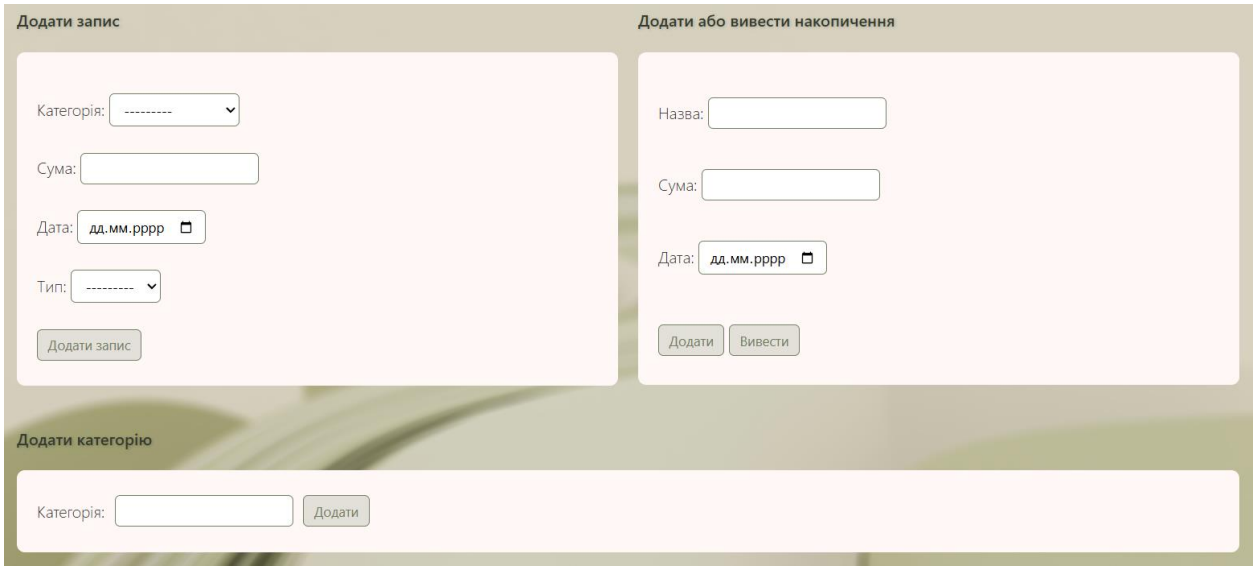


Рисунок 3.19 – Форми додавання записів про доходи чи витрати, накопичення та категорії фінансових операцій

Обчислення актуального балансу за обраний користувачем місяць повинно враховувати стан бюджету за попередній час. У функції budget_view(request) це реалізовано через окреме визначення суми доходів і витрат до початку поточного місяця, розрахунок залишку за цей період та додавання його до залишку за обраний місяць (рис. 3.20).

```
# баланс за попередні місяці (до початку вибраного місяця)
prev_items = BudgetItem.objects.filter(user=request.user, date__lt=start_current_month)
prev_income_sum = prev_items.filter(type='income').aggregate(Sum('amount'))['amount__sum'] or 0
prev_expense_sum = prev_items.filter(type='expense').aggregate(Sum('amount'))['amount__sum'] or 0
prev_balance = prev_income_sum - prev_expense_sum

# баланс за вибраний місяць
month_income_sum = items.filter(type='income').aggregate(Sum('amount'))['amount__sum'] or 0
month_expense_sum = items.filter(type='expense').aggregate(Sum('amount'))['amount__sum'] or 0
month_balance = month_income_sum - month_expense_sum

# загальний баланс з урахуванням попереднього місяця
balance = prev_balance + month_balance
```

Рисунок 3.20 – Розрахунок загального балансу у функції budget_view(request)

Функція budget_view(request) відповідає за обробку основної логіки, пов'язаної з формуванням звітності та додаванням фінансових записів, категорій і накопичень. У її межах реалізовано обробку POST-запитів, які надсилаються з форми, фільтрацію записів за обраний місяць і рік, розрахунок суми накопичень та обчислення актуального балансу.

Створення завдань відбувається через клас форми TaskForm, де вхідні дані – id користувача, назва, опис та кінцевий термін виконання задачі, а вихідні – створений об'єкт моделі Task, який зберігається у базі даних із відповідним користувачем. Даний клас наведено на рис. 3.21.

```
class TaskForm(forms.ModelForm):
    class Meta:
        model = Task
        fields = ['title', 'description', 'deadline', 'completed']
        widgets = {
            'deadline': forms.DateInput(attrs={'type': 'date'}),
        }
        labels = {
            'title': 'Назва',
            'description': 'Опис',
            'deadline': 'Кінцевий термін',
            'completed': 'Виконано',
        }
```

Рисунок 3.21 – Клас форми Task

Функція tasks_view(request) призначена для відображення списку завдань користувача з розподілом на активні та завершені, а також для обробки дій із завданнями. Одним із аспектів її роботи є створення нового завдання, для чого використовується форма TaskForm. Після успішної валідації вона зберігає завдання з автоматичним прив'язуванням до поточного користувача (рис. 3.22).

```
if request.method == 'POST':
    form = TaskForm(request.POST)
    if form.is_valid():
        task = form.save(commit=False)
        task.user = request.user
        task.save()
        return redirect('tasks')
```

Рисунок 3.22 – Програмна реалізація додавання нового завдання

Форма для створення завдань (рис. 3.23), а також їх список за поділом на активні та завершені з можливістю видалення та позначенням виконання відображається завдяки створеному шаблону в tasks.html, який реалізує HTML-структуру та використовує шаблонні теги Django.

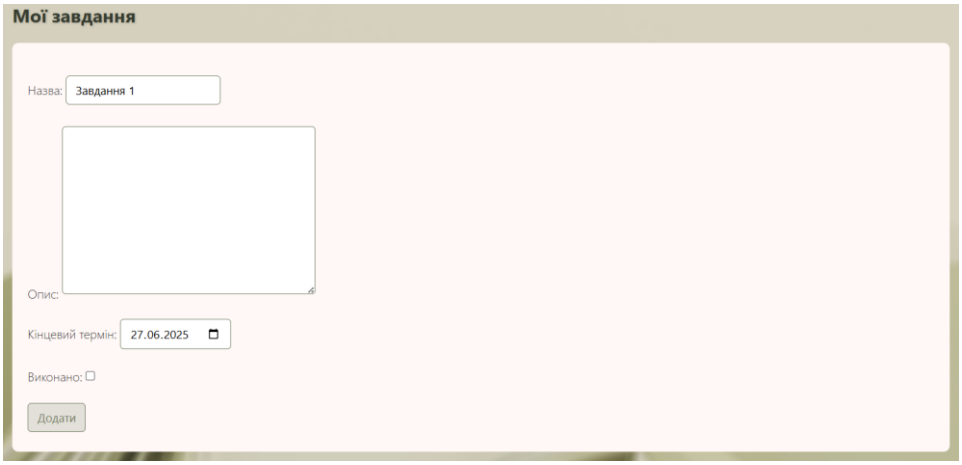


Рисунок 3.23 – Форма створення нового завдання

3.4 Керівництво користувача

Взаємодія користувача з вебдодатком розпочинається з реєстрації, для якої необхідно вказати логін та пароль, що відповідає вимогам (рис. 3.24).

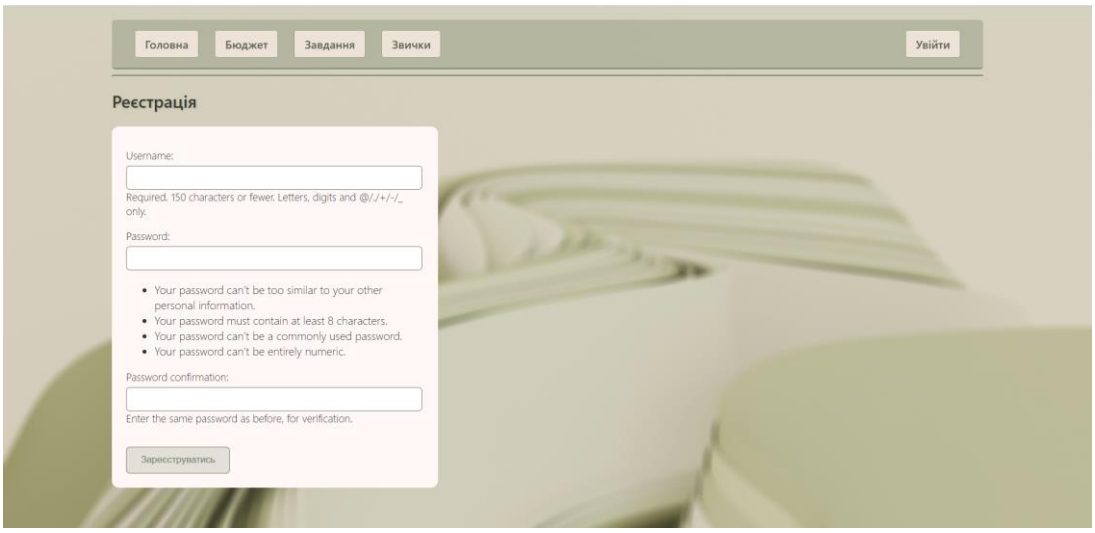


Рисунок 3.24 – Сторінка реєстрації облікового запису користувача

За наявності облікового запису користувач авторизується шляхом введення логіну та паролю (рис. 3.25).

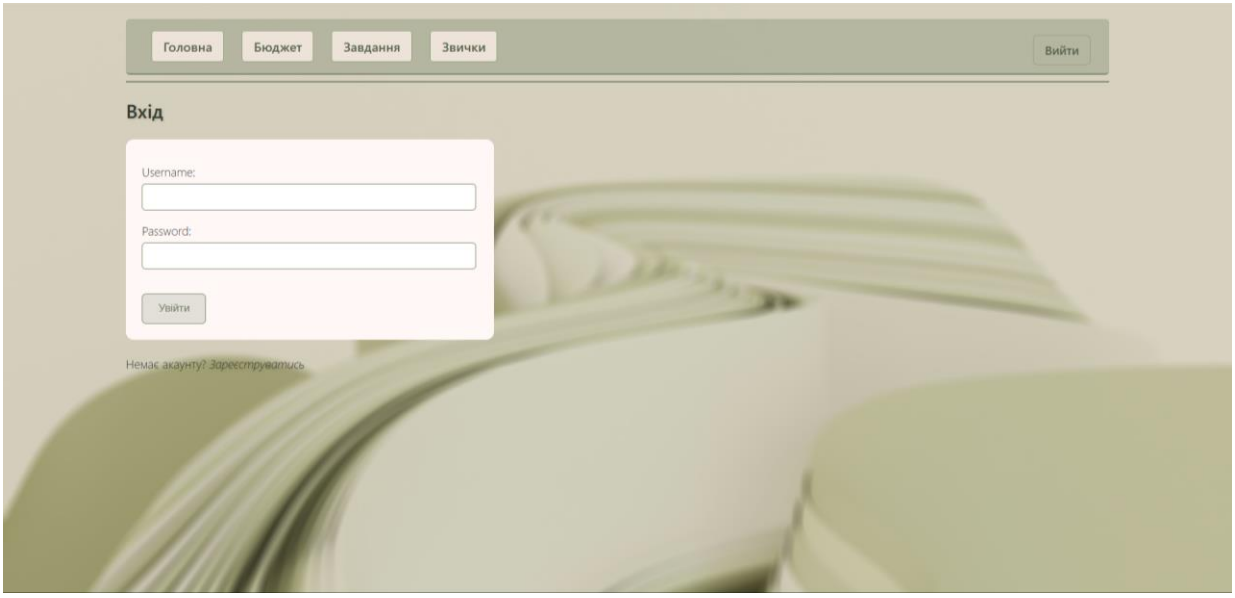


Рисунок 3.25 – Сторінка входу користувача в систему

Потрапивши на головну сторінку, користувач має можливість ознайомитися з короткою інформацією про можливості трьох складових вебдодатку: фінансів, звичок та завдань (рис. 3.26).

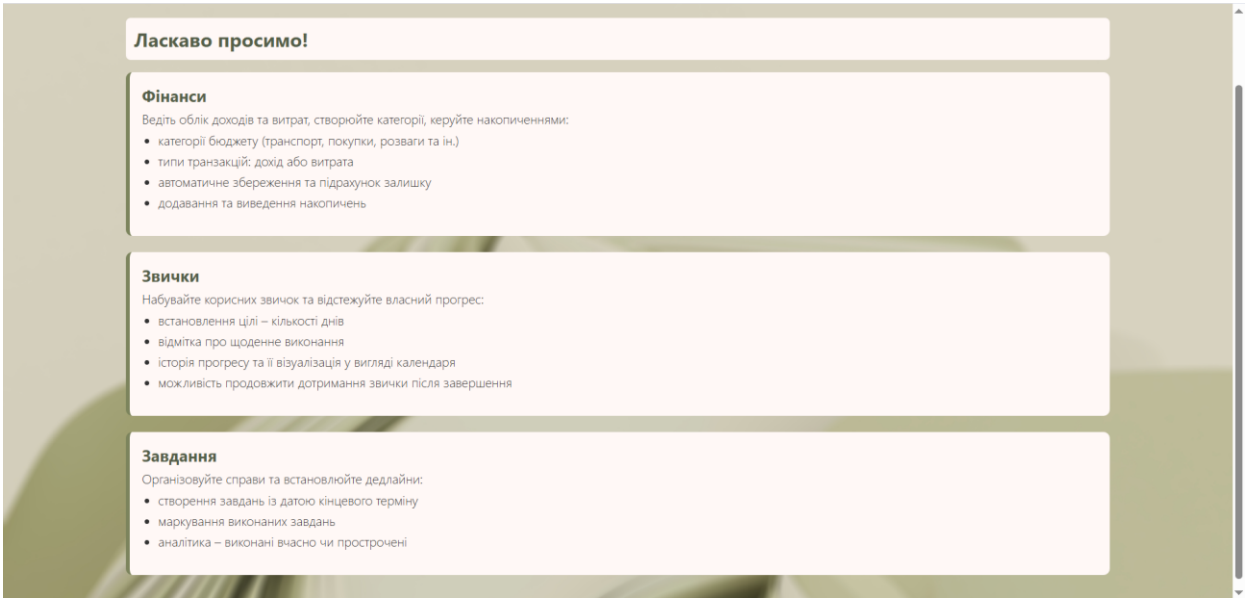


Рисунок 3.26 – Головна сторінка вебдодатку

Натиснувши на посилання в описі або на відповідну кнопку в навігаційному меню, користувач може перейти до керування бюджетом, що включає вибір періоду для перегляду записів, створення категорій фінансових операцій або заповнення форм для додавання доходів, витрат чи змін у накопиченнях (рис. 3.27).

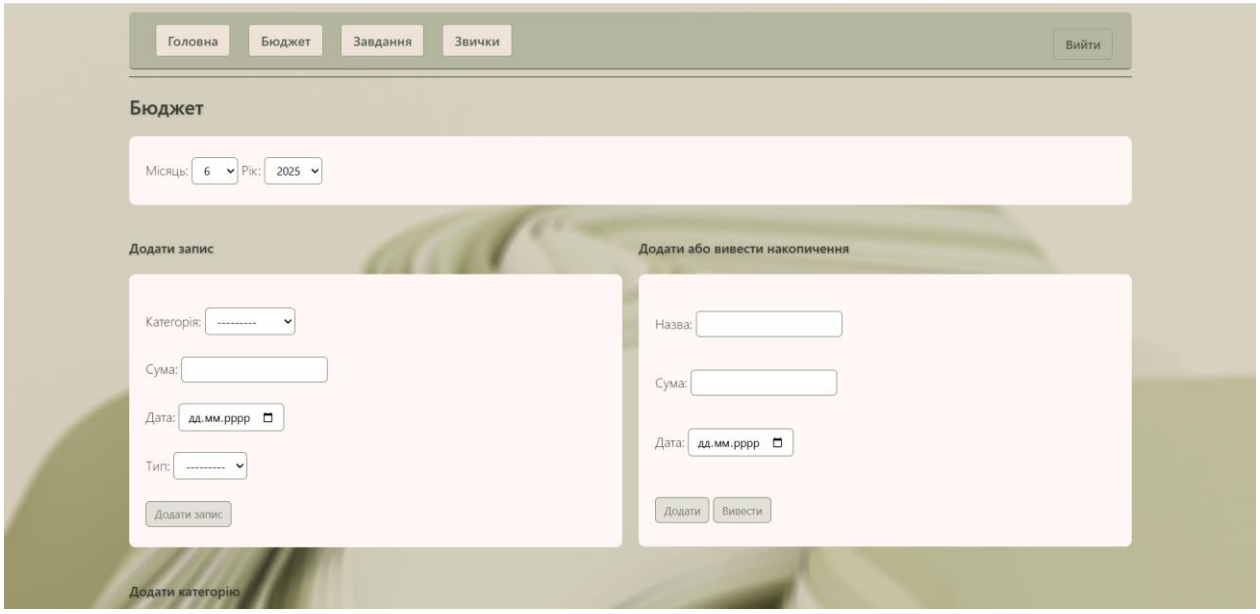


Рисунок 3.27 – Сторінка додавання записів про бюджет

У разі некоректного введення даних система повідомить про помилку (рис. 3.28).

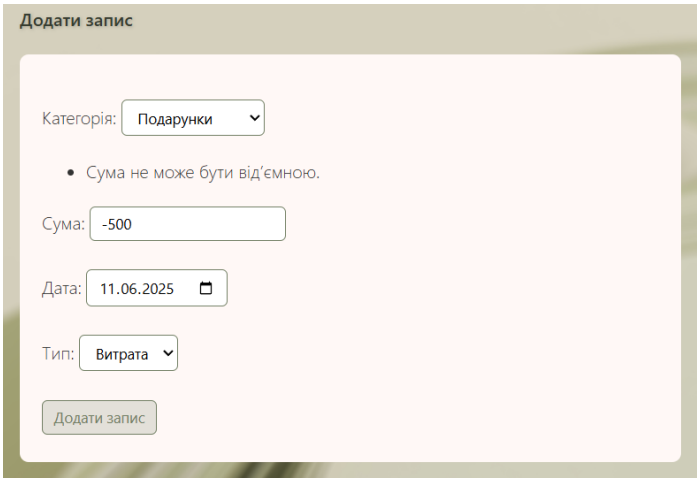
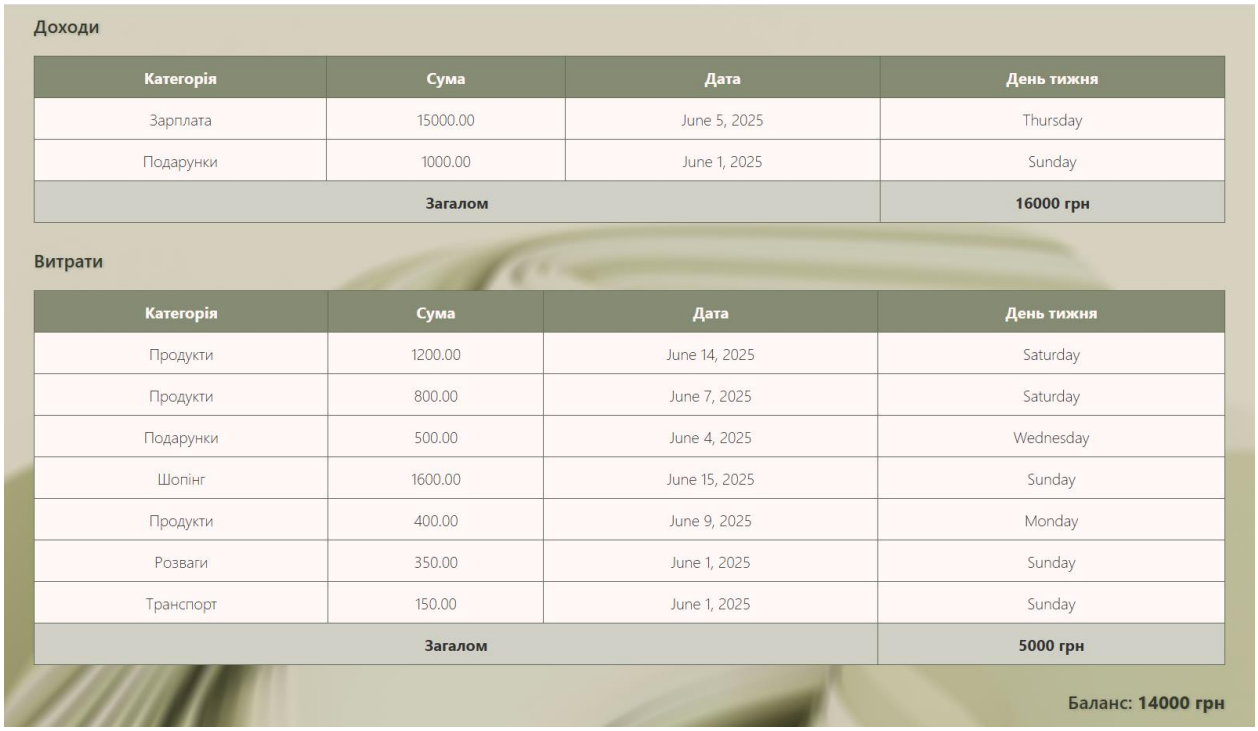


Рисунок 3.28 – Помилка від’ємної суми запису бюджету

Наявні фінансові записи можна переглянути нижче. Заповнені дані для доходів та витрат з розрахованим балансом, що враховує зміни в бюджеті за попередній період, наведено на рис. 3.29.



Доходи

Категорія	Сума	Дата	День тижня
Зарплата	15000.00	June 5, 2025	Thursday
Подарунки	1000.00	June 1, 2025	Sunday
Загалом			**16000 грн**

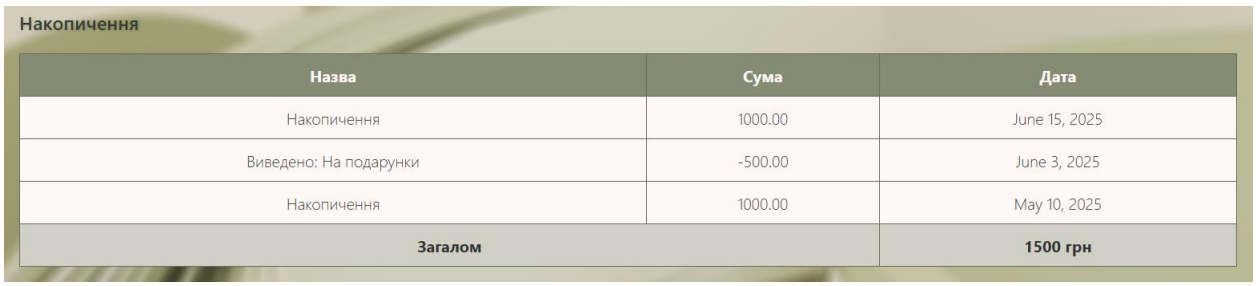
Витрати

Категорія	Сума	Дата	День тижня
Продукти	1200.00	June 14, 2025	Saturday
Продукти	800.00	June 7, 2025	Saturday
Подарунки	500.00	June 4, 2025	Wednesday
Шопінг	1600.00	June 15, 2025	Sunday
Продукти	400.00	June 9, 2025	Monday
Розваги	350.00	June 1, 2025	Sunday
Транспорт	150.00	June 1, 2025	Sunday
Загалом			**5000 грн**

Баланс: **14000 грн**

Рисунок 3.29 – Таблиці доходів та витрат з актуальним балансом

Записи про накопичення відображаються включно до обраного місяця з відображенням додавання та виведення заощаджень (рис. 3.30).



Накопичення

Назва	Сума	Дата
Накопичення	1000.00	June 15, 2025
Виведено: На подарунки	-500.00	June 3, 2025
Накопичення	1000.00	May 10, 2025
Загалом		**1500 грн**

Рисунок 3.30 – Таблиця з накопиченнями

На сторінці завдань наявна можливість створення задачі з назвою, описом, кінцевим терміном та станом виконання (рис. 3.31).

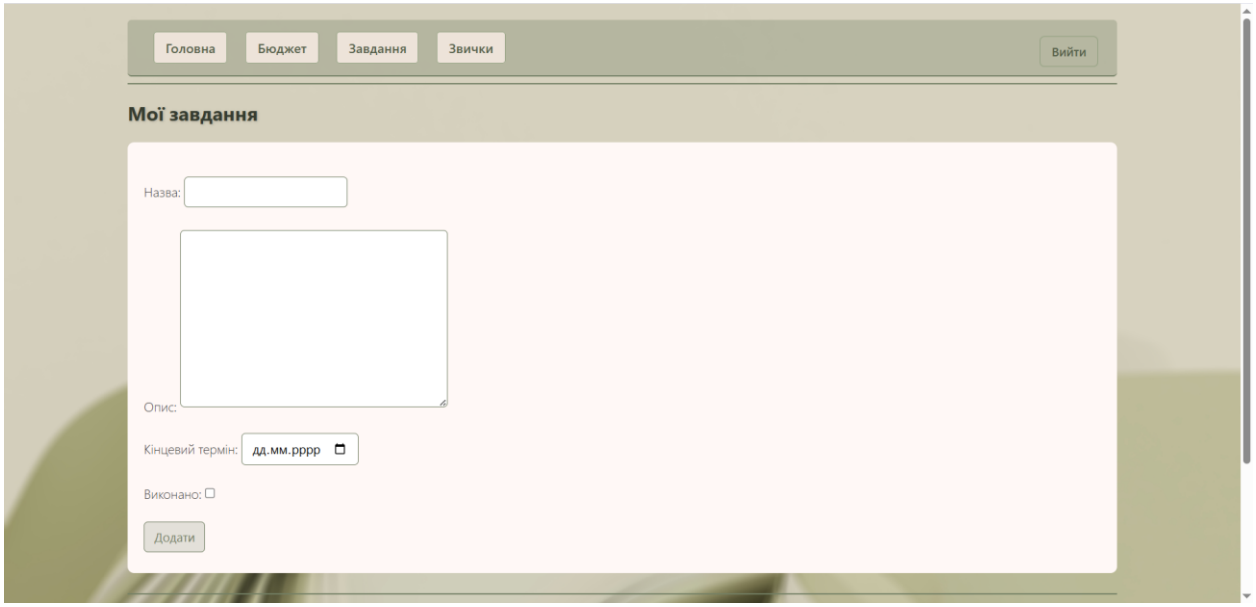


Рисунок 3.31 – Сторінка створення нового завдання

Створені завдання можуть переміститися до однієї з двох категорій:

– активні, які можна видалити або позначити як виконані;

– виконані – завершені задачі.

У разі прострочення термін виконання завдання набуде червоного забарвлення. Приклад списків завдань наведено на рис. 3.32.

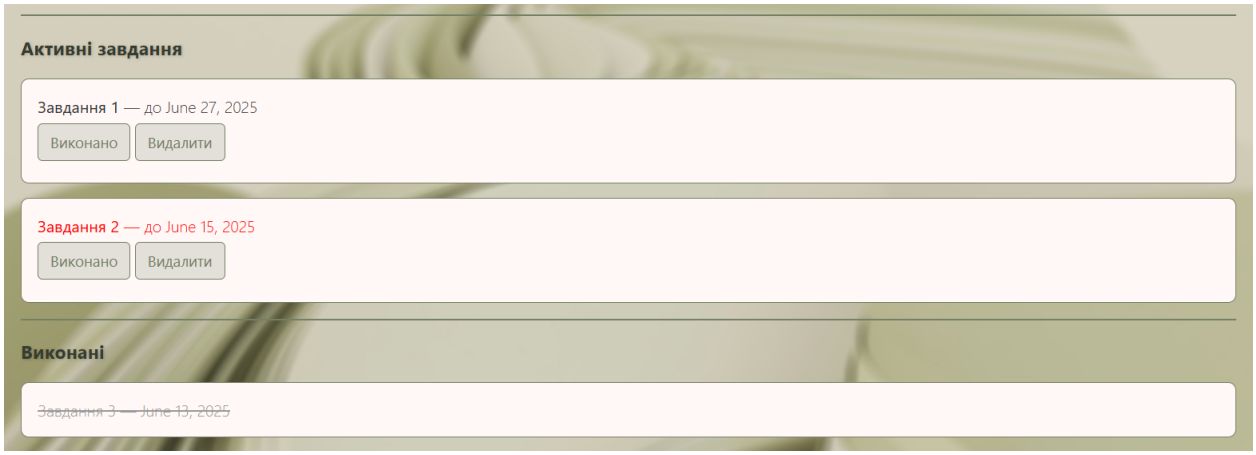


Рисунок 3.32 – Активні та виконані завдання

Також користувач має можливість створювати власні звички та індивідуально визначати мету, яка являє собою бажану кількість днів дотримання (рис. 3.33).

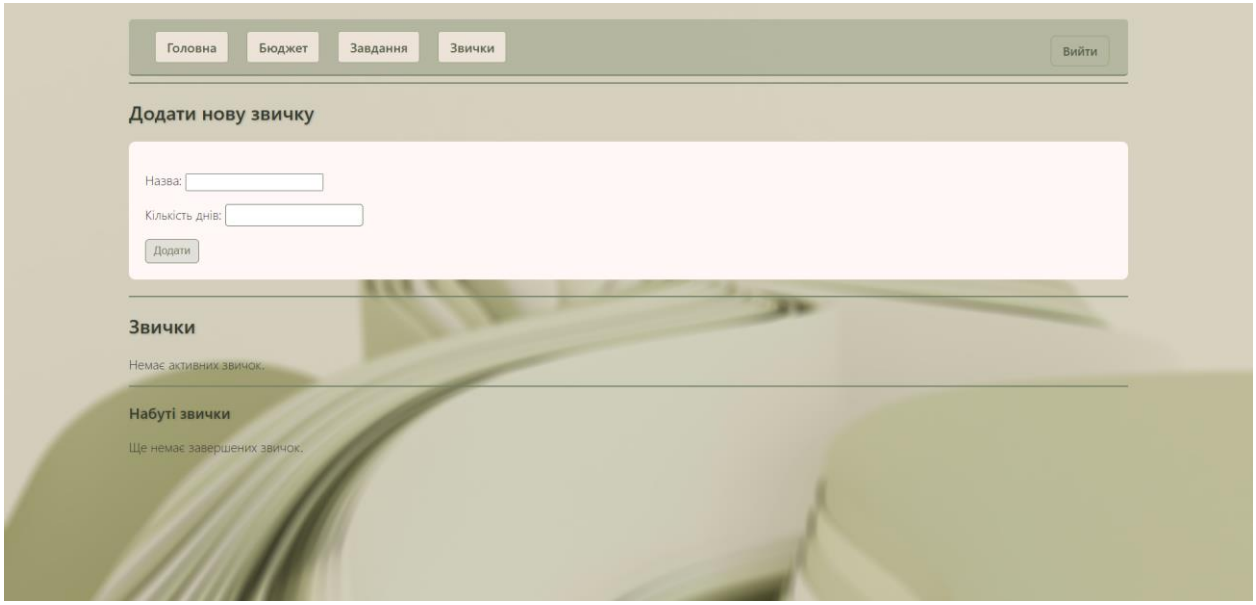


Рисунок 3.33 – Сторінка створення нової звички

Наявні звички відображаються у вигляді таблиці, де можна побачити власний прогрес, відмітити поточний день виконання та переглянути історію. Набутими вважаються ті звички, мета яких була досягнута, вони наводяться окремо з можливістю продовження на бажану кількість днів (рис. 3.34).

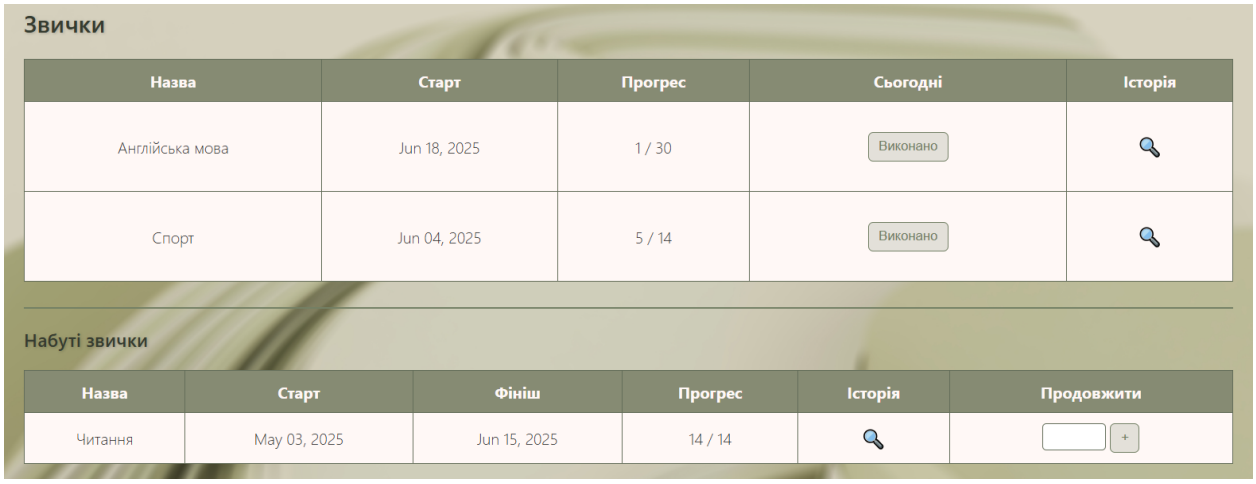


Рисунок 3.34 – Активні та набуті звички

Історія дотримання звичок відображається у вигляді календаря з позначками успішності виконання поставленої користувачем мети (рис. 3.35).

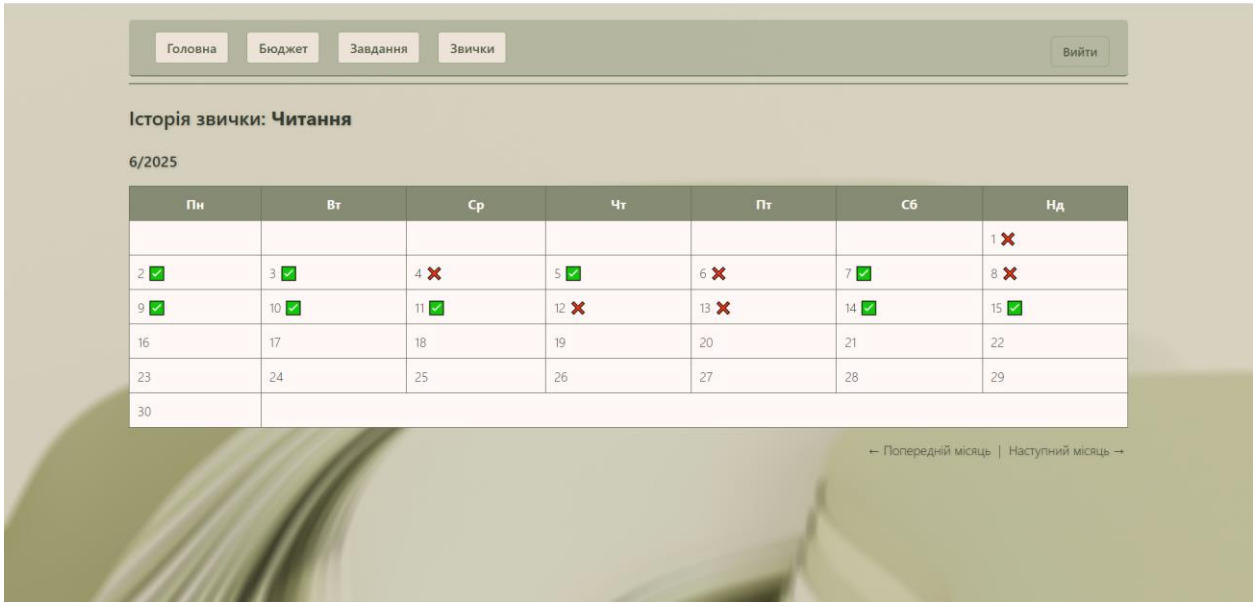


Рисунок 3.35 – Історія дотримання звички

Висновки до розділу

У межах розділу було розглянуто засоби розробки, які включають мову програмування Python, фреймворк для веброзробки Django та базу даних SQLite. Вибір таких інструментів пов’язаний з їх доступністю та широкими можливостями для реалізації вебдодатку. Для створення шаблонів інтерфейсу користувача та оформлення зовнішнього вигляду було використано HTML та CSS. Середовищем розробки став Visual Studio Code, а для взаємодії з базою даних було обрано DB Browser. В сукупності дані засоби сприяють створенню функціонального вебдодатку.

Було сформульовано вимоги до технічного та програмного забезпечення та розглянуто процес розробки вебдодатку включно зі створенням бази даних та реалізацією логіки. Ключовими аспектами програмної реалізації стали структура моделей, взаємодія з формами та шаблонами, створення функцій обробки запитів та гарантування безпеки при обробці даних користувача.

Результатом розробки є вебдодаток для самоорганізації, що об’єднує інструменти контролю бюджету, відстеження звичок та виконання завдань, процес взаємодії з яким описано в керівництві користувача.

РОЗДІЛ 4
ОХОРОНА ПРАЦІ

4.1 Регулювання питань охорони праці на законодавчому рівні

Охорона праці є складовою трудового процесу, що охоплює організаційні, технічні та соціальні заходи, мета яких – гарантування безпеки, збереження здоров'я та запобігання травматизму працівників під час виконання ними професійних обов'язків.

Згідно з Законом України «Про охорону праці», дане поняття розкривається як система правових, соціально-економічних, організаційно-технічних, санітарно-гігієнічних і лікувально-профілактичних заходів та засобів, спрямованих на збереження життя, здоров'я і працездатності людини у процесі трудової діяльності [14].

Таким чином, охорона праці забезпечує не лише фізичну безпеку працівника, а й створює умови для ефективної праці, профілактики професійних захворювань і виробничого травматизму, а завдяки законодавству в Україні встановлено єдиний порядок її організації на підприємствах. Крім того, Закон України «Про охорону праці» визначає положення, відповідно до яких втілюється конституційне право працівників на належні умови праці та охорону їх життя і здоров'я під час трудової діяльності. Його дія поширюється на всіх працівників та юридичних чи фізичних осіб, які використовують найману працю.

Важливим для забезпечення охорони праці на робочих місцях підприємств та організацій є й Кодекс законів про працю України, який регулює трудові відносини всіх працівників, сприяючи не лише зростанню продуктивності роботи та поліпшенню її якості, а й підвищенню ефективності суспільного виробництва, матеріального і культурного рівня життя працівників і зміцненню трудової дисципліни [15]. Даний кодекс встановлює

високий рівень умов праці та охороняє трудові права працівників, завдяки чому створюються передумови для стабільного функціонування трудових колективів, зменшення виробничих ризиків і формування здорового мікроклімату на підприємстві.

Трудовий договір є угодою між роботодавцем та найманим працівником, за умовами якого не лише виконуються завдання, а й гарантується безпека – це означає, що роботодавець зобов'язаний створити належні умови праці, виплачувати заробітну плату, зазначену в договорі, і дотримуватись КЗпП та інших законів, які регулюють питання праці [16, с. 392].

Однак у сфері інформаційних технологій досить поширеним є цивільно-правовий договір, який укладається між двома чи декількома особами задля здійснення дій, що забезпечують досягнення цілі учасників угоди і задоволення їх інтересів [17, с. 92]. Це передбачає виконання узгоджених завдань з орієнтуванням на кінцевий результат. Однак, важливо враховувати, що, на відміну від трудового договору, в цивільно-правовому не передбачено гарантій, пов'язаних із охороною праці, таких як нормований робочий час чи безпечні умови праці.

Окремою категорією цивільно-правового договору є гіг-контракт, який регулюється Законом України «Про стимулювання розвитку цифрової економіки в Україні» [18].

Вагому роль у реалізації законодавчих положень з охорони праці у діяльності організацій відіграють роботодавці, оскільки їхнім обов'язком є створення, подання на розгляд або ж схвалення окремих локальних нормативно-правових актів [19, с. 96] у даній сфері. Такі документи включають важливі аспекти влаштування заходів безпеки, зокрема і складання інструкцій з охорони праці та систему оплати за роботу у важких умовах. Можливості локального регулювання охорони праці для роботодавця є обмеженими, тому це гарантує належне дотримання вимог законодавства у питаннях безпеки людей під час виконання професійних обов'язків.

Також важливо зазначити і роль трудового колективу, адже він володіє правом ініціювання позитивних змін в охороні праці та повноваженнями щодо контролю за таким утіленням [20, с. 43], сприяючи мінімізації ризиків у процесі трудової діяльності та створенню належних для цього умов.

4.2 Виявлення потенційних небезпек стосовно об’єкту проєктування

Трудова діяльність працівників ІТ-сфери, зокрема і розробників, може виконуватися віддалено, однак зазвичай вона зосереджена в офісних приміщеннях. Така робота супроводжується низкою потенційних небезпечних і шкідливих виробничих факторів різного походження, до яких належать і фізичні фактори, які наведено у таблиці 4.1.

Таблиця 4.1 – Фізичні небезпеки для працівників ІТ-сфери

№ з/п	Небезпека	Джерело виникнення	Можливі наслідки
1	Підвищений рівень шуму	Вентилятори, кондиціонери, сусідні офіси	Погіршення слуху, зниження концентрації
2	Недостатнє або надмірне освітлення	Неправильне розташування ламп	Втома очей, головний біль, зниження продуктивності
3	Підвищений рівень електромагнітного випромінювання	Комп’ютери, Wi-Fi-роутери, інша техніка	Можливий хронічний вплив на організм у разі тривалої дії
4	Підвищена або знижена температура повітря	Недостатня вентиляція, несправне опалення або кондиціонування	Дискомфорт, порушення терморегуляції, зниження працездатності
5	Гострі краї меблів	Робочі столи, шафи	Травмування при зіткненні
6	Підвищена або знижена вологість повітря	Несправна вентиляція, відсутність зволожувачів або осушувачів	Сухість слизових, подразнення очей або дихальних шляхів, загострення алергій
7	Підвищена або знижена рухливість повітря	Протяги, застійне повітря через погану циркуляцію	Простудні захворювання, головний біль, зниження концентрації
8	Підвищена або знижена іонізація повітря	Робота великої кількості електронної техніки, відсутність вентиляції	Втома, головний біль, зниження концентрації, порушення самопочуття

На працівників ІТ-сфери впливають й хімічні небезпечні та шкідливі виробничі фактори, до яких належать токсичні, дратівливі, сенсибілізуючі, канцерогенні чи мутагенні речовини, а також ті, що чинять негативний вплив на репродуктивну функцію. Потрапити в організм людини вони можуть через органи дихання, шлунково-кишковий тракт або шкірні покриви і слизові оболонки, внаслідок чого виникають гострі та хронічні захворювання, порушення імунної системи чи алергічні реакції. Джерелами таких речовин є матеріали, застосовані в ремонті, засоби для дезінфекції, а також продукти зношування техніки, що використовується в офісному середовищі.

Біологічні небезпеки розглянуто в таблиці 4.2.

Таблиця 4.2 – Біологічні небезпеки для працівників ІТ-сфери

№ з/п	Небезпека	Джерело виникнення	Можливі наслідки
1	Патогенні мікроорганізми	Контакт із зараженими поверхнями, повітря	Інфекційні захворювання
2	Продукти життєдіяльності мікроорганізмів	Пліснява, бактерії	Алергії, інтоксикації
3	Макроорганізми (пилові кліщі, пліснява)	Пил, вологі кути приміщень	Алергічні реакції, проблеми з диханням

Психофізіологічні небезпечні й шкідливі виробничі фактори пов’язані безпосередньо з особливостями та умовами виконання роботи ІТ-фахівців, вони наведені в таблиці 4.3.

Таблиця 4.3 – Психофізіологічні небезпеки для працівників ІТ-сфери

№ з/п	Небезпека	Джерело виникнення	Можливі наслідки
1	Малорухомість	Тривала робота за комп’ютером	Захворювання опорно-рухового апарату, варикоз
2	Психоемоційне перенавантаження	Висока інтенсивність праці, дедлайни	Стрес, вигорання, зниження працездатності
3	Втома очей	Тривала робота за комп’ютером	Погіршення зору, головний біль
4	Монотонність роботи	Повторювані однотипні завдання	Втома, стрес, вигорання, зниження продуктивності

Задля нейтралізації визначених небезпечних та шкідливих виробничих факторів необхідно впроваджувати регулярне прибирання та провітрювання офісних приміщень, застосовувати засоби захисту від алергенів і патогенів, а також організовувати робочий процес із урахуванням перерв для фізичної активності та відпочинку очей. Потрібно проводити моніторинг психологічного стану ІТ-фахівців та запроваджувати гнучкі графіки роботи.

Загроза життю та здоров'ю працівників суттєво зростає через повітряні тривоги, тому в умовах воєнного стану важливо організовувати надійні укриття та дотримуватися правил евакуації. Крім того, задля створення безпечних умов праці необхідно враховувати можливість виникнення пожежі, джерелом якої можуть бути несправні електроприлади. Важливо проводити технічний контроль та впроваджувати належні заходи пожежної безпеки.

4.3 Дослідження ризику реалізації потенційних небезпек на об'єкті проєктування та розробка заходів щодо їх попередження

Оцінка ризику – це процедура, за допомогою якої визначаються потенційно небезпечні фактори, що можуть негативно впливати на здоров'я і життя працівників. Даний процес вимагає глибокого аналізу умов праці, технічного стану обладнання, організації виробництва, а також особливостей людського фактора.

Метою оцінки ризику є зниження ймовірності виникнення небезпек або невизначеностей, що чинять негативний вплив на цілі підприємства. Це можуть бути нещасні випадки, аварійні ситуації, фізичні чи психологічні перенавантаження та професійні захворювання. Аналіз ризиків дозволяє ідентифікувати найбільш вразливі об'єкти та процеси на підприємстві і визначити необхідні заходи для їх захисту [21, с. 540].

Основні етапи процедури оцінки ризику включають:

– виявлення потенційних небезпек;

– аналіз ймовірності виникнення ризиків;
– оцінка впливу визначених ризиків;
– розробка заходів контролю для усунення небезпек чи їх пом’якшення.

Оцінку ризиків на підприємстві можна здійснити силами діючої служби охорони праці або відділу, відповідального за розвиток виробництва [22, с. 65]. Максимальний результат досягається завдяки їх об’єднаній співпраці.

Одним із методів оцінки ризиків є аналізування «дерева відмов», що полягає у визначенні усіх можливих способів виникнення небажаної події, тобто небезпеки.

Для працівників ІТ-сфери існує велика ймовірність появи професійного вигорання внаслідок психоемоційного перенавантаження, оскільки під час роботи вони можуть зіштовхнутися з багатьма факторами, які розглянуто на рисунку 4.1.

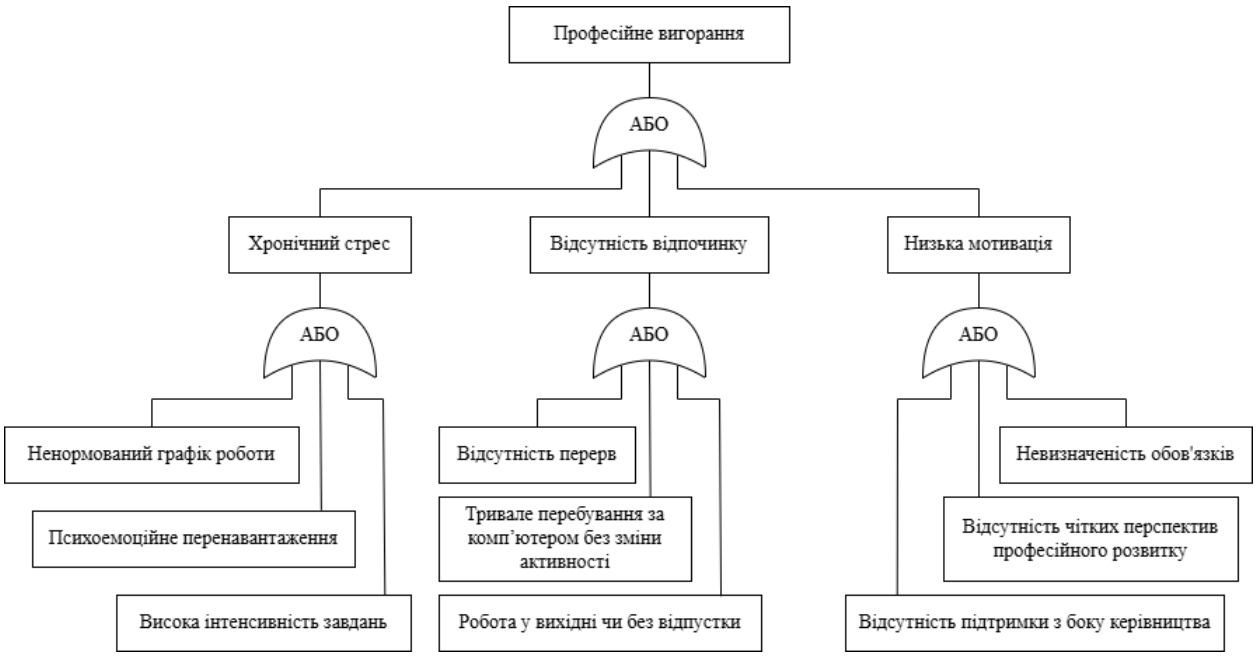


Рисунок 4.1 – Дерево відмов професійного вигорання ІТ-фахівця

Щоб запобігти виникненню вигорання, важливо впровадити регулярні перерви під час виконання професійних обов’язків, проводити тренінги з управління стресом, стимулювати психологічну підтримку та створити комфортне робоче середовище. Крім того, графік з чітким розмежуванням

особистого та робочого часу сприяє збереженню балансу та відновленню ресурсів працівника. Ці заходи допоможуть знизити вплив таких факторів, як хронічний стрес, відсутність якісного відпочинку та низька мотивація, на появу професійного вигорання.

Враховуючи, що трудова діяльність ІТ-фахівців зосереджена в офісному приміщенні, існує ризик виникнення пожежі, яка може бути спричинена коротким замиканням електромережі, перегрівом електронного обладнання, використанням несправних подовжувачів або порушенням правил протипожежної безпеки (рис. 4.2).

Рисунок 4.2 – Дерево відмов виникнення пожежі в офісі ІТ-компанії

Задля попередження пожежі необхідно дотримуватись правил електробезпеки, використовувати сертифіковане обладнання, проводити регулярну перевірку стану електромереж, встановити засоби пожежогасіння, забезпечити евакуаційні плани та навчити працівників діям у разі надзвичайної ситуації.

Причиною погіршення зору чи проблем з опорно-руховим апаратом для працівників ІТ-сфери може стати тривала робота за комп’ютером без перерв на відпочинок. Додатковий вплив на фізичне здоров’я можуть чинити й інші

фактори, такі як використання невідповідного освітлення, неправильне положення тіла за робочим місцем або невідрегульоване крісло чи монітор (рис. 4.3).

Рисунок 4.3 – Дерево відмов розвитку професійних захворювань через сидячу роботу за ПК

Допомогти у протидії ризикам розвитку професійних захворювань могло б правильне облаштування робочого місця, що включає використання ергономічного крісла, розміщення монітора на рівні очей на відстані 50-70 см, якісне освітлення та регулювання висоти столу відповідно до зросту працівника. Додатково важливими є регулярні перерви, які дають можливість відпочити очам та виконати фізичні вправи.

Висновки до розділу

Головною метою розділу є визначення законодавчого регулювання охорони праці в Україні, аналіз потенційних небезпек для ІТ-фахівців та

формулювання заходів для усунення або зменшення їх впливу на життя та здоров’я працівників у процесі трудової діяльності.

Було встановлено, що Кодекс законів про працю та Закон України «Про охорону праці» є основними законодавчими актами в питаннях охорони праці, але є й інші нормативно-правові документи, які доповнюють їх положення та регламентують окремі аспекти безпечної організації праці.

У ході роботи було ідентифіковано найбільш поширені ризики для ІТ-фахівців, які включають низку потенційних небезпечних і шкідливих виробничих факторів різного походження, зокрема фізичні, хімічні, біологічні та психофізіологічні.

Задля зменшення ймовірності виникнення професійних захворювань та нещасних випадків запропоновано комплекс заходів, що включає ергономічну організацію робочого місця, регулярні перерви під час виконання обов’язків, дотримання правил електробезпеки, психологічну підтримку персоналу, санітарно-гігієнічні заходи та організацію безпечного середовища під час повітряних тривог.

ВИСНОВКИ

У ході кваліфікаційної роботи було проведено аналіз предметної області та наявних програмних продуктів у сфері організації особистої продуктивності, що дозволило виявити напрямок для розробки комплексного рішення, яке поєднує контроль бюджету й завдань та відстеження звичок.

Проєктування архітектури та функціоналу інформаційної системи було проведено із застосуванням BPMN-схем, що відображають ключові процеси взаємодії користувача з вебдодатком. Також було розглянуто математичне забезпечення для реалізації складової контролю бюджету, проєктування бази даних та об’єктно-орієнтованої моделі.

Інструментами розробки стали Python, Django, SQLite, HTML та CSS, а середовищем – Visual Studio Code, взаємодія з базою даних відбувалася у DB Browser. У ході роботи було обґрунтовано вибір засобів розробки та розглянуто вимоги до технічного та програмного забезпечення. Реалізація функціональності вебдодатку включно зі створенням облікового запису користувача та інструментами для керування фінансами, звичками й завданнями налічувала два етапи: створення бази даних та реалізація логіки. Було складено керівництво користувача з відображенням усіх наявних процесів взаємодії з системою, що підтвердило дієздатність створеного рішення та зручність його використання.

Результатом проведеної роботи став вебдодаток для комплексного управління особистою ефективністю, що не лише надає звучний інтерфейс користувача, а й створює безпечні умови для фінансового обліку, організації завдань, відстеження успішності впровадження звичок та наочного демонстрування продуктивності у вигляді календаря.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Origin. The State of Personal Finance Apps 2024 [Електронний ресурс] // Режим доступу: https://www.useorigin.com/resources/blog/origins-state-of-personal-finance-apps-2024

2. Global Growth Insights. Habit Tracking App Market – Global Industry Analysis and Forecast to 2032 [Електронний ресурс] // Режим доступу: https://www.globalgrowthinsights.com/market-reports/habit-tracking-app-market-100455

3. Новікова Д. Г., Братерська Н. М. ОСНОВНІ ЗАГРОЗИ БЕЗПЕЦІ ТА МЕТОДИ ЗАХИСТУ ДАНИХ У ВЕБ-ДОДАТКАХ ДЛЯ РОБОТИ З ФІНАНСАМИ. Матеріали XVIII Всеукраїнської науково-технічної конференції «Сталий розвиток міст: поствоєнний період» (90-ї науково-технічної конференції ХНУМГ ім. О. М. Бекетова) : в 5-и ч. / Ч. 2. – Харків : ХНУМГ ім. О. М. Бекетова, 2025. – С. 180-182 [Електронний ресурс] // Режим доступу: https://science.kname.edu.ua/images/dok/konferentsii/stalyirozvytok2019/2025/C.%202_Energeticna%20informacijna%20ta%20transportna%20infrastruktura_25.pdf

4. БАГРІЙ Р., БАРМАК О., МАНЗЮК Е. ПІДВИЩЕННЯ СТІЙКОСТІ ПАРОЛІВ У ВЕБ-СИСТЕМАХ ЗА ДОПОМОГОЮ ВДОСКОНАЛЕНИХ СХЕМ ХЕШУВАННЯ. Herald of Khmelnytskyi National University. Technical sciences. 2024. Т. 331, № 1. С. 48–51 [Електронний ресурс] // Режим доступу: https://doi.org/10.31891/2307-5732-2024-331-6

5. Головна сторінка вебдодатку Notion [Електронний ресурс] // Режим доступу: https://www.notion.com/

6. Головна сторінка вебдодатку TickTick [Електронний ресурс] // Режим доступу: https://ticktick.com/

7. SQLite [Електронний ресурс] // Режим доступу: https://www.sqlite.org/

8. Мамалига Н. Є., Катаєва А. І. Система керування базами даних в сучасних умовах IT-індустрії. Вісник студентського наукового товариства Донецького національного університету імені Василя Стуса. Том 1 / Ред. кол. Хаджинов І. В. (голова) та ін. Вінниця : ДонНУ імені Василя Стуса, 2021. Вип. 13. Т. 1. С. 275–279 [Електронний ресурс] // Режим доступу: https://jvestnik-sss.donnu.edu.ua/article/view/10004

9. Python Software Foundation. Python Programming Language [Електронний ресурс] // Режим доступу: https://www.python.org/

10. Django Software Foundation. Django Web Framework [Електронний ресурс] // Режим доступу: https://www.djangoproject.com/

11. Visual Studio Code [Електронний ресурс] // Режим доступу: https://code.visualstudio.com/

12. DB Browser for SQLite [Електронний ресурс] // Режим доступу: https://sqlitebrowser.org/

13. Парфьонов Ю. Е. Питання міграції схеми бази даних під час супроводу веб-застосунків на базі фреймворку Django. Системи обробки інформації: збірник наукових праць. - Харків: Харківський університет Повітряних Сил імені Івана Кожедуба, 2024. – Вип. 2 (177). С. 63–67 [Електронний ресурс] // Режим доступу: https://doi.org/10.30748/soi.2024.177.07

14. Про охорону праці : Закон України від 14.10.1992 № 2694-XII : станом на 4 квіт. 2025 р. [Електронний ресурс] // Режим доступу: https://zakon.rada.gov.ua/laws/show/2694-12#Text

15. Кодекс законів про працю України : Кодекс України від 10.12.1971 № 322-VIII : станом на 2 трав. 2025 р. [Електронний ресурс] // Режим доступу: https://zakon.rada.gov.ua/laws/show/322-08#Text

16. Вознюк В. С. Правове регулювання праці IT-працівників. Аналітично-порівняльне правознавство. 2024. № 6. С. 390–396. [Електронний ресурс] // Режим доступу: https://doi.org/10.24144/2788-6018.2024.06.63

17. Мороз О. В. ПОНЯТТЯ ЦИВІЛЬНО-ПРАВОВОГО ДОГОВОРУ. Право і суспільство. 2023. № 1. С. 90–95. [Електронний ресурс] // Режим доступу: https://doi.org/10.32842/2078-3736/2023.1.13

18. Про стимулювання розвитку цифрової економіки в Україні : Закон України від 15.07.2021 № 1667-IX : станом на 1 січ. 2025 р. [Електронний ресурс] // Режим доступу: https://zakon.rada.gov.ua/laws/show/1667-20#Text

19. Вареник О. С. РОБОТОДАВЕЦЬ, ЯК СУБ'ЄКТ ЛОКАЛЬНОГО РЕГУЛЮВАННЯ ОХОРОНИ ПРАЦІ. Юридична наука. 2019. № 12(102). С. 92–99. [Електронний ресурс] // Режим доступу: https://journal-nam.com.ua/index.php/journal/issue/download/19/29#page=92

20. Вареник О. С. ТРУДОВИЙ КОЛЕКТИВ ЯК СУБ'ЄКТ ЛОКАЛЬНОГО РЕГУЛЮВАННЯ ОХОРОНИ ПРАЦІ. Науковий вісник публічного та приватного права. 2021. № 4. С. 40–44. [Електронний ресурс] // Режим доступу: https://doi.org/10.32844/2618-1258.2021.4.7

21. Литвин, А., Лаун, С., Кролівець, Н. Охорона праці на підприємствах в умовах воєнного стану. Challenges and Issues of Modern Science, 1. 2023. С. 539–543. [Електронний ресурс] // Режим доступу: https://cims.fti.dp.ua/j/article/view/100

22. Курепін В. М. Оцінка ризиків на підприємстві: планування, практичні дії щодо профілактики виробничого травматизму. Актуальні проблеми та перспективи розвитку охорони праці, безпеки життєдіяльності та цивільного захисту : матеріали VI міжнародної науково-практичної конференції. Одеса : ОДАБА, 2024. С. 64-69. [Електронний ресурс] // Режим доступу: https://dspace.mnau.edu.ua/jspui/handle/123456789/17611

Додаток А
Апробація результатів роботи

ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
МІСЬКОГО ГОСПОДАРСТВА ІМЕНІ О. М. БЕКЕТОВА
КАФЕДРА КОМП'ЮТЕРНИХ НАУК ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

XVIII ВСЕУКРАЇНСЬКА НАУКОВО-ТЕХНІЧНА КОНФЕРЕНЦІЯ ЗДОБУВАЧІВ ВИЩОЇ ОСВІТИ (90-А НАУКОВО-ТЕХНІЧНА КОНФЕРЕНЦІЯ ХНУМГ ІМ. О. М. БЕКЕТОВА) «СТАЛИЙ РОЗВИТОК МІСТ: ПОСТВОЄННИЙ ПЕРІОД»

СЕРТИФІКАТ

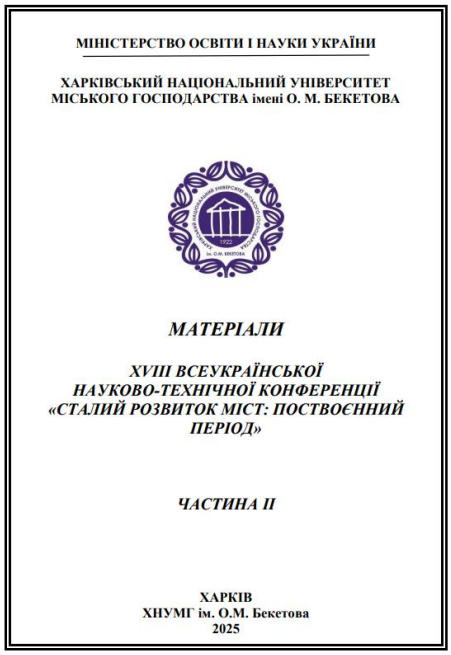
за кращу доповідь учасника секції
Комп'ютерних наук та інформаційних технологій

Діана Новікова

КЕРІВНИК СЕКЦІЇ
ЗАВІДУВАЧКА КАФЕДРИ КН ТА ІТ
ДОКТОР ФІЗ.-МАТ. НАУК, ПРОФЕСОР

МАРИНА НОВОЖИЛОВА

25 квітня 2025
Харків



МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

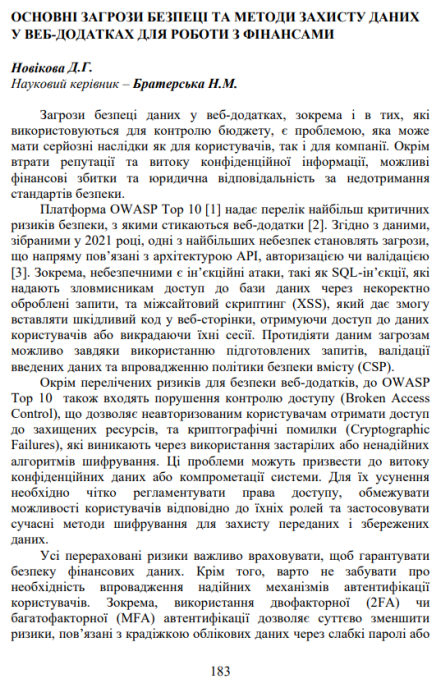
ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ МІСЬКОГО ГОСПОДАРСТВА імені О. М. БЕКЕТОВА

МАТЕРІАЛИ

XVIII ВСЕУКРАЇНСЬКОЇ НАУКОВО-ТЕХНІЧНОЇ КОНФЕРЕНЦІЇ «СТАЛИЙ РОЗВИТОК МІСТ: ПОСТВОЄННИЙ ПЕРІОД»

ЧАСТИНА ІІ

ХАРКІВ
ХНУМГ ім. О.М. Бекетова
2025



ОСНОВНІ ЗАГРОЗИ БЕЗПЕЦІ ТА МЕТОДИ ЗАХИСТУ ДАНИХ У ВЕБ-ДОДАТКАХ ДЛЯ РОБОТИ З ФІНАНСАМИ

Новікова Д.Г.
*Науковий керівник – **Братерська Н.М.***

Загрози безпеці даних у веб-додатках, зокрема і в тих, які використовуються для контролю бюджету, є проблемою, яка може мати серйозні наслідки як для користувачів, так і для компанії. Окрім втрати репутації та витоку конфіденційної інформації, можливі фінансові збитки та юридична відповідальність за недотримання стандартів безпеки.

Платформа OWASP Top 10 [1] надає перелік найбільш критичних ризиків безпеки, з якими стикаються веб-додатки [2]. Згідно з даними, зібраними у 2021 році, одні з найбільших небезпек становлять загрози, що напряму пов'язані з архітектурою API, авторизацією чи валідацією [3]. Зокрема, небезпечними є ін'єкційні атаки, такі як SQL-ін'єкції, які надають зловмисникам доступ до бази даних через некоректно оброблені запити, та міжсайтовий скриптинг (XSS), який дає змогу вставляти шкідливий код у веб-сторінки, отримуючи доступ до даних користувачів або викрадаючи їхні сесії. Протидіяти даним загрозам можливо завдяки використанню підготовлених запитів, валідації введених даних та впровадженню політики безпеки вмісту (CSP).

Окрім перелічених ризиків для безпеки веб-додатків, до OWASP Top 10 також входять порушення контролю доступу (Broken Access Control), що дозволяє неавторизованим користувачам отримати доступ до захищених ресурсів, та криптографічні помилки (Cryptographic Failures), які виникають через використання застарілих або ненадійних алгоритмів шифрування. Ці проблеми можуть призвести до витоку конфіденційних даних або компрометації системи. Для їх усунення необхідно чітко регламентувати права доступу, обмежувати можливості користувачів відповідно до їхніх ролей та застосовувати сучасні методи шифрування для захисту переданих і збережених даних.

Усі перераховані ризики важливо враховувати, щоб гарантувати безпеку фінансових даних. Крім того, варто не забувати про необхідність впровадження надійних механізмів автентифікації користувачів. Зокрема, використання двофакторної (2FA) чи багатофакторної (MFA) автентифікації дозволяє суттєво зменшити ризики, пов'язані з крадіжкою облікових даних через слабкі паролі або

183