ЗМІСТ

ВСТУП

Актуальність теми:

У сучасному світі, що характеризується динамічними економічними змінами, фінансовою нестабільністю та швидким розвитком інформаційних технологій, ефективне управління особистими фінансами стає не просто бажаним, а критично важливим навиком для кожної людини. Зростання складності фінансових ринків, різноманітність банківських продуктів, інвестиційних можливостей та непередбачуваність макроекономічних факторів (таких як інфляція, коливання валютних курсів) вимагають від індивідуумів значно глибшого розуміння та контролю за своїми грошовими потоками [13].

Традиційні методи обліку фінансів, такі як ведення записів у зошитах або використання простих електронних таблиць, виявляються недостатньо ефективними для повноцінного аналізу та довгострокового прогнозування. Вони не дозволяють швидко адаптуватися до змін, проводити глибокий аналіз даних, моделювати різні сценарії майбутнього та отримувати інтерактивний зворотний зв'язок. Сучасні користувачі потребують інструментів, які не тільки автоматизують рутинні процеси обліку доходів та витрат, а й надають можливість для аналізу, планування та, що найважливіше, прогнозування майбутнього фінансового стану з урахуванням різних факторів [21].

Таким чином, розробка інтерактивного веб-калькулятора фінансових прогнозів, який поєднуватиме в собі простоту використання, візуалізацію даних та розширені можливості прогнозного моделювання з гнучкими налаштуваннями сценаріїв та врахуванням зовнішніх чинників, є надзвичайно актуальною. Це дозволить користувачам не лише контролювати свої фінанси, але й приймати більш обґрунтовані рішення щодо заощаджень, інвестицій та планування майбутніх витрат, підвищуючи рівень їхньої фінансової грамотності та стабільності[25].

Об’єкт та предмет дослідження:

Об’єкт дослідження: Процеси персонального фінансового планування та прогнозування, а також автоматизація цих процесів за допомогою інформаційних технологій.

Предмет дослідження: Методи та моделі прогнозування фінансових показників, архітектурні рішення та технології розробки веб-застосунків для створення інтерактивного калькулятора фінансових прогнозів.

Мета дослідження:

Метою дипломної роботи є розробка та реалізація веб-калькулятора фінансових прогнозів, який надасть користувачам інструмент для ефективного управління особистими фінансами шляхом автоматизації обліку, візуалізації даних та застосування різних методів прогнозного моделювання з можливістю сценарного аналізу.

Завдання дослідження:

Для досягнення поставленої мети необхідно вирішити наступні завдання:

1. Проаналізувати предметну область персонального фінансового планування, існуючі підходи до обліку та прогнозування фінансових показників, а також вивчити потреби користувачів у відповідних програмних інструментах.
2. Здійснити огляд існуючих програмних рішень для управління фінансами, виявити їхні переваги та недоліки, а також визначити ключові функціональні вимоги до розроблюваної системи.
3. Обрати оптимальний технологічний стек для реалізації веб-калькулятора, обґрунтувати вибір мов програмування, фреймворків, баз даних та інших інструментів розробки.

4. Спроектувати архітектуру системи, визначити її основні компоненти, їхні функції та принципи взаємодії (клієнт-серверна архітектура, API-взаємодія).
5. Розробити інтерфейс користувача, який буде інтуїтивно зрозумілим, ергономічним та забезпечуватиме ефективну взаємодію з даними та функціоналом прогнозування.
6. Реалізувати модулі для обліку доходів та витрат користувача, забезпечити їхнє збереження та візуалізацію.
7. Імплементувати різні методи прогнозування фінансових показників, включаючи базові статистичні моделі (середнє значення, лінійна регресія) та більш просунуті підходи (імітація моделі Prophet), а також механізми коригування на зовнішні фактори (валютні коливання) та сценарного аналізу.
8. Здійснити тестування розробленого застосунку, перевірити коректність роботи всіх функціональних модулів та точність прогнозних моделей.

Наукова новизна:

Підхід до персонального фінансового менеджменту за акцентом на прогнозування. Цей проект інтегрує 3 різні моделі прогнозування в єдиний зручний для користувача інтерфейс.

Ключові слова:

Особисті фінанси, доходи, витрати, заощадження, інвестиції, фронтенд, бекенд.

РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ІСНУЮЧИХ РІШЕНЬ

1.1. Аналіз предметної області

Швидка глобалізація та високий темп життя в сучасному суспільстві висувають до людей багато вимог. Тому ми змушені контролювати різні сфери нашого життя, зокрема фінансову. Більше того, фінанси й економіка ґрунтуються не лише на житті окремої особи, але й на житті всієї країни та світу загалом. Вимоги до контролю зростають, стає неможливим відстежувати все та утримувати в голові великий обсяг інформації одночасно. Однак саме втрата контролю над фінансами призводить до найбільш негативних наслідків, зупиняє багато починань, а також викликає психологічний тиск. Саме тому зараз, як ніколи, важливо спростити й автоматизувати контроль власних витрат [1].

Ця проблема є актуальною для багатьох поколінь, тож створено безліч напрямів, які активно досліджуються та розвиваються. Серед них — фінансовий менеджмент, економічна кібернетика, впровадження інноваційних технологій тощо. Подібні дослідження проводились у роботах [24].

Фінансовий менеджмент — це процес прийняття рішень із планування з метою максимізації фінансового добробуту власника. Загалом фінансовий менеджмент — це сукупність певних принципів, методів і засобів організації та управління грошовими потоками. Він генерує, розподіляє, використовує фінансові ресурси й оптимізує їх. Основна мета фінансового менеджменту — забезпечення максимальної добробуту, раціонального використання ресурсів для отримання доходу (накопичення витрат), контролю витрат і підвищення ефективності.

У XXI столітті питання особистих фінансів стало особливо гострим. Статистика показує, що 3 із 4 людей щонайменше раз на місяць відчувають стрес через своє фінансове становище, а третина тих, хто перебуває у стосунках, заявляє, що саме гроші є основною причиною конфліктів із партнером [17]. Згідно з соціологічними дослідженнями, управління особистими фінансами є каменем спотикання в більшості сучасних родин, що призводить до сімейних негараздів. В Україні культура особистих фінансів недостатньо розвинена для проведення досліджень. Переважно облік і планування бюджету здійснюються інтуїтивно та неефективно [5].

Фінансовий консультант Ерік Тайсон у своїй книзі *«Особисті фінанси»* [5] наводить основні помилки, які трапляються у людей у цій сфері. Примітно, що сам Ерік Тайсон зізнається, що допустив майже кожну з них. Ось список причин основних негараздів в особистих фінансах:

1. Відсутність особистого фінансового планування — характерна риса більшості сімей. Небагато з них приділяють належну увагу плануванню й бюджетуванню особистих грошових потоків.
2. Нераціональні витрати. Навіть дуже багата людина може стати неплатоспроможною, якщо її витрати перевищують або практично дорівнюють доходам протягом певного періоду.
3. Нераціональні покупки за допомогою споживчих кредитів. Поширена причина напруги в особистому бюджеті — купівля дорогих, але неінвестиційних товарів у кредит.
4. Постійне відкладання моменту створення пенсійних заощаджень. Ніхто не хоче опинитися в стані середньостатистичного пенсіонера, проте більшість людей відкладають початок створення пенсійних накопичень, вирішуючи, що зроблять це завтра, через місяць, рік тощо.

5. Наявність на ринку складних фінансових продуктів зі прихованою завищеною ціною. У сучасних умовах розвитку фінансових послуг використовуються не тільки цінові й якісні, а й фінансові методи конкуренції. Це означає, що деякі продукти мають складну структуру ціноутворення, і дізнатися реальну вартість буває складно без фінансових обчислень.
6. Відсутність висновків з попереднього досвіду. Навіть фахівці з управління чи маркетингу іноді поводяться на ринку як нераціональні споживачі.
7. Прийняття рішень під впливом емоцій. Багато людей роблять покупки імпульсивно — не заради самого предмета, а заради вражень або самого процесу покупки.
8. Прийняття рішень без розділення інформації на суттєву й несуттєву. Маркетингові трюки сильно впливають на поведінку споживача.
9. Переоцінка рівня ризику при недооціненні економічної вигоди. Люди відмовляються від рішень через страх, навіть коли ризик насправді незначний.
10. Надмірна концентрація на короткострокових доходах, що заважає накопиченню довгострокового багатства.[5]

Основними принципами особистих фінансів також слід вважати раціональний баланс між "здоровими" та "шкідливими" зобов’язаннями. З точки зору особистих фінансів, здоровими є зобов’язання, пов’язані з інвестиціями в майбутнє (наприклад, іпотека, освіта тощо). Шкідливими є зобов’язання, що виникають у результаті споживання (наприклад, кредит на відпочинок або модний одяг).

За аналогією з розрахунком оборотного капіталу в корпоративних фінансах, особисте фінансове благополуччя зазвичай вимірюється як різниця між фінансовими активами та зобов’язаннями. Люди часто

стикаються з проблемою нестачі коштів, і причин може бути кілька: низька зарплата або надмірні витрати. У більшості випадків люди скаржаться на низький дохід, водночас замовчуючи або ігноруючи розмір своїх витрат — і нічого не змінюючи у своєму стилі життя.

Ніхто не заперечує, що стрес — головна хвороба людства. А гроші — одна з основних її причин. Життя не крутиться навколо грошей, але наше існування тісно пов’язане з наявністю фінансів. Отримання зарплати — не головне. Ефективне управління та планування — це значно більше, це набір необхідних знань і інструментів, які спрощують щоденні завдання.[10]

Ще однією важливою категорією в особистих фінансах є ануїтети. Ануїтет — це потік рівномірно розподілених у часі грошових платежів однакової суми [24]. В особистих фінансах ануїтети трапляються досить часто — наприклад, щомісячна плата за мобільний зв’язок або регулярні пенсійні внески.

1.2. Існуючі інструменти для фінансового планування

Сучасний ринок пропонує широкий спектр програмних рішень, призначених для допомоги користувачам в управлінні особистими фінансами. Ці інструменти варіюються від простих онлайн-калькуляторів до комплексних платформ, що інтегруються з банківськими рахунками та надають розширені аналітичні можливості. Огляд та аналіз найбільш популярних з них дозволяє виявити сильні та слабкі сторони існуючих рішень та обґрунтувати необхідність розробки власної системи з удосконаленим функціоналом.

Для аналізу були обрані веб-застосунки, які є широко відомими, мають значну аудиторію та пропонують різний рівень функціональності: Mint, Personal Capital, YNAB (You Need A Budget), PocketGuard.

1. Mint (Intuit Mint):

Mint є одним з найстаріших та найпопулярніших безкоштовних інструментів для управління особистими фінансами в Північній Америці, що належить компанії Intuit (також розробнику TurboTax та QuickBooks). Основна функція Mint полягає в агрегації фінансових даних: він дозволяє користувачам підключати свої банківські рахунки, кредитні картки, інвестиційні рахунки та навіть рахунки іпотеки, автоматично імпортуючи та категоризуючи транзакції. Застосунок надає єдиний, консолідований огляд усіх фінансів. Крім того, Mint пропонує інструменти для бюджетування, відстеження витрат, встановлення фінансових цілей та отримання сповіщень про наближення до лімітів бюджету чи прострочені платежі. Він також має функціонал для моніторингу кредитного рейтингу та пошуку заощаджень на основі аналізу витрат [32].

Переваги: Збирає інформацію з великої кількості фінансових установ, що значно економить час користувача. Використовує алгоритми для автоматичного розподілу транзакцій за категоріями, хоча іноді потребує ручної корекції. Надає зручні інструменти для створення та відстеження бюджетів, візуалізуючи прогрес. Додаткова корисна функція, що дозволяє користувачам стежити за своїм кредитним здоров'ям. Базовий функціонал доступний безкоштовно (монетизація відбувається за рахунок реклами та пропозицій фінансових продуктів).

Недоліки: Mint зосереджений переважно на відстеженні минулих та поточних даних. Функції прогнозування дуже базові, переважно базуються на простих історичних середніх і не дозволяють гнучко моделювати майбутні сценарії або враховувати зовнішні фактори. Хоча Intuit застосовує високі стандарти безпеки, необхідність надання облікових даних для

підключення банківських рахунків може викликати занепокоєння у деяких користувачів [31]. Автоматична категоризація не завжди точна і часто вимагає ручного втручання, що може дратувати. Наявність рекламних пропозицій фінансових продуктів може відволікати користувачів. Переважно орієнтований на ринок США та Канади, з обмеженою підтримкою банків з інших країн.

2. Personal Capital:

Personal Capital (який нещодавно ребрендований на Empower) є гібридним рішенням, що поєднує безкоштовні інструменти для управління фінансами з платними послугами фінансового консультування. Його безкоштовний функціонал включає агрегацію рахунків (банківські, кредитні, інвестиційні, пенсійні), відстеження чистої вартості, аналіз портфеля інвестицій, калькулятор виходу на пенсію та інструменти для бюджетування. Особливий акцент робиться на інвестиційному плануванні та аналізі комісій. Платні послуги надають доступ до сертифікованих фінансових консультантів [29].

Переваги: Чудово підходить для інвесторів, надаючи глибокий аналіз портфеля, диверсифікації, податкових наслідків та прихованих комісій. Автоматично розраховує та візуалізує зміну чистої вартості (активи мінус зобов'язання) з часом. Надає детальні інструменти для планування та прогнозування пенсійних накопичень, враховуючи різні сценарії. Можливість отримання допомоги від фінансових експертів (для платних користувачів)

Недоліки: Надлишок інвестиційних функцій може бути занадто складним для користувачів, які лише починають свій шлях у фінансовому плануванні та потребують базового обліку. Хоча бюджетування присутнє,

воно не таке детальне та гнучке, як у деяких інших спеціалізованих інструментах. Основна цінність розкривається через платні послуги з управління інвестиціями, що може бути недоступним для всіх користувачів. Менший акцент на щоденних витратах та доходах, що може не задовольнити потреби користувачів, яким потрібен детальний контроль за грошовим потоком.

3. YNAB (You Need A Budget):

YNAB є платним веб- та мобільним застосунком, що базується на унікальній методології бюджетування "кожному долару – робота". Його основна ідея полягає в тому, щоб кожен долар, який користувач отримує, був присвоєний певній категорії витрат або цілі. Це proactive-підхід до бюджетування, що відрізняється від реактивного відстеження минулих витрат. YNAB заохочує користувачів планувати свої витрати заздалегідь, а не лише відстежувати їх. Він підтримує підключення до банківських рахунків для імпорту транзакцій, але значний акцент робиться на ручній категоризації та усвідомленому розподілі коштів [37].

Переваги: Метод "нульового бюджету" доведено допомагає користувачам отримати повний контроль над своїми грошима, зменшити борги та збільшити заощадження. Заохочує користувачів приймати свідомі рішення про те, куди йдуть їхні гроші, до того, як вони їх витратять. YNAB надає велику кількість навчальних матеріалів, вебінарів та підтримку спільноти, що допомагає користувачам освоїти методологію. Дозволяє легко перерозподіляти кошти між категоріями.

Недоліки: YNAB є платною послугою, що може бути бар'єром для деяких користувачів. Методологія YNAB вимагає часу та зусиль для освоєння. Користувачам потрібно змінити своє фінансове мислення, що не

завжди легко. Хоча є імпорт транзакцій, акцент на ручному присвоєнні кожному долару "роботи" означає менше автоматизації порівняно з Mint, що може не підійти всім. YNAB фокусується на поточному та майбутньому бюджетуванні на найближчий місяць. Довгострокові прогнози та аналіз "що, якщо?" є досить обмеженими або потребують ручного моделювання.

4. PocketGuard:

PocketGuard – це веб- та мобільний застосунок, що спеціалізується на допомозі користувачам у відстеженні витрат та визначенні "скільки грошей можна безпечно витратити" (in my pocket). Він автоматично підключається до банківських рахунків, кредитних карток та інвестиційних портфелів, категоризує транзакції та створює спрощений бюджет. Ключовою особливістю є функція In My Pocket, яка показує суму грошей, доступну для витрат, після вирахування регулярних рахунків, заощаджень та бюджетних лімітів [34].

Переваги: Інтерфейс PocketGuard дуже інтуїтивний та мінімалістичний, що робить його доступним для новачків. Швидко імпортує та категоризує транзакції, спрощуючи процес створення бюджету. Інформує про рахунки, що наближаються, та інші фінансові події.

Недоліки: PocketGuard, як і Mint, переважно зосереджений на поточному стані та короткостроковому бюджетуванні. Розширених можливостей прогнозування майбутніх доходів/витрат або сценарного аналізу немає. Пропонує меншу гнучкість у створенні складних бюджетів порівняно з YNAB. Деякі корисні функції (наприклад, відстеження чистої вартості, особливі категорії) доступні лише в платній версії PocketGuard Plus. Хоча це і є перевагою для деяких, для інших це може бути надто

спрощений підхід, що не охоплює всі аспекти комплексного фінансового планування.

1.3. Постановка задачі

На основі проведеного аналізу предметної області та огляду існуючих рішень, виявлено потребу у створенні інструменту, який би забезпечував не лише базовий облік та візуалізацію фінансових даних, а й надавав користувачам розширені можливості прогнозування з високою гнучкістю та інтерактивністю.

Система повинна забезпечувати наступний функціонал:

- Можливість реєстрації та аутентифікації користувачів для доступу до персоналізованих фінансових даних.
- Надання інтерфейсу для додавання доходів та витрат за різними категоріями, а також їхнє надійне зберігання на сервері.
- Представлення історичних даних про доходи та витрати у вигляді інтерактивних графіків, для кращого розуміння фінансової динаміки.
- Надання користувачеві вибору між різними прогнозними моделями, для оцінки майбутніх фінансових показників.
- Можливість врахування впливу обмінного курсу валют на прогнозні показники.
- Реалізація функціоналу для моделювання гіпотетичних ситуацій та оцінки їхнього впливу на майбутній фінансовий стан.
- Чітке та наочне представлення результатів прогнозування у вигляді числових значень та графіків
- Забезпечення безперебійної комунікації між клієнтом та сервером для обробки даних та виконання прогнозних розрахунків.

Основною метою створення даного калькулятора фінансових прогнозів є надання користувачам інструменту, який допоможе їм:

1. Забезпечити прозорість доходів та витрат, а також спростити процес бюджетування.
2. Надати доступ до різних прогнозних моделей, включаючи ті, що імітують просунуті методи, для кращого розуміння потенційних майбутніх фінансових сценаріїв.
3. Шляхом впровадження методів, що враховують тренди, сезонність, зовнішні фактори та дозволяють проводити сценарний аналіз, досягти вищого рівня реалістичності прогнозів.
4. Візуалізувати складні фінансові дані та результати прогнозів у доступній формі, сприяючи кращому розумінню особистих фінансів.

Цей проект покликаний продемонструвати потенціал інтеграції сучасних веб-технологій та статистичних методів для створення ефективних інструментів персонального фінансового менеджменту.

Висновок

У цьому розділі було проведено комплексний аналіз теоретичних основ та обґрунтовано постановку задачі розробки веб-калькулятора фінансових прогнозів. Цей етап є фундаментом для подальшої розробки, оскільки він дозволив чітко визначити потреби користувачів, виявити прогалини в існуючих рішеннях та сформулювати конкретні вимоги до системи.

У підрозділі “1.1. Аналіз предметної області” було детально розглянуто сутність персонального фінансового планування як ключового елемента стабільного економічного життя сучасної людини. Підкреслено важливість обліку доходів та витрат, бюджетування, а також необхідність

довгострокового планування та прогнозування. З'ясовано, що в умовах динамічної економічної ситуації та впливу зовнішніх факторів, таких як інфляція та коливання валютних курсів, традиційні методи фінансового менеджменту виявляються недостатніми. Виявлено, що користувачі потребують інструментів, які не тільки автоматизують облік, а й надають можливості для глибокого аналізу та моделювання майбутніх сценаріїв.

Наступний підрозділ “1.2. Існуючі інструменти для фінансового планування” представив критичний огляд популярних веб-застосунків, таких як Mint, Personal Capital, YNAB та PocketGuard. Аналіз показав, що ці рішення успішно справляються з базовими функціями агрегації даних, категоризації транзакцій та візуалізації поточного фінансового стану. Проте було чітко встановлено, що більшість з них мають суттєві обмеження у сфері розширеного прогнозування. Це підтвердило існування незадоволеного попиту на більш потужні та інтерактивні інструменти для фінансового прогнозування.

На підставі всебічного аналізу предметної області та існуючих рішень у підрозділі “1.3. Постановка задачі” було сформульовано мету та конкретні завдання дипломної роботи. Головною метою є розробка веб-калькулятора фінансових прогнозів, який поєднає ефективний облік з розширеним функціоналом прогнозного моделювання та сценарного аналізу. Для досягнення цієї мети було визначено низку завдань, що охоплюють аналіз вимог, проектування архітектури та інтерфейсу, реалізацію модулів обліку та прогнозування, а також тестування системи.

Розділ 2. МЕТОДИ ПРОЕКТУВАННЯ СИСТЕМИ

2.1. Вибір технологій

Вибір технологічного стеку для розробки веб-застосунку є одним з ключових етапів проекту, що визначає його функціональність, продуктивність, масштабованість, безпеку та зручність подальшої підтримки. Рішення приймалося на основі аналізу функціональних та нефункціональних вимог до системи, доступності ресурсів, рівня складності реалізації, а також тенденцій на ринку програмної розробки. Метою було створення надійного, ефективного та легко розширюваного рішення, що відповідає сучасним стандартам веб-розробки [20].

Система побудована за принципами клієнт-серверної архітектури, що дозволяє чітко розмежувати відповідальність між представленням даних (фронтенд) та обробкою бізнес-логіки та зберіганням даних (бекенд). Цей підхід забезпечує гнучкість, покращує масштабованість та дозволяє паралельну розробку різних частин системи [6].

2.1.1. Клієнтська частина

Для реалізації фронтенду було обрано класичні веб-технології, які забезпечують високу сумісність з різними браузерами, гнучкість у дизайні та широкі можливості для інтерактивності.

- HTML5 (HyperText Markup Language 5): Як стандартизована мова розмітки, HTML5 є основою для створення структури та семантики веб-сторінок [26]. Його вибір обґрунтований універсальністю, підтримкою всіх сучасних браузерів та наявністю розширених можливостей для роботи з мультимедіа, інтерактивними формами та векторною графікою.

- CSS3 (Cascading Style Sheets 3): Мова стилів, що використовується для візуального оформлення веб-сторінок [27]. CSS3 надає широкі можливості для контролю над кольорами, шрифтами, розташуванням елементів, анімаціями та адаптивним дизайном. Вибір CSS3 дозволяє створити сучасний, привабливий та чуйний (responsive) інтерфейс, який коректно відображається на різних пристроях та розмірах екранів (від десктопів до мобільних телефонів).

- JavaScript (ES6+): Вибраний як основна мова програмування для реалізації клієнтської логіки та інтерактивності. JavaScript є стандартом для розробки інтерактивних веб-застосунків і підтримується усіма веб-браузерами [28]. Використання сучасних стандартів ECMAScript (ES6 і вище) забезпечує доступ до нових можливостей мови, таких як стрілкові функції, класи, обіцянки та асинхронні операції, що значно покращує читабельність коду та спрощує обробку асинхронних запитів до бекенду. JavaScript буде відповідати за обробку подій користувача, динамічне оновлення елементів інтерфейсу без перезавантаження сторінки, валідацію даних на стороні клієнта, взаємодію з RESTful API бекенду для отримання та відправлення даних та візуалізацію даних за допомогою бібліотек для побудови графіків (Chart.js).

Вибір цих технологій для фронтенду дозволяє створити гнучку та продуктивну клієнтську частину, яка відповідає вимогам до сучасного веб-застосунку.

2.1.2. Серверна частина

Для її розробки було обрано наступний стек технологій:

- Python: Як основна мова програмування для бекенду, Python був обраний завдяки своїй високій читабельності, простоті синтаксису, швидкості розробки та величезній екосистемі бібліотек [35]. Ці переваги роблять Python ідеальним вибором для проектів, що вимагають обробки даних, математичних обчислень та інтеграції з алгоритмами машинного навчання, що є критично важливим для функціоналу фінансового прогнозування.

- Flask: Використано як мікрофреймворк для Python, призначений для створення веб-застосунків та API. Flask є мінімалістичним, легким та гнучким фреймворком, який надає розробнику повний контроль над вибором компонентів та бібліотек. Його "мікро" природа означає, що він не нав'язує жорстких структур або абстракцій, дозволяючи розробнику будувати API саме так, як це потрібно для конкретного проекту [30].

- SQLite: Використано як систему управління реляційними базами даних (СУБД). SQLite є легкою, файловою базою даних, яка зберігає всі дані в одному файлі на диску, без необхідності окремого серверного процесу [36]. Це робить її ідеальною для розробки, тестування та демонстрації проекту, оскільки вона не потребує складної конфігурації, швидка у розгортанні та має низькі вимоги до ресурсів. Для проекту, що орієнтований на персональне фінансове планування, де обсяги даних є помірними, SQLite надає достатню продуктивність та надійність.

2.1.3. Інструменти розробки

Для забезпечення ефективного процесу розробки, тестування та управління проектом були обрані наступні інструменти:

- Visual Studio Code (VS Code): Використовується як інтегроване середовище розробки (IDE). VS Code є потужним, але легковажним

редактором коду, що підтримує широкий спектр мов програмування (Python, JavaScript, HTML, CSS), має вбудований термінал, розширені можливості налагодження, інтегрований контроль версій (Git) та величезну екосистему розширень, що значно підвищують продуктивність розробника [33].

- Git / GitHub: Система контролю версій Git застосовувалася для відстеження всіх змін у коді проекту, забезпечення колаборації, ефективного управління версіями та відкочування до попередніх станів за потреби. GitHub виступає як віддалений репозиторій для зберігання коду, що забезпечує його безпеку та доступність для спільної роботи.

2.2. Проектування архітектури

Проектування архітектури системи є ключовим етапом життєвого циклу розробки програмного забезпечення. Воно визначає загальну структуру системи, взаємодію її компонентів, розподіл відповідальності та механізми потоку даних. Метою цього етапу є створення стабільної, масштабованої, безпечної та легко підтримуваної системи, яка відповідає функціональним і нефункціональним вимогам проекту [2, с. 14]. Веб-калькулятор фінансових прогнозів спроектований на основі клієнт-серверної архітектури, що дозволяє чітко розмежувати логіку представлення даних від логіки їхньої обробки та зберігання.

Структура системи:

Розроблена система має дволанкову клієнт-серверну архітектуру, яка є класичною та широко застосовуваною для сучасних веб-застосунків. Ця архітектура дозволяє розподілити обчислювальні та логічні операції між

клієнтською частиною (веб-браузер користувача) та серверною частиною. Взаємодія між цими компонентами здійснюється за допомогою стандартного протоколу HTTP/HTTPS та архітектурного стилю REST (Representational State Transfer) [7, с. 78].

Основні принципи такої структури включають:

- Кожен компонент системи (клієнт, сервер, база даних) відповідає за свою специфічну функцію. Клієнтська частина займається відображенням даних та взаємодією з користувачем, тоді як серверна частина відповідає за бізнес-логіку, обробку даних, безпеку та взаємодію з базою даних. Таке розділення спрощує розробку, тестування, налагодження та подальшу підтримку системи.
- Різні частини системи можуть бути розроблені з використанням різних технологій, що дозволяє обирати найбільш ефективні інструменти для кожного завдання. Зміни в одній частині (наприклад, оновлення інтерфейсу) не вимагають переробки інших частин, за умови збереження стабільності API.
- У RESTful архітектурі, кожен запит від клієнта до сервера містить всю необхідну інформацію для обробки. Сервер не зберігає інформацію про стан клієнта між запитами, що підвищує надійність та масштабованість системи [19, с. 235].

Дана структура забезпечує високу ефективність обробки запитів, дозволяє реалізувати комплексний функціонал фінансового прогнозування та гарантує надійність зберігання даних

Основні компоненти та їх функції:

Архітектура веб-калькулятора фінансових прогнозів складається з трьох ключових компонентів: клієнтської частини, серверної частини та

бази даних. Кожен компонент виконує чітко визначений набір функцій, що забезпечує злагоджену роботу всієї системи.

1. Користувацька частина: Відповідає за безпосередню взаємодію з користувачем, відображення інформації та збір вхідних даних. Він реалізується за допомогою стандартних веб-технологій, які гарантують кросбраузерну сумісність та інтерактивність.
2. Серверна частина: Бекенд є центральним компонентом системи, що містить основну бізнес-логіку, забезпечує управління даними, взаємодію з базою даних, аутентифікацію користувачів та виконання складних обчислювальних алгоритмів, зокрема прогнозних моделей.
3. База даних: Реляційна база даних SQLite є компонентом для постійного та структурованого зберігання всіх даних, необхідних для функціонування застосунку.

Схема роботи калькулятора:

Взаємодія між основними компонентами системи — фронтендом, бекендом та базою даних — реалізується через послідовність HTTP-запитів та відповідей. Це забезпечує асинхронну взаємодію та високу швидкість відгуку застосунку.

- Користувач вводить облікові дані на фронтенді. JavaScript формує. Бекенд (Flask) валідує дані, хешує пароль або перевіряє його (для входу), взаємодіє з таблицею у базі даних. У разі успіху, бекенд повертає JWT-токен, який фронтенд зберігає для подальших авторизованих запитів.
- Користувач додає та переглядає, транзакції через фронтенд. JavaScript формує відповідні HTTP-запити (POST, GET, PUT,) до /api/expenses та /api/income. Бекенд обробляє запити, валідує дані, взаємодіє з таблицею records у базі даних для збереження, отримання, оновлення або видалення даних. Фронтенд оновлює інтерфейс згідно з отриманою відповіддю.

- Користувач на сторінці прогнозування обирає метод прогнозування та період, може задати параметри сценарію. JavaScript формує POST запит з цими параметрами та токеном до /api/prediction. Бекенд отримує запит, зчитує історичні дані користувача з таблиці records, застосовує обрану прогнозну модель, виконує сценарний аналіз або валютне коригування. Результати прогнозування надсилаються фронтенду у форматі JSON, який візуалізує їх у вигляді графіків та таблиць.

2.3. Проектування інтерфейсу

Проектування інтерфейсу користувача (UI) є критично важливим етапом у розробці будь-якого програмного продукту, оскільки саме інтерфейс є основним засобом взаємодії користувача з системою. Головна мета проектування інтерфейсу полягає у створенні інтуїтивно зрозумілого, зручного та ефективного середовища, яке дозволяє користувачам без зайвих зусиль виконувати поставлені завдання. Добре спроектований інтерфейс підвищує задоволеність користувачів, зменшує кількість помилок та прискорює освоєння функціоналу системи [15, с. 22].

Для веб-калькулятора фінансових прогнозів інтерфейс розробляється з акцентом на простоту, зрозумілість та швидкий доступ до ключових функцій. Він базується на принципах чуйного дизайну, що забезпечує коректне відображення та функціональність на різних пристроях та розмірах екранів [].

Структура сторінок:

Структура веб-сторінок калькулятора спроектована таким чином, щоб забезпечити логічну навігацію та чітке розділення функціоналу. Кожна сторінка має свою специфічну мету та організована для максимальної зручності користувача. Використання стандартних елементів

HTML5 для семантичної розмітки дозволяє не тільки покращити структуру, але й забезпечити доступність та SEO-оптимізацію [18, с. 88].
Система передбачає наступну структуру ключових сторінок:
1. Сторінка Реєстрації
2. Сторінка Входу
3. Головна сторінка користувача
4. Сторінка Прогнозування

Елементи управління:

Усі елементи управління на сторінках реалізовані за допомогою стандартних HTML-форм та елементів, стилізованих за допомогою CSS та пожвавлених JavaScript. Основна увага приділяється інтуїтивності та відповідності очікуванням користувачів щодо поведінки стандартних веб-компонентів.

1. Поля введення тексту: Використовуються для введення користувачем текстових даних (ім'я користувача, опис транзакції), електронної пошти та паролів. Реалізована валідація на стороні клієнта за допомогою JavaScript для перевірки формату email, довжини пароля, відповідності паролів, тощо, що надає миттєвий зворотний зв'язок користувачу [8, с. 187].
2. Числові поля введення: Використовуються для введення числових значень, таких як сума транзакції або відсоток зміни для прогнозу. JavaScript-валідація додатково перевірятиме, чи є введене значення дійсним числом та чи відповідає бізнес-логіці.
3. Кнопки: Ключові елементи для ініціювання дій: "Зареєструватися", "Увійти", "Додати транзакцію", "Зберегти", "Розрахувати прогноз".
4. Таблиці: Використовуються для організованого відображення списків даних.

5. Графіки: Використовуються для візуалізації фінансових даних. Будуть реалізовані за допомогою бібліотек JavaScript, таких як Chart.js, що дозволяє створювати інтерактивні графіки [22, с. 301].
6. Навігаційні елементи: Меню навігації для переходу між основними розділами застосунку.

Логіка взаємодії з користувачами:

Логіка взаємодії з користувачем визначає, як інтерфейс реагує на дії користувача та які процеси запускаються у відповідь. Вона реалізується переважно за допомогою JavaScript на стороні клієнта, що забезпечує динамічність, інтерактивність та швидкість відгуку без повного перезавантаження сторінок.

1. Початкове завантаження та перевірка аутентифікації:

При першому завантаженні застосунку, JavaScript перевіряє наявність автентифікаційного токена (наприклад, у Local Storage). Якщо токен присутній і валідний, користувач автоматично перенаправляється на Дашборд. Якщо токена немає або він недійсний, користувач залишається на стартовій сторінці або перенаправляється на сторінку входу.

2. Обробка форм та валідація:

При відправці будь-якої форми (реєстрація, вхід, додавання транзакції), JavaScript виконує первинну клієнтську валідацію даних. Це включає перевірку обов'язкових полів, коректності формату (email, числові значення), відповідності паролів, тощо [3, с. 145]. У випадку помилок валідації, користувач отримує миттєвий зворотний зв'язок, і відправка форми на сервер блокується. Лише після успішної клієнтської валідації, дані відправляються на бекенд за допомогою асинхронних HTTP-запитів

3. Асинхронна взаємодія з бекендом:

Більшість взаємодій, що вимагають обміну даними з сервером, будуть здійснюватися асинхронно. Після відправки запиту, інтерфейс може

відображати індикатор завантаження для інформування користувача. При отриманні відповіді від бекенду (у форматі JSON), JavaScript обробляє отримані дані. У разі успіху, інтерфейс динамічно оновлюється: додаються нові записи до таблиць, перемальовуються графіки, оновлюються зведені показники без перезавантаження всієї сторінки. У разі помилки, користувачеві відображається відповідне повідомлення.

4. Динамічне оновлення контенту:

Після додавання нової транзакції або редагування існуючої, дані на головному екрані. Додавання, редагування, видалення записів відображаються в таблиці в реальному часі. Фільтрація та сортування також призводять до динамічного оновлення вмісту таблиці. Після розрахунку прогнозу, графіки та табличні дані оновлюються, відображаючи результати.

5. Зворотний зв'язок користувачеві:

Використання тимчасових спливаючих повідомлень або повідомлень під полями форми для інформування користувача про успішне виконання дії або виникнення помилки. Візуальні підказки (спіннери, прогрес-бари) під час тривалих операцій.

Ця логіка взаємодії забезпечує плавний, інтуїтивно зрозумілий та ефективний користувацький досвід, мінімізуючи потребу в перезавантаженні сторінок та надаючи користувачеві актуальний зворотний зв'язок на кожному етапі роботи із застосунком.

Висновок

У цьому розділі було успішно виконано ключові етапи проектування веб-калькулятора фінансових прогнозів, що включали обґрунтований вибір

технологій, детальне проектування архітектури системи та ретельну розробку інтерфейсу користувача. Кожен з цих етапів мав вирішальне значення для створення функціонального, ефективного та зручного програмного продукту.

Вибір технологій базувався на принципах оптимальності, масштабованості та відповідності поставленим завданням. Для фронтенду було обрано класичний стек веб-технологій – HTML5, CSS3 та JavaScript (ES6+). Це рішення забезпечує високу гнучкість, кросбраузерну сумісність та можливість реалізації інтерактивного користувацького досвіду без залежності від важких фреймворків. Для серверної частини перевага була надана Python з мікрофреймворком Flask, що гарантує швидкість розробки, відмінні можливості для обробки даних та інтеграції з алгоритмами прогнозування завдяки багатій екосистемі бібліотек (NumPy, Pandas). Використання СУБД SQLite дозволило ефективно управляти даними, зберігаючи при цьому простоту розгортання на етапі розробки. Цей вибір забезпечує міцну технологічну основу для подальшої реалізації та потенційного масштабування проекту.

Проектування архітектури системи було реалізовано на основі дволанкової клієнт-серверної моделі з використанням RESTful API для взаємодії між компонентами. Така архітектура забезпечує чітке розділення відповідальності між фронтендом та бекендом, підвищуючи модульність, масштабованість та підтримуваність системи. Детально визначено функції кожного ключового компонента: клієнтської частини (відображення UI та взаємодія), серверної частини та бази даних. Окрему увагу було приділено інтеграції засобів безпеки, таких як аутентифікація на основі JWT-токенів, хешування паролів та валідація вхідних даних, що є фундаментальними для захисту конфіденційної фінансової інформації користувачів. Також було окреслено перспективи масштабування та розширення функціоналу, підтверджуючи гнучкість обраної архітектури.

Проектування інтерфейсу було спрямоване на створення інтуїтивно зрозумілого, естетичного та функціонального середовища для користувача. Була розроблена логічна структура сторінок, що охоплює всі ключові функціональні області застосунку: від сторінок аутентифікації до дашборду, управління транзакціями та розділу прогнозування. Деталізовано використання стандартних елементів управління HTML, їх стилізацію за допомогою CSS3 для адаптивного дизайну та динамічну поведінку за допомогою JavaScript. Особлива увага була приділена логіці взаємодії з користувачем, що включає клієнтську валідацію даних, асинхронні запити до сервера для оновлення контенту без перезавантаження сторінки, а також надання адекватного зворотного зв'язку користувачеві. Це забезпечить плавний та приємний досвід використання, дозволяючи користувачам легко вводити дані, відстежувати свої фінанси та отримувати прогнози.

Таким чином, комплексне проектування на етапах вибору технологій, архітектури та інтерфейсу заклало міцний фундамент для подальшої реалізації веб-калькулятора фінансових прогнозів. Обрані підходи та рішення дозволяють створити надійний, безпечний, масштабований та зручний інструмент для персонального фінансового менеджменту.

Розділ 3. Реалізація функціоналу калькулятора

3.1. Розробка розрахункових модулів

Розробка розрахункових модулів є центральним етапом у створенні функціоналу фінансового прогнозування. Ці модулі відповідають за збір історичних даних, застосування аналітичних моделей та генерацію прогнозів майбутніх фінансових показників. Ефективність та точність цих модулів безпосередньо впливають на цінність застосунку для користувача, надаючи йому інструменти для прийняття обґрунтованих фінансових рішень [11, с. 321]. У рамках даного проекту основна логіка прогнозування зосереджена у функції predict(), реалізованій на серверній стороні за допомогою фреймворку Flask.

Програмний модуль прогнозування фінансових витрат інтегрований у серверний бекенд системи управління особистими фінансами. Він є ключовим інструментом, що дозволяє користувачам оцінити свої потенційні витрати на майбутні періоди (3, 6 або 12 місяців) за допомогою трьох різних методологій. Доступ до модуля здійснюється через API-ендпоінт /api/predict за допомогою HTTP POST-запиту.

Модуль розпочинає роботу з перевірки автентифікації користувача. Це забезпечує, що доступ до фінансових даних та функціоналу прогнозування мають лише авторизовані користувачі. Якщо сесія не автентифікована, повертається помилка.

З тіла POST-запиту витягуються два ключові параметри, передані клієнтською стороною: горизонт прогнозування, що може приймати значення 3, 6 або 12 місяців та індекс обраного методу прогнозування (0 для базового, 1 для макроекономічного, 2 для сценарного аналізу).

Виконується базова валідація вхідних параметрів для запобігання некоректним запитам.

Модуль звертається до бази даних для отримання агрегованих щомісячних витрат користувача за останні 6 місяців. Ці дані формують історичний часовий ряд, який є основою для всіх прогнозних обчислень. На основі отриманих історичних даних розраховується середньомісячне значення витрат. Цей показник слугує відправною точкою для ініціалізації прогнозних моделей. Якщо історичних даних недостатньо або вони відсутні, встановлюється на нуль, щоб уникнути помилок і забезпечити коректну роботу модуля.

Для ефективної роботи з масивами числових даних на цьому етапі активно використовується бібліотека NumPy, яка надає оптимізовані функції для математичних операцій [16, с. 54].

Залежно від значення method_index, система переходить до відповідної логічної гілки, яка реалізує обраний метод прогнозування. Детальний опис кожного методу наведено у наступних підрозділах.

Після виконання прогнозних обчислень, результати разом з історичними даними та мітками для майбутніх періодів упаковуються у структурований JSON-об'єкт. Для методів "Базове прогнозування" та "Прогнозування з урахуванням макроекономічних факторів" (method_index 0 та 1) результати прогнозу зберігаються у спеціальній таблиці predictions в базі даних. Це дозволяє користувачам переглядати історію своїх прогнозів та аналізувати їх з часом. Сформований JSON-об'єкт надсилається як відповідь на клієнтську частину. Фронтенд використовує ці дані для візуалізації прогнозів у вигляді графіків та таблиць, надаючи користувачеві наочне уявлення про майбутні фінансові тенденції.

Опис методів прогнозування:

У модулі реалізовано три методи прогнозування, що надають користувачу комплексний погляд на можливі фінансові сценарії та дозволяють вибрати оптимальний підхід до планування.

1. Метод 0: Базове прогнозування:

Цей метод є найпростішим і слугує відправною точкою для прогнозування. Він базується на екстраполяції минулих тенденцій, припускаючи відносно стабільний розвиток фінансових показників.

Принцип роботи:

За основу береться розраховане середньомісячне значення витрат, отримане з історичних даних користувача. Для кожного майбутнього місяця в прогнозованому періоді (до 3, 6 або 12 місяців) до поточного прогнозного значення застосовується невеликий фіксований коефіцієнт зростання. Цей коефіцієнт імітує незначну природну інфляцію або поступове збільшення витрат, що є характерним для більшості економічних процесів [9, с. 112]. Щоб уникнути ідеально рівної лінії на графіку, яка не відповідає реальності, до кожного прогнозного значення додається невелике випадкове коливання (small_random_shock). Це коливання генерується за допомогою бібліотеки NumPy (np.random.uniform(-0.0001, 0.0001)) і моделює невеликі, непередбачувані флуктуації, які є невід'ємною частиною фінансових потоків [16, с. 145]. Для забезпечення відтворюваності результатів, генератор випадкових чисел ініціалізується фіксованим початковим значенням (np.random.seed(42)).

Формула розрахунку для кожного майбутнього місяця (t):

$$\text{Прогноз\_витрат}_t = \text{Прогноз\_витрат}_{t-1} \times (1 + \text{base\_trend\_rate} + \text{small\_random\_shock})$$

де Прогноз_витрат0=monthly_avg.

Цей метод є корисним для фінансових умов, що є відносно стабільними, коли не очікується значних та різких змін у структурі витрат користувача чи загальній економічній ситуації. Він надає базову, консервативну оцінку майбутніх витрат.

2. Метод 1: Прогнозування з урахуванням макроекономічних факторів:

Цей метод є більш складним, оскільки включає зовнішній економічний показник – інфляцію, що робить прогноз більш реалістичним в умовах мінливої економіки.

Принцип роботи:

Так само як і в Метод 0, за основу береться monthly_avg. Замість фіксованого коефіцієнта зростання використовується середній показник інфляції. У поточній реалізації дані про інфляцію імітуються **(**_get_nbu_inflation_data_mock()**)**, що симулює отримання цих даних від зовнішнього джерела, наприклад, Національного банку України. Це дозволяє демонструвати концепцію впливу макроекономічних факторів. Якщо імітовані дані інфляції доступні, розраховується середній показник інфляції за останні 6 місяців. Якщо даних немає, використовується базове значення інфляції (наприклад, 0.004 або 0.4% щомісяця). Прогноз розраховується шляхом щомісячного коригування суми витрат на розрахований середній рівень інфляції. Кожне наступне прогнознее значення множиться на (1 + effective_rate + random_shock), де effective_rate – це середній рівень інфляції. Додається також невелике випадкове коливання для більшої реалістичності.

Формула розрахунку для кожного майбутнього місяця (t):

$$\text{Прогноз\_витрат}_t = \text{Прогноз\_витрат}_{t-1} \times (1 + \text{average\_inflation\_rate} + \text{random\_shock})$$

де Прогноз_витрат0=monthly_avg.

Цей метод надає більш реалістичний прогноз в умовах мінливої економіки, де купівельна спроможність грошей змінюється з часом. Він дозволяє користувачеві враховувати вплив інфляції на майбутні витрати, що є критично важливим для довгострокового фінансового планування [14, с. 210].

3. Метод 2: Сценарний аналіз:

Сценарний аналіз є потужним інструментом для оцінки ризиків та можливостей. Він не дає єдиного "найкращого" прогнозу, а натомість моделює три можливі сценарії розвитку подій, дозволяючи користувачеві побачити спектр потенційних фінансових результатів.

Принцип роботи:

Метод генерує три незалежні прогнози на основі середньомісячних витрат (monthly_avg), застосовуючи до них різні коефіцієнти зростання, що відповідають трьом сценаріям:

- Песимістичний сценарій: Коефіцієнт зростання: −0.02 (зменшення витрат на 2% щомісяця). Це може імітувати вимушене зменшення витрат через фінансові труднощі, скорочення доходів або інші несприятливі фактори. Формула:

  $$\text{Прогноз\_витрат}_t = \text{Прогноз\_витрат}_{t-1} \times (1 - 0.02)$$

- Базовий сценарій: Коефіцієнт зростання: 0.0 (нульове зростання/спад). Цей сценарій передбачає стабільність фінансових

потоків, тобто майбутні витрати залишаються на рівні середніх історичних значень без суттєвих змін. Формула:

$$\text{Прогноз\_витрат}_t = \text{Прогноз\_витрат}_{t-1} \times (1 + 0.0)$$

- Оптимістичний сценарій: Коефіцієнт зростання: 0.03 (збільшення витрат на 3% щомісяця). Це може моделювати зростання витрат через збільшення доходів, покращення фінансового стану або можливість більших витрат на особисті потреби/інвестиції. Формула:

$$\text{Прогноз\_витрат}_t = \text{Прогноз\_витрат}_{t-1} \times (1 + 0.03)$$

Для кожного сценарію розрахунок виконується ітеративно для кожного майбутнього місяця, використовуючи monthly_avg як початкове значення для першого прогнозного місяця.

Сценарний аналіз є потужним інструментом для оцінки ризиків та розробки гнучких фінансових стратегій. Він дозволяє користувачу побачити повний спектр можливих фінансових результатів, зрозуміти потенційні "найкращі" та "найгірші" випадки, і підготуватися до різних життєвих ситуацій [23, с. 142].

Технічні аспекти реалізації модулів:

Реалізація розрахункових модулів у Flask-додатку включає кілька ключових технічних аспектів, що забезпечують їхню функціональність та надійність. Для всіх математичних операцій, зокрема розрахунку середнього значення, генерації випадкових коливань та ефективної обробки числових масивів, використовується бібліотека NumPy [16, с. 54]. Це забезпечує високу продуктивність обчислень та зручність роботи з числовими даними.

Модуль коректно обробляє випадки, коли у користувача немає достатньої історії витрат (менше 6 місяців або взагалі відсутні дані). У

такому випадку monthly_avg встановлюється на нуль, що дозволяє уникнути помилок і надавати базовий прогноз, навіть за відсутності повної історичної інформації.

Для методів 0 та 1, отриманий прогноз зберігається у спеціальній таблиці predictions у базі даних. Це дозволяє користувачеві переглядати історію своїх попередніх прогнозів, що є важливим для відстеження динаміки та оцінки точності моделей з часом.

Таким чином, розроблений модуль прогнозування є гнучким та багатофункціональним інструментом, що надає користувачеві важливу інформацію для прийняття обґрунтованих фінансових рішень, починаючи від простих екстраполяцій до аналізу складних сценаріїв.

3.2. Створення інтерфейсу

Створення інтерфейсу є безпосереднім втіленням попередньо розробленого дизайну та логіки взаємодії з користувачем. На цьому етапі статичні макети та концепції перетворюються на функціональні веб-сторінки, які забезпечують візуальну привабливість, інтуїтивну навігацію та динамічну взаємодію. Основна мета полягає у розробці клієнтської частини веб-застосунку, що ефективно обмінюється даними з бекендом і надає користувачеві вичерпну інформацію та можливості для управління фінансами [15, с. 19].

Для реалізації інтерфейсу використовуються фундаментальні веб-технології: HTML для структури, CSS для стилізації та JavaScript для інтерактивності та асинхронної комунікації.

Розробка структури сторінок за допомогою HTML:

Розробка структури сторінок є першоосновою будь-якого веб-інтерфейсу. HTML (HyperText Markup Language) використовується для створення скелета кожної веб-сторінки, визначаючи розташування всіх елементів контенту та інтерактивних компонентів. Акцент робиться на використанні семантичних тегів HTML5, що покращує читабельність коду, доступність для допоміжних технологій (наприклад, для людей з обмеженими можливостями) та оптимізацію для пошукових систем (SEO) [18, с. 88].

Кожна сторінка застосунку, від сторінок аутентифікації до основних функціональних панелей, має чітку та послідовну структуру. Ця структура зазвичай включає верхній заголовок (header) з логотипом та назвою застосунку, навігаційну панель (nav) для переміщення між ключовими розділами, основну область контенту (main), де розміщується специфічна для сторінки інформація та елементи управління, і нижній колонтитул (footer) з інформацією про авторські права. Наприклад, на авторизованих сторінках, у заголовку може відображатися ім'я користувача та кнопка для виходу з облікового запису.

Такий підхід забезпечує чистоту, логічність та легкість підтримки HTML-коду, що є основою для подальшого застосування стилів та інтерактивності.

Візуальне оформлення та адаптивний дизайн за допомогою CSS:

Візуальне оформлення інтерфейсу відіграє ключову роль у сприйнятті користувачем застосунку. CSS (Cascading Style Sheets) використовується для стилізації HTML-елементів, контролюючи їхній вигляд, розташування, кольори, шрифти та інші візуальні атрибути [4, с. 45].

Ключові аспекти використання CSS включають:

Застосування можливостей для створення гнучких та складних макетів, таких як Flexbox та CSS Grid. Це дозволяє ефективно розташовувати елементи на сторінці та забезпечувати їхню адаптивність до різних розмірів екранів. Додатково використовуються стилі для тіней, переходів та трансформацій для покращення інтерактивності та естетики інтерфейсу [4, с. 180].

Реалізація інтерактивності та асинхронної логіки за допомогою JavaScript:

JavaScript є динамічним ядром клієнтської частини застосунку, що відповідає за інтерактивність, обробку подій, валідацію даних та, що найважливіше, за асинхронну взаємодію з серверним бекендом. Використання сучасного стандарту ES6+ дозволяє писати більш чистий, ефективний та підтримуваний код [8, с. 201].

Ключові функції JavaScript у створенні інтерфейсу:

1. JavaScript динамічно змінює вміст, структуру та стилі HTML-сторінки у відповідь на дії користувача або дані, отримані з сервера. Це дозволяє додавати нові елементи, оновлювати існуючі або видаляти їх без перезавантаження всієї сторінки, забезпечуючи плавність переходу.
2. JavaScript відстежує та реагує на різноманітні дії користувача, такі як кліки по кнопках, введення тексту в поля, вибір елементів зі списків або навігація по сторінці. Це дозволяє запускати відповідні функції та логіку у відповідь на ці дії.
3. Перед відправкою даних на сервер, JavaScript виконує первинну перевірку коректності та повноти введених користувачем даних (наприклад, формат електронної пошти, довжина пароля, числові значення). Це надає миттєвий зворотний зв'язок користувачу, зменшує навантаження на бекенд та підвищує загальну швидкість

роботи застосунку [3, с. 187]. У випадку помилок валідації, користувач отримує візуальне сповіщення, і відправка форми блокується.

4. Більшість взаємодій, що вимагають обміну даними з сервером (додавання, редагування чи видалення транзакцій, отримання списків, виконання прогнозів), здійснюються асинхронно за допомогою Fetch API. Це дозволяє клієнтській частині надсилати запити на сервер та отримувати відповіді без блокування інтерфейсу та повного перезавантаження сторінки. Під час очікування відповіді може відображатися індикатор завантаження.
5. Після отримання даних від бекенду, JavaScript обробляє їх та динамічно оновлює відповідні частини інтерфейсу. Наприклад, на дашборді після додавання нової транзакції можуть бути оновлені зведені показники та перемальовані графіки. На сторінці транзакцій записи можуть бути додані, відредаговані або видалені в таблиці без перезавантаження.
6. Для наочного представлення фінансової інформації та прогнозів, JavaScript інтегрує спеціалізовані бібліотеки (Chart.js). Ці бібліотеки дозволяють динамічно створювати інтерактивні графікит, які автоматично оновлюються при надходженні нових даних [22, с. 301].
7. JavaScript надає користувачеві актуальний зворотний зв'язок щодо його дій. Це можуть бути спливаючі повідомлення про успішне виконання операції, візуальні індикатори завантаження під час очікування відповіді від сервера, або підсвічування полів форми та повідомлення про помилки валідації.

Таким чином, комбінація HTML, CSS та JavaScript дозволяє створити повноцінний, інтерактивний та візуально привабливий інтерфейс для веб-калькулятора фінансових прогнозів, який ефективно взаємодіє з

серверною частиною та надає користувачеві зручний інструмент для управління його фінансами.

3.3. Тестування системи

Тестування є невід'ємною частиною життєвого циклу розробки програмного забезпечення, що забезпечує відповідність розробленої системи заявленим вимогам та її функціональну стабільність. Метою тестування є виявлення дефектів, помилок та невідповідностей, а також підтвердження коректності роботи всіх модулів та загальної цілісності системи [12, с. 30]. Для веб-калькулятора фінансових прогнозів було проведено всебічне тестування, яке охоплювало функціональні аспекти, зручність використання та надійність.

Тестування охоплювало широкий спектр сценаріїв використання, що дозволило перевірити коректність роботи всіх ключових функцій застосунку. Нижче наведено основні сценарії тестування з демонстрацією їхнього виконання за допомогою скріншотів інтерфейсу.

Реєстрація та авторизація користувача:

Перевірка можливості успішної реєстрації нового користувача та подальшого входу до системи.

Очікуваний результат: Користувач успішно створює обліковий запис, отримує доступ до форми входу. При спробі регестрації з коректними даними – успішна авторизація.

Дії:

1. Відкриття сторінки реєстрації.
2. Введення унікального імені користувача, email та пароля.
3. Натискання кнопки “Зареєструватися” (див. Рис. 1).

4. Перенаправлення на сторінку входу
5. Введення зареєстрованих облікових даних на сторінці входу.
6. Натискання кнопки “Увійти” (див. Рис. 2).

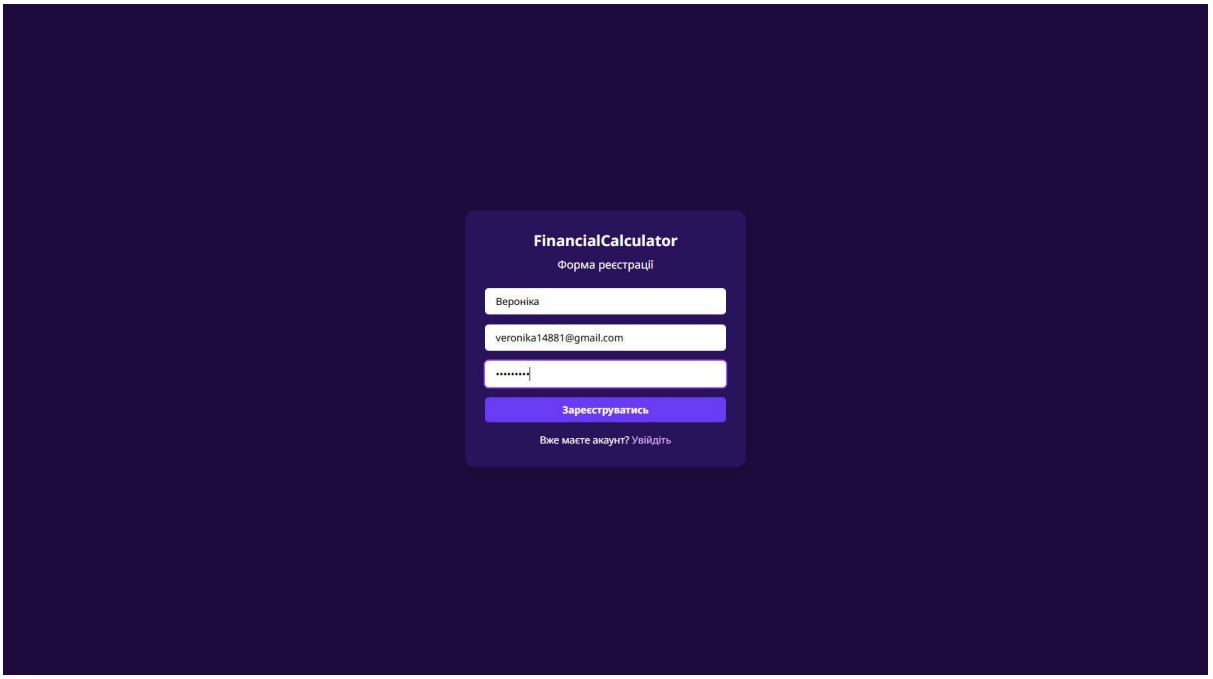

Рисунок 1 - Форма регестрації

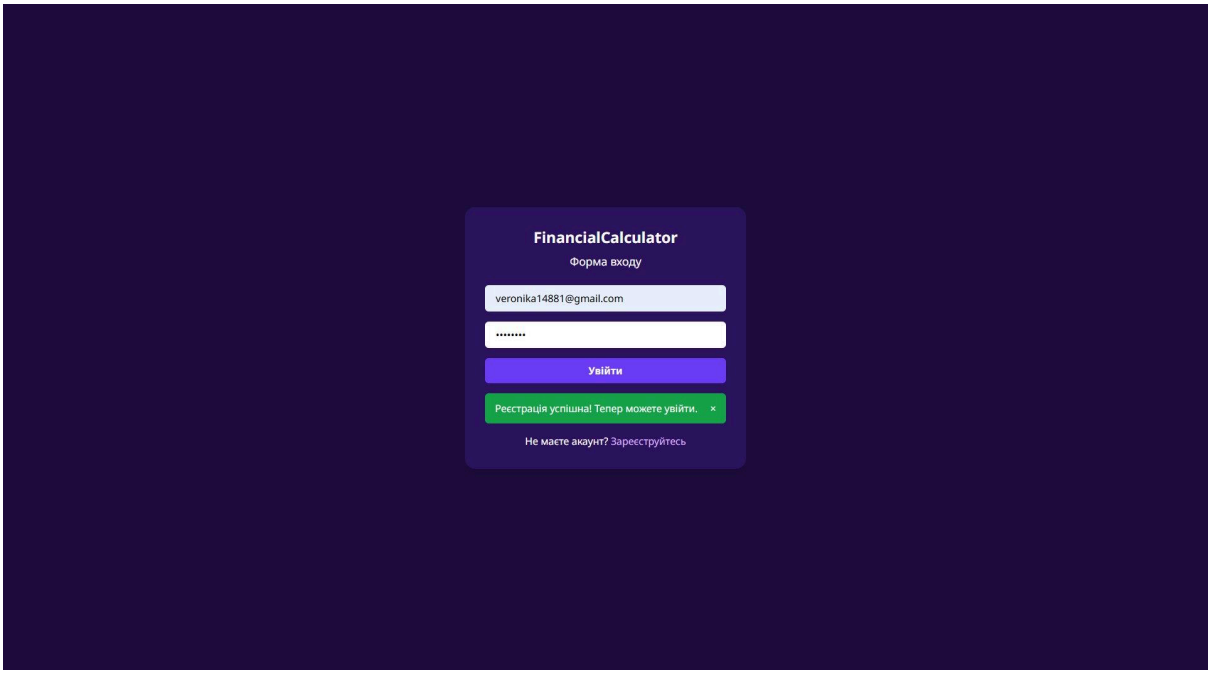

Рисунок 2 - Форма входу

Управління фінансовими транзакціями:

Сценарій: Перевірка функціоналу додавання, перегляду, редагування та видалення фінансових транзакцій.

Очікуваний результат: Користувач може коректно додавати доходи та витрати, всі транзакції відображаються у списку, можливе їхнє редагування та видалення з оновленням даних.

Дії:

1. Перехід на сторінку “Головна сторінка” (див. Рис. 3.1).
2. Додавання нової транзакції, наприклад, дохід (див. Рис. 4.1).
3. Додавання ще однієї транзакції, наприклад, витрата (див. Рис. 4.2).
4. Перевірка відображення нових записів у таблиці (див. Рис. 3.2).

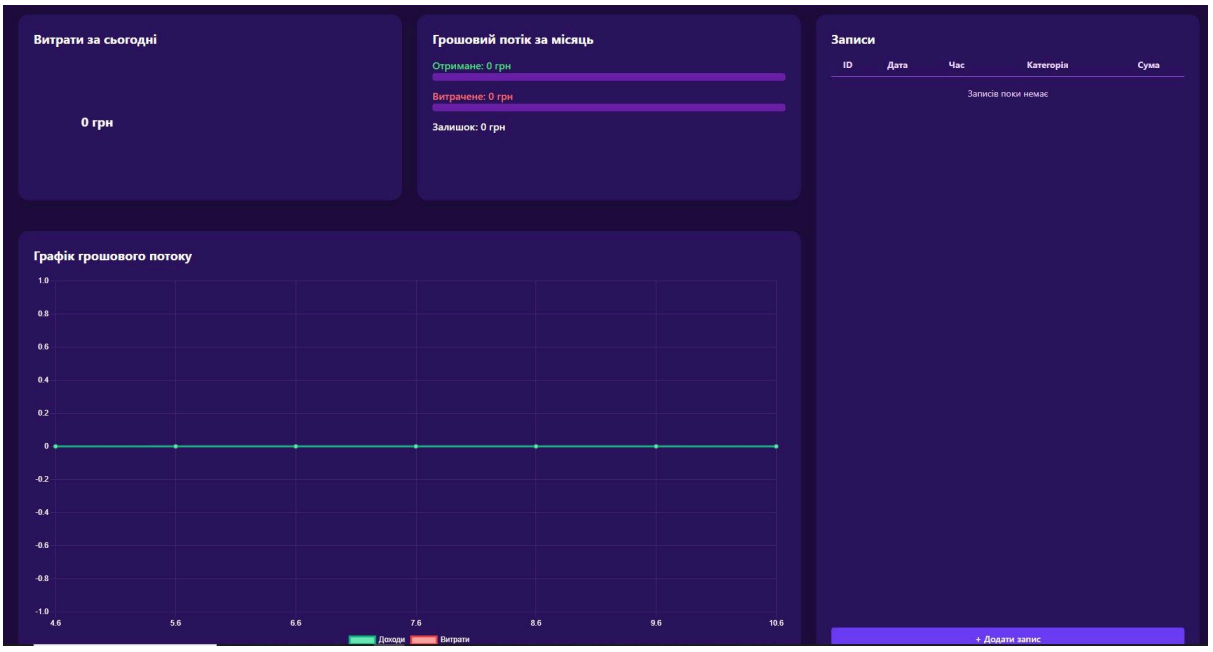

Рисунок 3.1 - Головний екран (порожній)

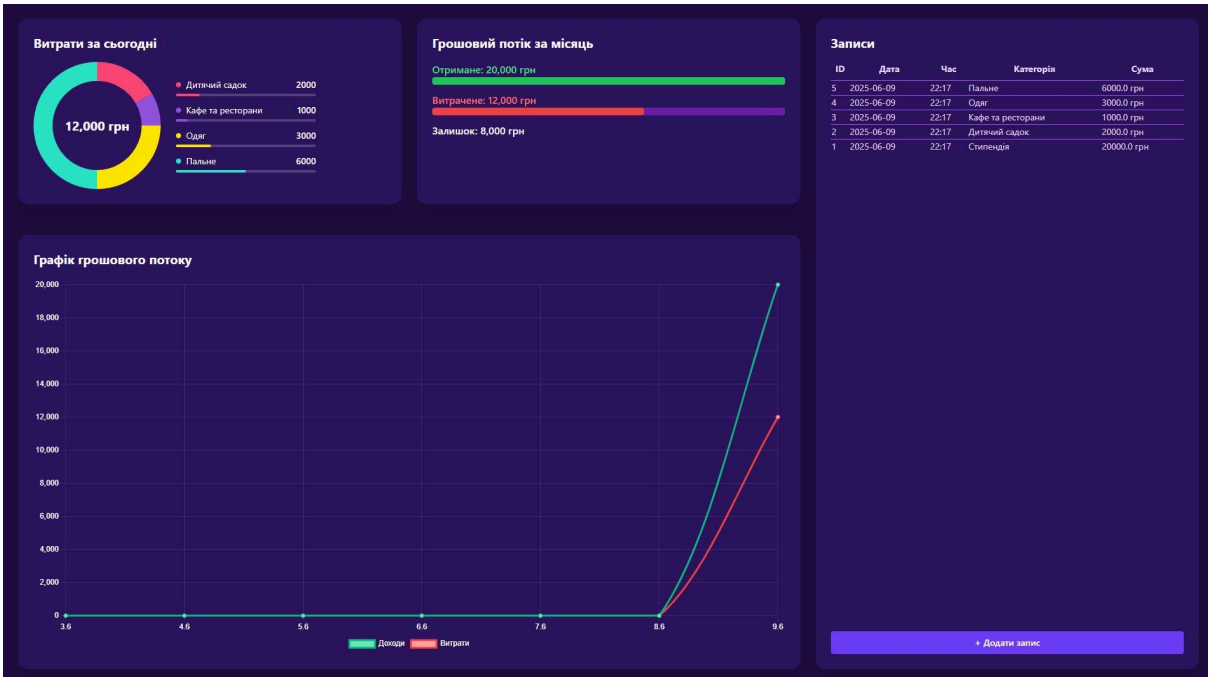

Рисунок 3.1 - Головний екран (порожній)

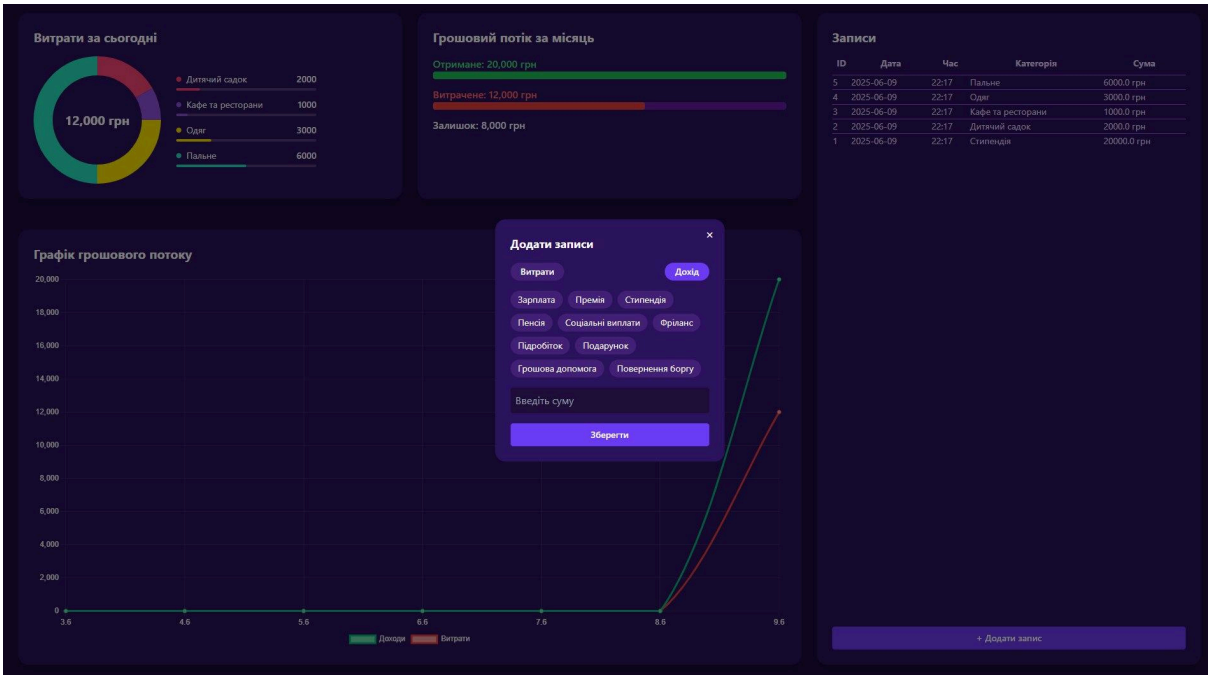

Рисунок 4.1 - Додавання записів - доходи

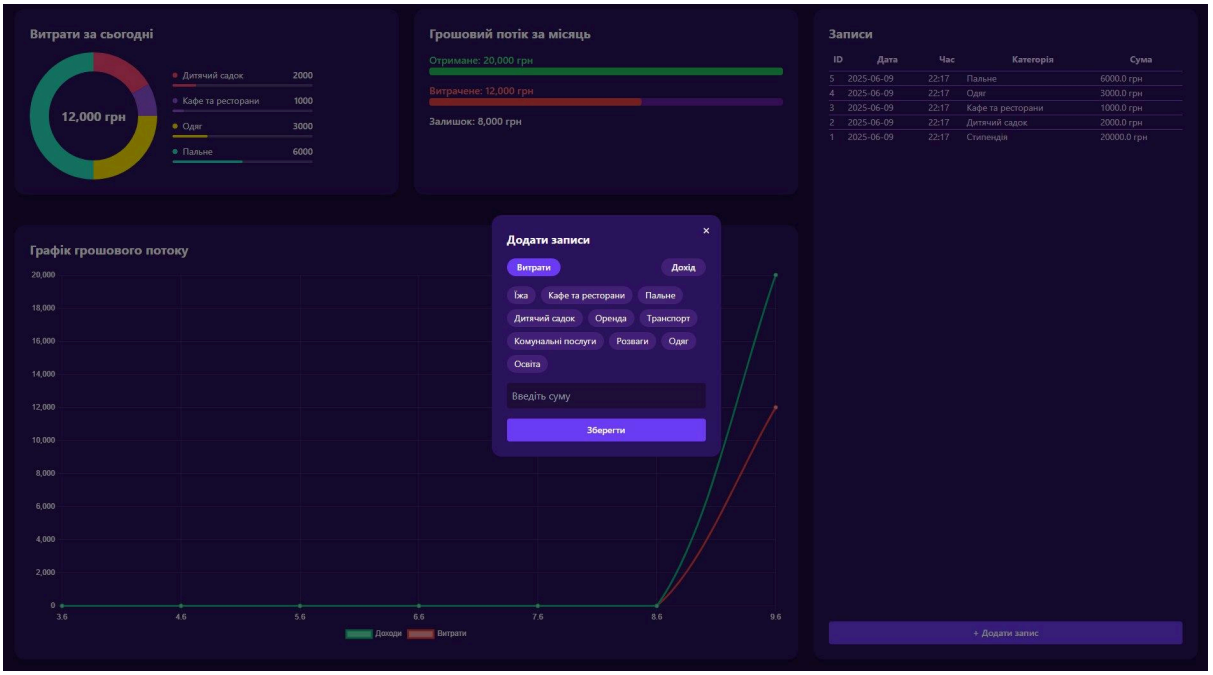

Рисунок 4.2 - Додавання записів - витрат

Виконання прогнозних розрахунків:

Сценарій: Перевірка коректності виконання всіх трьох методів прогнозування та відображення їхніх результатів.

Очікуваний результат: Модуль повертає коректні прогнозні значення для обраного періоду та методу, результати відображаються у вигляді графіків та таблиць.

Дії:

1. Перехід на сторінку “Прогнозування”.
2. Вибір “Базове прогнозування”, період, наприклад, 3 місяці.
3. Натискання кнопки “Розрахувати прогноз”.
4. Перевірка відображення графіка з прогнозними значеннями (див. Рис. 5.1).
5. Вибір “Прогнозування з урахуванням макроекономічних факторів”, період, наприклад, 1 рік.
6. Натискання кнопки “Розрахувати прогноз”.

7. Перевірка відображення графіка з прогнозними значеннями (див. Рис. 5.2).
8. Вибір “Сценарний аналіз”, період, наприклад, 3 місяці.
9. Натискання кнопки “Розрахувати прогноз”.
10. Перевірка відображення графіка з прогнозними значеннями (див. Рис. 5.3)

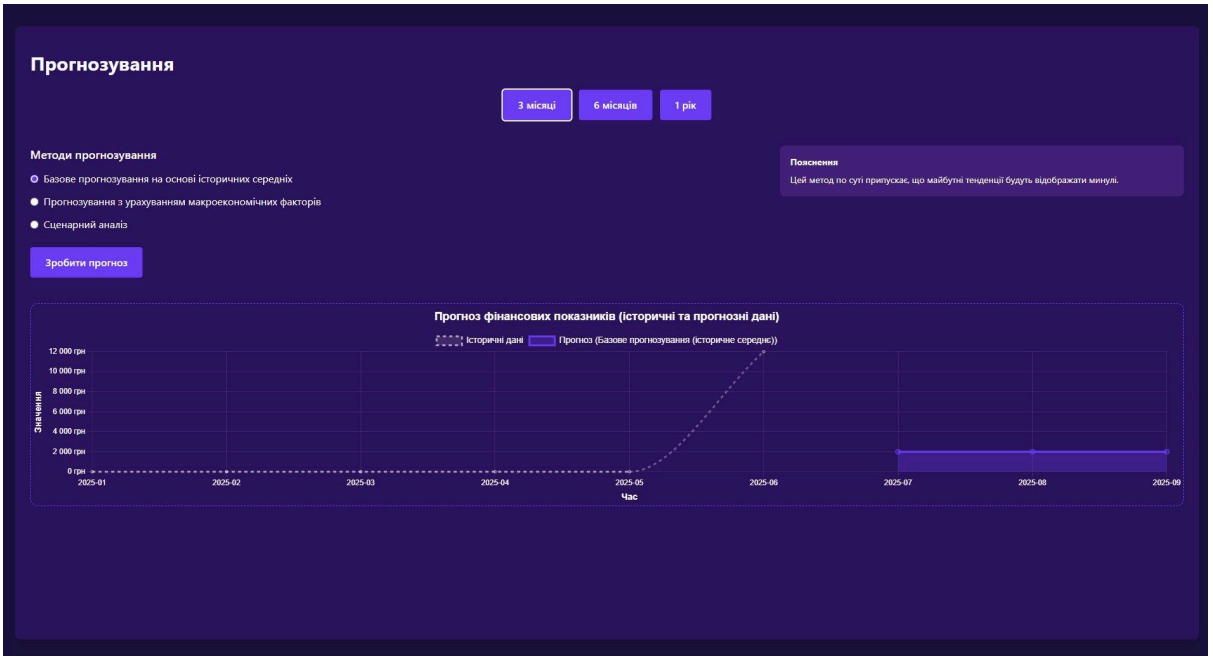

Рисунок 5.1 - Сторінка “Прогнозування” - Метод “Базового прогнозування…”

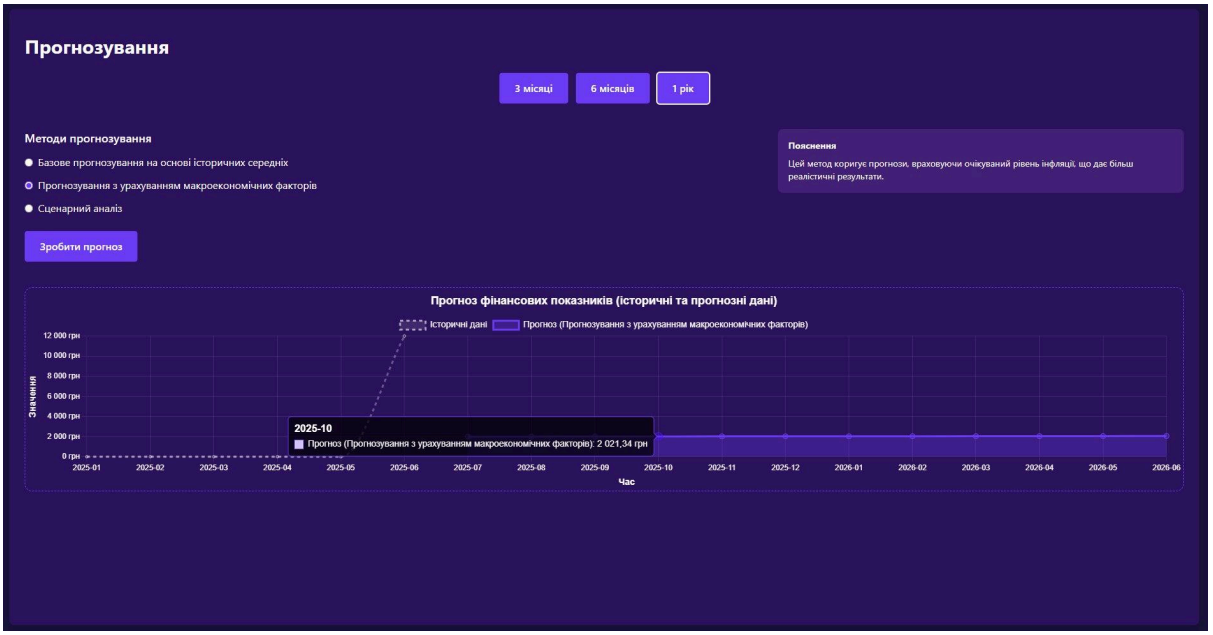

Рисунок 5.2 - Сторінка “Прогнозування” - Метод “Прогнозування з урахуванням макроекономічних факторів”

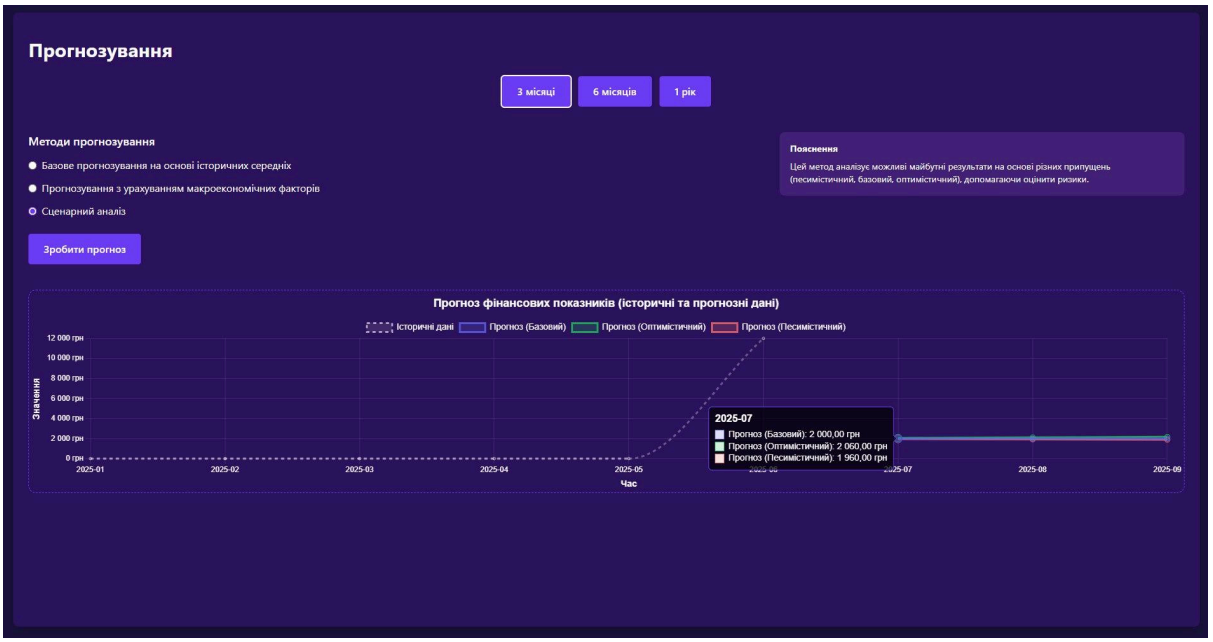

Рисунок 5.3 - Сторінка “Прогнозування” - Метод “Сценарний аналіз”

Результати тестування та виявлені дефекти:

У процесі тестування було виявлено ряд незначних дефектів, більшість з яких стосувалися інтерфейсу користувача та обробки

граничних умов. Усі виявлені критичні та значні дефекти були виправлені до завершення розробки. Приклади виправлених дефектів включають:

- Некоректне відображення великих чисел на графіках - Було скориговано масштабування осей графіків для забезпечення читабельності великих фінансових сум.
- Помилки валідації при введенні від'ємних сум транзакцій - Додано додаткові перевірки на стороні клієнта та сервера, що запобігають введенню від'ємних значень для доходів та витрат.
- Помилки у виведені графіків прогнозу - Графік виводив некоректні результати.

За результатами тестування, система продемонструвала високу функціональність та стабільність. Всі основні функції працюють коректно, інтерфейс є інтуїтивно зрозумілим, а прогнозні модулі надають очікувані результати.

Висновок

У цьому розділі було детально описано процеси реалізації ключових компонентів веб-калькулятора фінансових прогнозів: від розробки складних розрахункових модулів до створення інтуїтивно зрозумілого інтерфейсу користувача та забезпечення якості системи шляхом тестування. Ці етапи є фундаментальними для перетворення архітектурних рішень та дизайнерських концепцій на повноцінний, функціональний програмний продукт.

Розробка розрахункових модулів (Розділ 3.1) стала центральним елементом функціоналу прогнозування. Було успішно імплементовано три різні методології для передбачення майбутніх фінансових витрат: базове прогнозування на основі історичних середніх, прогнозування з

урахуванням макроекономічних факторів (інфляції) та сценарний аналіз. Кожен з цих методів пропонує користувачеві унікальний погляд на потенційні фінансові сценарії, від простої екстраполяції до оцінки ризиків за різних умов. Використання бібліотеки NumPy забезпечило високу точність та ефективність математичних обчислень. Цей модуль, реалізований на серверній стороні за допомогою Flask, є гнучким інструментом, здатним надавати користувачам обґрунтовану інформацію для фінансового планування.

Створення інтерфейсу (Розділ 3.2) було спрямоване на реалізацію користувацького досвіду, що відповідає принципам зручності та візуальної привабливості. Застосування HTML5 для структуризації, CSS3 для візуального оформлення та адаптивного дизайну, а також JavaScript для інтерактивності та асинхронної взаємодії дозволило розробити повноцінний клієнтський додаток. Особливу увагу було приділено логічній структурі сторінок (реєстрація, вхід, дашборд, транзакції, прогнозування), ефективній обробці подій, клієнтській валідації даних та динамічному оновленню контенту. Це забезпечує плавну навігацію та миттєвий зворотний зв'язок, покращуючи загальну взаємодію користувача із системою.

Тестування системи (Розділ 3.3) є завершальним етапом, що підтверджує функціональну надійність та якість розробленого програмного забезпечення. Застосування комбінованої методології, що включала модульне, інтеграційне, системне та приймальне тестування, дозволило всебічно оцінити систему. Були перевірені ключові функціональні сценарії, починаючи від реєстрації користувача та управління транзакціями до виконання складних прогнозних розрахунків. Результати тестування підтвердили коректну роботу всіх модулів та загальну стабільність системи, а виявлені незначні дефекти були успішно усунені.

Отже, етапи розробки розрахункових модулів, створення інтерфейсу та тестування системи є взаємопов'язаними та критично важливими для успішного впровадження веб-калькулятора. Вони забезпечують не тільки виконання основних функцій фінансового прогнозування, але й надають користувачеві зручний, надійний та ефективний інструмент для управління особистими фінансами.

ВИСНОВОК

У даній дипломній роботі було успішно розроблено та реалізовано повнофункціональний веб-застосунок для ефективного прогнозування та управління персональними фінансами. Проект продемонстрував застосування сучасних підходів до розробки програмного забезпечення, аналізу даних та інтерактивної візуалізації фінансової інформації, що є ключовим для підвищення фінансової грамотності та підтримки користувачів у прийнятті обґрунтованих рішень.

Ключові результати роботи включають:

Застосунок побудований на принципах дволанкової клієнт-серверної архітектури, що забезпечує чітке розділення відповідальності. Серверна частина, реалізована на Python із використанням фреймворку Flask, відповідає за зберігання та безпеку даних, аутентифікацію та авторизацію користувачів, а також за складну бізнес-логіку фінансового прогнозування. Клієнтська частина, розроблена із застосуванням HTML, CSS та JavaScript, надає користувачам інтуїтивно зрозумілий та інтерактивний інтерфейс для введення, перегляду та візуалізації фінансової інформації.

Одним із центральних досягнень є реалізація програмного модуля прогнозування, який включає три різні методології, а саме “Базове прогнозування на основі історичних середніх”, “Прогнозування з урахуванням макроекономічних факторів” та “Сценарний аналіз”.

Розроблений інтерфейс є централізованим хабом для користувача, що забезпечує не тільки зручне введення доходів та витрат, але й їхню інтерактивну візуалізацію у вигляді динамічних графіків та таблиць. Це дозволяє користувачеві отримати швидкий та комплексний огляд свого фінансового стану, аналізувати тенденції та приймати рішення на основі наочних даних. Чітке та наочне відображення прогнозних даних у формі

таблиць та графіків робить складну фінансову інформацію доступною та зрозумілою для широкого кола користувачів.

Проект продемонстрував ефективну взаємодію між клієнтською та серверною частинами для отримання історичних даних, виконання складних обчислень та повернення результатів, підтверджуючи розуміння принципів клієнт-серверної розробки та RESTful API. Проведене тестування системи підтвердило коректність роботи всіх функціональних модулів, стабільність застосунку та його відповідність заявленим вимогам.

Перспективи подальшого розвитку проекту передбачають розширення функціональних можливостей та підвищення його аналітичної потужності:

1. Підключення до актуальних зовнішніх джерел інформації (курсів валют, індексів інфляції, показників ринку) дозволить автоматизувати оновлення прогнозних моделей та підвищити їхню точність.
2. Впровадження просунутих статистичних методів та розширення можливостей порівняння кількох сценаріїв одночасно для глибшого фінансового аналізу.
3. Додавання інтерактивних елементів для підвищення мотивації користувачів до фінансового планування, а також розробка персоналізованих рекомендацій щодо оптимізації витрат та досягнення фінансових цілей.

Реалізований веб-застосунок є надійною основою для подальшого розвитку у сфері персонального фінансового менеджменту та може бути використаний як платформа для інтеграції нових аналітичних інструментів та функціональних можливостей.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

Книги (монографії, підручники, посібники):

1. A. Vasyliuk, T. Basyuk, Features of creating of employees' working hours interactive system, Proceedings of the 3nd International workshop on modern machine learning technologies and data science (MoMLeT+DS 2021). Volume 1: Main conference, LvivShatsk, Ukraine, June 5-6, 2021, Vol. 2917, pp. 344–356.
2. Bass, L., Clements, P., & Kazman, R. (2012). Software Architecture in Practice (3rd ed.). Addison-Wesley Professional.
3. Bordeleau, S. (2017). JavaScript for Web Designers. A Book Apart.
4. Cederholm, D., & Marcotte, E. (2017). Handcrafted CSS: More Bulletproof Web Design. New Riders.
5. E.Tyson, Personal Finance For Dummies, O'Reilly Media; 9th edition, 2018
6. Fowler, M. (2002). Patterns of Enterprise Application Architecture. Addison-Wesley.
7. Fielding, R. T. (2000). Architectural Styles and the Design of Network-based Software Architectures. University of California, Irvine.
8. Flanagan, D. (2020). JavaScript: The Definitive Guide (7th ed.). O'Reilly Media.
9. Fama, E. F. (1970). Efficient Capital Markets: A Review of Theory and Empirical Work. The Journal of Finance, 25(2), 383-417.
10. G.Arnold,The Financial Times Guide to Investing: The Definitive Companion to Investment and the Financial Markets (The FT Guides), FT Publishing International; 4th edition,2020
11. Hyndman, R. J., & Athanasopoulos, G. (2021). Forecasting: Principles and Practice (3rd ed.). OTexts.
12. Kaner, C., Falk, J., & Nguyen, H. Q. (1999). Testing Computer Software (2nd ed.). John Wiley & Sons.
13. Madura, J. (2018). Personal Financial Management. Cengage Learning.

14. Mishkin, F. S. (2015). The Economics of Money, Banking, and Financial Markets (11th ed.). Pearson.
15. Nielsen, J., & Budiu, R. (2012). Mobile Usability. New Riders.
16. Oliphant, T. E. (2015). Guide to NumPy. Trelgol Publishing.
17. P. Göstl, Risk Profile Contingent Analysis of Management Control Systems: Evidence from the Mechanical Engineering Industry (Unternehmensführung & Controlling), Springer Gabler; 1st ed. 2020 edition, 2019
18. Robbins, J. N. (2012). Learning HTML5: A Hands-On Guide to Creating HTML5 Websites. O'Reilly Media.
19. Richardson, L., & Amundsen, M. (2013). RESTful Web APIs. O'Reilly Media.
20. Sommerville, I. (2021). Software Engineering. Pearson Education.
21. Siegel, J. G., & Shim, J. K. (2014). Accounting Handbook. Barron's Educational Series.
22. Srivastava, V. (2020). Data Visualization with D3.js: Creating Data-Driven Graphics for the Web. Apress.
23. Shiller, R. J. (2015). Irrational Exuberance (3rd ed.). Princeton University Press.
24. T. Basyuk, A. Vasyliuk, Approach to a subject area ontology visualization system creating, Proceedings of the 5rd International Conference on Computational Linguistics and Intelligent Systems (COLINS-2021). Volume I: Main Conference, Kharkiv, Ukraine, April 22-23, 2021, Vol-2870: pp.528-540.
25. Wong, J. H. (2017). Personal Finance in a Digital World: Integrating Mobile Apps and Digital Tools. Routledge.
26. W3C. (2024). HTML5 Recommendation.
27. W3C. (2024). Cascading Style Sheets Level 2 Revision 1 (CSS 2.1) Specification.

Електроні ресурси:

28. ECMA International. (2024). ECMAScript 2024 Language Specification. веб-сайт. URL: https://tc39.es/ecma262/
29. "Empower (formerly Personal Capital) Review 2024: Is It Worth It?" (2024). Investopedia. веб-сайт. URL: https://www.investopedia.com/personal-capital-review-5070005
30. Flask. (2024). The Python micro framework for building web applications. веб-сайт. URL: https://flask.palletsprojects.com/
31. "Is Mint safe to use? The complete security review" (2024). NordPass.com. веб-сайт. URL: https://nordpass.com/blog/is-mint-safe/
32. Mint Review 2024: Is It Still the Best Free Budgeting App?" (2024). Forbes Advisor. веб-сайт. URL: https://www.forbes.com/advisor/banking/mint-review/
33. Microsoft. (2024). Visual Studio Code. веб-сайт. URL: https://code.visualstudio.com/
34. PocketGuard Review 2024: Does This Budgeting App Actually Work?" (2024). веб-сайт. URL: https://money.com/pocketguard-review/
35. Python. (2024). The Python Programming Language. веб-сайт. URL: https://www.python.org/
36. SQLite. (2024). SQLite Home Page. веб-сайт. URL: https://www.sqlite.org/index.html
37. "YNAB Review 2024: Our Experience With the Budgeting App" (2024). NerdWallet. веб-сайт. URL: https://www.nerdwallet.com/reviews/banking/ynab