

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ  
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ  
імені ІГОРЯ СІКОРСЬКОГО»  
Факультет інформатики та обчислювальної техніки

(повне найменування інституту, факультету)  
**Інформатики та програмної інженерії**

(повна назва кафедри)

«До захисту допущено»

Ілля АХАЛАДЗЕ

(підпис)

(ініціали, прізвище)

“ ”

2024 р.

## Курсова робота

з дисципліни Компоненти програмної інженерії

за освітньо-професійною програмою «Програмне забезпечення інформаційних  
управляючих систем та технологій»

спеціальності «121 Інженерія програмного забезпечення»

на тему

Веб-застосунок підтримки роботи пункту обміну валют

Виконав: студент III курсу, групи

ІП-13 Бондаренко Максим

Вікторович

(прізвище, ім'я, по батькові)

(підпис)

Керівник

асистент Ахаладзе Ілля Елдарійович

посада, науковий ступінь, вчене звання, прізвище, і ім'я, по батькові

(підпис)

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

Студент

(підпис)

Київ – 2024 року

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

Факультет (інститут) Інформатики та обчислювальної техніки  
(повна назва)

Кафедра інформатики та програмної інженерії  
(повна назва)

Рівень вищої освіти – перший (бакалаврський)

Спеціальність – *121 Інженерія програмного забезпечення*

Освітньо-професійна програма – *Програмне забезпечення інформаційних  
управляючих систем та технологій*

**ЗАТВЕРДЖУЮ**

\_\_\_\_\_  
(підпис) Ілля АХАЛАДЗЕ

“ ” \_\_\_\_\_ 2023 р.

**ЗАВДАННЯ  
НА КУРСОВУ РОБОТУ СТУДЕНТУ**

Бондаренку Максиму Вікторовичу  
(прізвище, ім'я, по батькові)

**1. Тема роботи** «Веб-застосунок підтримки роботи пункту обміну валют»

керівник роботи Ахаладзе Ілля Елдарійович, асистент  
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

**2. Термін подання студентом роботи** «31» грудня 2023 року

**3. Вихідні дані до роботи**

*Технічне завдання*

**4. Зміст пояснювальної записки**

*1) Аналіз вимог до програмного забезпечення: основні визначення та терміни,  
опис предметного середовища, огляд існуючих технічних рішень та відомих  
програмних продуктів, розробка функціональних та нефункціональних вимог*

*2) Моделювання та конструювання програмного забезпечення: моделювання та  
аналіз програмного забезпечення, засоби розробки, технічні рішення, архітектура  
програмного забезпечення*

*3) Розгортання та впровадження програмного забезпечення*

*4) Керівництво користувача, методика випробувань програмного продукту*

**5. Перелік графічного матеріалу**

1) Схема структурна варіантів використань

2) Схема бази даних

3) Схема структурна класів програмного забезпечення

## 6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання «6» жовтня 2023 року

## КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів виконання курсової роботи	Термін виконання етапів роботи	Примітка
1.	Вивчення рекомендованої літератури	12.11.2023	
2.	Аналіз існуючих методів розв'язання задачі	25.11.2023	
3.	Постановка та формалізація задачі	28.11.2023	
4.	Аналіз вимог до програмного забезпечення	04.12.2023	
5.	Алгоритмізація задачі	06.12.2023	
6.	Моделювання програмного забезпечення	15.12.2023	
7.	Обґрунтування використовуваних технічних засобів	15.12.2023	
8.	Розробка архітектури програмного забезпечення	19.12.2023	
9.	Розробка програмного забезпечення	24.12.2023	
10.	Налагодження програми	26.12.2023	
11.	Виконання графічних документів	27.12.2023	
12.	Оформлення пояснювальної записки	29.12.2023	
13.	Подання КР на перевірку	31.12.2023	
14.	Захист КР	05.01.2024	

Студент \_\_\_\_\_ Максим БОНДАРЕНКО  
(підпис)

Керівник \_\_\_\_\_ Ілля АХАЛАДЗЕ  
(підпис)

## АНОТАЦІЯ

**Структура та обсяг роботи.** Пояснювальна записка курсової роботи складається з 4 розділів, містить 21 рисуноків, 44 таблиць, 4 додатки, 17 джерел.

**Мета.** Розробка ставить перед собою досягнення наступних цілей:

- оптимізація процесу комунікації між клієнтом та пунктом обміну валют;
- надання користувачам актуальної інформації щодо змін валютних курсів, що дозволить їм приймати інформовані рішення щодо проведення обміну валют;
- створення ефективного механізму сповіщень для користувачів про зміни в курсах;
- забезпечення зручного адміністрування та моніторингу всіх створених запитів.

У розділі аналізу вимог були поставлені поставлені вимоги для програмного забезпечення.

У розділі моделювання програмного забезпечення було описано архітектуру програмного забезпечення та алгоритми вирішення прикладних задач.

У розділі аналіз якості були описані основні тест кейси, та стани системи після проведення тестування.

У розділі впровадження та супровід було описано процеси автоматизованого розгортання програмного забезпечення на виділеному сервері.

**КЛЮЧОВІ СЛОВА:** ВЕБ-ЗАСТОСУВАННЯ, ОБМІН ВАЛЮТ, ФІНАНСИ, FIREBASE

**Пояснювальна записка  
до курсової роботи**

на тему: Веб-застосунок підтримки роботи пункту обміну валют

КПІ.ІІ-1304.045440.02.81

Київ – 2024

## ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ.....	4
ВСТУП .....	5
1 АНАЛІЗ ВИМОГ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ .....	6
1.1 Загальні положення .....	6
1.2 Змістовний опис і аналіз предметної області .....	7
1.3 Аналіз існуючих технологій та успішних ІТ-проектів .....	8
1.3.1 Аналіз відомих алгоритмічних та технічних рішень .....	9
1.3.2 Аналіз допоміжних програмних засобів та засобів розробки.....	12
1.3.3 Аналіз відомих програмних продуктів.....	15
1.4 Аналіз вимог до програмного забезпечення .....	16
1.4.1 Розроблення функціональних вимог .....	24
1.4.2 Розроблення нефункціональних вимог .....	28
1.5 Постановка задачі .....	29
Висновки до розділу .....	30
2 МОДЕЛЮВАННЯ ТА КОНСТРУЮВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ.....	31
2.1 Моделювання та аналіз програмного забезпечення.....	31
2.2 Архітектура програмного забезпечення.....	34
2.3 Конструювання програмного забезпечення.....	36
2.4 Аналіз безпеки даних .....	40
Висновки до розділу .....	40
3 АНАЛІЗ ЯКОСТІ ТА ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ	42
3.1 Аналіз якості ПЗ.....	42
3.2 Опис процесів тестування.....	42
3.3 Опис контрольного прикладу .....	48
Висновки до розділу .....	52
4 ВПРОВАДЖЕННЯ ТА СУПРОВІД ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ.	53
4.1 Розгортання програмного забезпечення.....	53
4.2 Підтримка програмного забезпечення.....	54

Висновки до розділу .....	54
ВИСНОВКИ.....	55
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ.....	57

## ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

IDE	– Integrated Development Environment – інтегроване середовище розробки.
API	– Application programming interface, прикладний програмний Інтерфейс
IT	– Інформаційні технології
ER	– Entity-Relation diagram
БД	– База даних.
HTML	– HyperText Markup Language
JSX	– JavaScript XML
JS	– JavaScript
VS Code	– Visual Studio Code
BPMN	– Business Process Model and Notation
XSS	– Cross Site Scripting



## ВСТУП

У сучасному світі, де технологічний прогрес стає вирішальним чинником для розвитку різних галузей, надважливою стає роль інноваційних рішень у спрощенні та оптимізації буденних процесів. Враховуючи це, виникає актуальна потреба у створенні ефективних інструментів для обміну валют та взаємодії між клієнтом та фізичними пунктами обміну із застосуванням новітніх технологій.

Розробка веб-застосунку "CashFlow Exchange" ставить перед собою важливі цілі, спрямовані на вирішення актуальних проблем та вдосконалення валютних обмінних процесів. На тлі світових тенденцій у сфері фінансових технологій та сервісів, створення зручного інструменту для моніторингу курсів валют та ефективної взаємодії між клієнтами та обмінними пунктами стає важливим напрямом розвитку.

Сучасний стан об'єкта розробки дозволяє виявити прогресивні підходи та вже існуючі рішення, які використовують провідні наукові установи та організації. Детальний аналіз відомих розробок та наукових підходів розкриває сильні та слабкі сторони існуючих рішень, визначаючи можливості для подальшого вдосконалення та унікальні аспекти, які реалізуються в рамках даного проекту.

Ця курсова робота також зосереджена на розгляді сучасних глобальних тенденцій у сфері обміну валют та фінансових технологій, а також визначенні перспективних сфер застосування розробленого веб-застосунку.

Важливість інновацій у фінансовому секторі та необхідність вдосконалення процесів валютного обміну роблять цей проект високоактуальним та важливим для впровадження в практику фінансових послуг та економічної взаємодії.

# 1 АНАЛІЗ ВИМОГ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

## 1.1 Загальні положення

Фізичні пункти обміну валют<sup>[1]</sup> – обмінні пункти, які належать до певних кредитно-фінансових установ або до суб'єктів підприємницької діяльності, що діють на підставі агентських угод з банками, та мають ліцензію Національного банку України на здійснення операцій з торгівлі іноземною валютою.

Торгівля валютою, як явище, існує дуже давно та є однією з основних операцій обміну товарів між людьми. В давні часи люди міняли просто між собою різні речі, в сучасному ж світі відбувається обмін валюти на потрібні речі, але по всьому світу поширені безліч валют, які належать різним країнам і безпосередньо залежать від статусу та стабільності в країнах. Так, наприклад, американський долар є однією з найстабільніших та головною валютою світу, якраз через статус країни володаря. Люди купують американську валюту, будучи впевненими, що США будуть існувати, як стабільний гарант цієї валюти ще багато років. Оскільки валюта повністю залежить від держави, якій належить, то нестабільна ситуація в Україні змушує українців проводити маніпуляції з валютою задля збереження власних коштів та уникнути залежності від інфляції, бо нікому не відомо, що може статись з гривнею завтра. Вже було безліч прикладів сильного знецінення, як, наприклад, гіперінфляція в Німеччині в 1923 р., де темп становив  $3.25 \times 10^6$  відсотки на місяць (тобто ціни подвоювались кожні два дні), а найбільша інфляція в історії в Угорщині 1945 р. мала темпи  $1.295 \times 10^{16}$  відсотки протягом місяця. <sup>[2]</sup> В сучасному ж світі такі країни, як Аргентина та Турція потерпають від безперервної інфляції, саме тому обмін валют – дуже важливий процес для людства

Виходячи з пояснень, наданих раніше, дуже важливо забезпечити, щоб обмін валюти залишався простим і ефективним процесом, не перевантажуючи людей своїми складнощами. Зважаючи на це, сучасна сфера обміну валюти

вимагає нових підходів для взаємодії клієнтів із фізичними пунктами обміну. Ініціатива «CashFlow Exchange» спрямована на інтеграцію технологій, які автоматизують облік операцій, пропонують онлайн-доступ до курсів у реальному часі та створюють ефективний інструмент для залучення клієнтів.

Предметна область передбачає підхід до процесу обміну валют, який є буденним для багатьох людей, з новими технологіями для оптимізації процесу обміну валют між людиною та сервісом. В еру сучасних валютних бірж та криптотехнологій пункти обміну валют мають відповідати планці, яку задали різні існуючі проекти, та відповідати всім можливим вимогам користувача, оскільки процес обміну валют не має здаватись складним для людей, які бажають здійснити процес, який має бути максимально простим.

Напрями розвитку в цій сфері фокусуються на розширенні можливостей веб-додатка, зокрема, розробці інтуїтивних інтерфейсів, максимальне спрощення інтерфейсу задля зручності та зміни ставлення людей до процесу обміну валют.

## 1.2 Змістовний опис і аналіз предметної області

На сучасному етапі розвитку ІТ-технологій пункти обміну валют використовують веб-застосунки для підтримки своєї роботи, проте існують деякі важливі аспекти, які потребують уваги та оптимізації.

На сьогоднішній день, веб-застосунки для підтримки роботи пунктів обміну валют забезпечують функціонал, пов'язаний з реєстрацією та авторизацією користувачів, переглядом актуальних курсів валют, розрахунком обміну, створенням запитів на обмін та веденням історії операцій. Однак, існують обмеження у зручності та ефективності використання, такі як складність інтерфейсу, відсутність автоматизованих підказок для користувачів та обмежені можливості аналітики.

Недоліки поточного стану речей:

- Складний користувацький інтерфейс: Багато веб-застосунків валютних обмінників мають неінтуїтивний та заплутаний інтерфейс, що може ускладнювати використання користувачами.
- Відсутність персоналізації: Більшість застосунків не надають можливості налаштування особистого кабінету та персональних уподобань користувачів з приводу відстежування певних валют.
- Недостатній функціонал сповіщення клієнтів: Відсутність засобів сповіщення про зміну курсів валют може ускладнювати вчасне отримання потрібної інформації користувачем.

Можливі шляхи покращення ситуації в сфері ІТ:

- Удосконалення інтерфейсу: Розробка інтуїтивно зрозумілого та зручного інтерфейсу для користувачів, що сприятиме легкому використанню застосунку.
- Впровадження персоналізації: Розробка функцій налаштування та персоналізації особистого кабінету для зручності користувачів.
- Впровадження системи сповіщень: Впровадження системи, яка дозволяє користувачам вчасно отримувати важливу інформацію щодо курсів.

У межах курсової роботи покладено акцент на розробку веб-застосунку CashFlow Exchange, спрямованого на оптимізацію процесу обміну валют та полегшення комунікації між клієнтом та пунктом обміну. Проект передбачає впровадження інтуїтивного інтерфейсу, персоналізацію для користувачів та спрощення процесу обміну.

### 1.3 Аналіз існуючих технологій та успішних ІТ-проектів

Проаналізуємо відоме на сьогодні алгоритмічне забезпечення у даній області та технічні рішення, що допоможуть у реалізації застосунку підтримки роботи пункту обміну валют. Далі будуть розглянуті допоміжні програмні засоби, засоби розробки та готові програмні рішення. Для наглядного

порівняння візьмемо локальний продукт Money24, принцип роботи якого аналогічний до CashFlow Exchange.

### 1.3.1 Аналіз відомих алгоритмічних та технічних рішень

У розробці застосунку підтримки роботи пункту обміну валют важливо обрати оптимальні алгоритми та технічні рішення, що відповідають особливостям предметної області та завданням застосунку. Далі наведено аналіз відомих алгоритмів та технічних рішень, зроблено порівняльний аналіз та обрано оптимальні елементи для використання у розробці.

У сфері обміну валют та фінансових операцій існує кілька відомих рішень та платформ, які надають послуги обміну валют та підтримки фінансових операцій.

На локальному ринку я би розгледів застосунок Money24 (рисунок 1.3.1.4).

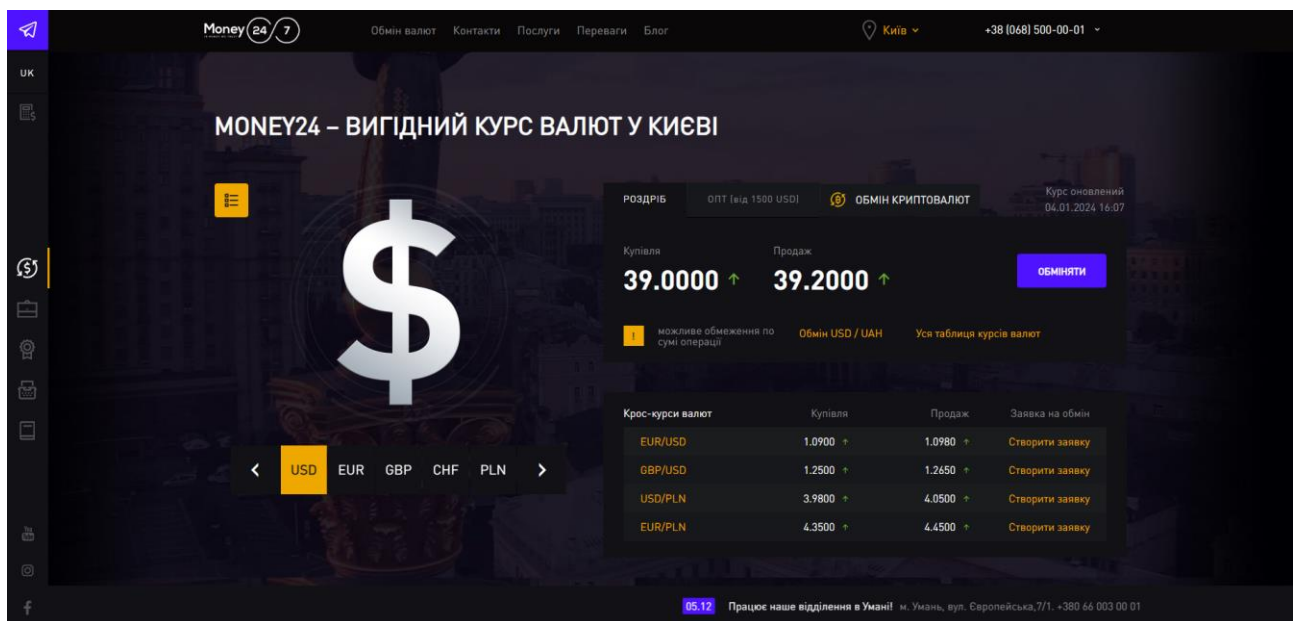


Рисунок 1.3.1.1 – застосунок підтримки роботи пункту обміну валют Money24.

Основний принцип роботи цього веб-застосунку схожий на те, що представлено в проекті CashFlow Exchange. Тут так само створюються запити на обмін (рисунок 1.3.1.2), які потім обробляють модератори, але функціонал користувачів досить обмежений, оскільки користувач не має власного облікового запису, який в нашому випадку дозволяє переглядати історію своїх

запитів та отримувати сповіщення про зміну курсів валют, це є важливим аспектом користувацького досвіду та оптимізації процесу комунікації між клієнтом та пунктом обміну, оскільки достатньо один раз створити акаунт та значно спростити процес створення тих же запитів в подальшому порівняно з Money24.

Рисунок 1.3.1.2 – вікно створення заявки на обмін Money24.

На світовому ринку я би розгледів два застосунки:

Wise - це сервіс для переказу грошей та обміну валют, який дозволяє користувачам здійснювати міжнародні перекази за більш низькими комісіями та реальним курсом валют (рис.1.3.1.2).

Рисунок 1.3.1.2 – конвертер валют Wise.

Особливості: Обмін валют без прихованих комісій, отримання місцевого банківського рахунку для зручних місцевих переказів тощо.

PayPal - популярна платіжна система, яка надає можливість здійснювати перекази грошей та обмін валют через свою платформу (рис. 1.3.1.3).

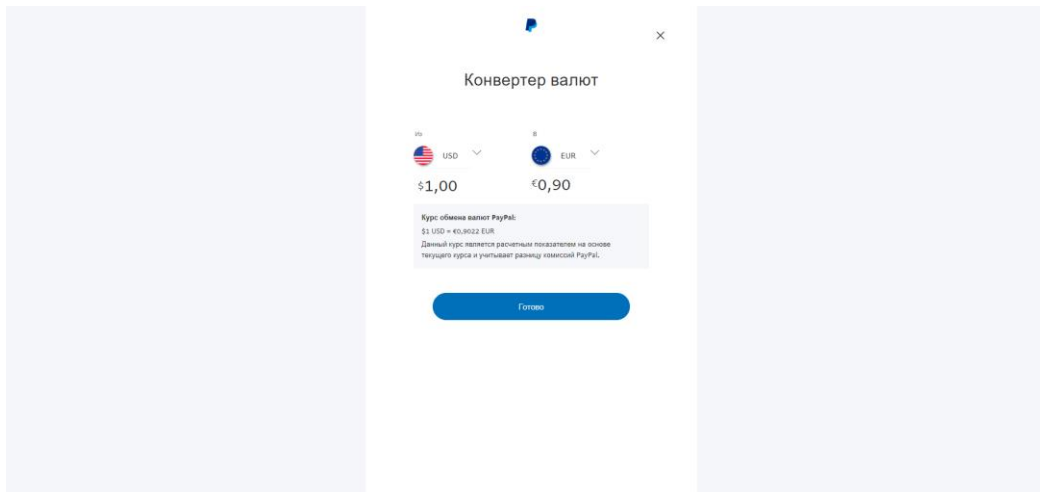


Рисунок 1.3.1.3 – конвертер валют PayPal.

Особливості: Легкість використання, міжнародний охоплення, різноманітні опції фінансових операцій тощо.

Firebase<sup>[3]</sup> надає потужні інструменти для реалізації безпечної та ефективної системи реєстрації та авторизації користувачів. Firebase Auth використовує алгоритм хешування для безпечного зберігання паролів в базі даних. Захист паролів за допомогою алгоритму хешування забезпечує конфіденційність особистих даних користувача. Після успішної реєстрації користувача, Firebase Auth використовує алгоритм для перевірки введеного пароля з збереженим хешем. У разі введення правильного пароля, користувач отримує унікальний токен, який використовується для авторизації при подальших запитаннях до сервера. Застосування токенів для авторизації забезпечує безпеку взаємодії із сервером.

Користувач виражає запит до сервера Firebase, вказуючи критерії пошуку або унікальний ідентифікатор запису. Firebase, використовуючи свою систему індексації, швидко знаходить та витягує необхідні дані. Переваги: швидкий доступ до даних завдяки ефективній системі індексації Firebase. Можливість синхронізації даних в реальному часі за допомогою Firestore.

Усе описано вище залежить від швидкості та стабільності Інтернет-з'єднання для ефективної роботи, але сьогодні Інтернет покриття не є проблемою для населення, особливо потребуючого регулярний обмін валют.

Інформація про актуальні курси валют надходить з зовнішньої API, що гарантує надійність та актуальність інформації. Використання готової API значно спрощує процес розробки, оскільки не потрібно витратити час на складний та потенційно небезпечний процес скрапінгу або парсингу даних. Також, оскільки зовнішні API призначені для ефективної взаємодії з віддаленими серверами, це дозволяє отримувати необхідні дані швидко та ефективно, зменшуючи навантаження на власний сервер. Також це дозволяє легко масштабувати функціонал застосунку, додаючи або змінюючи джерела даних без необхідності переробки всього збірника або скрапера.

На жаль, використання зовнішніх API робить застосунок залежним від інших сервісів, але існує безліч різних доступних сервісів, які зарекомендували себе, як надійні та стабільні.

Отже, використовуючи Firebase, можна забезпечити швидкий та ефективний доступ до БД, реалізувати аутентифікацію та безпеку даних, зовнішні API дозволяють отримувати актуальну інформацію про курси, Firebase дозволяє легко адаптувати систему до змінних потреб користувача через свою гнучкість та розширюваність, також пропонує готові рішення для обробки та зберігання даних без значних витрат на інфраструктуру, Firebase має вбудовані засоби для забезпечення безпеки та прискорення швидкості обробки даних. Також важливим аспектом є низька вартість підтримки проекту на Firebase, що є перевагою для невеликих застосунків.

### 1.3.2 Аналіз допоміжних програмних засобів та засобів розробки

У ході розробки веб-застосунку використовувалась низка допоміжних програмних засобів, які включають мови програмування, фреймворки та інструменти розробки. Нижче наведено опис та порівняльний аналіз цих засобів



JavaScript (JS)<sup>[5]</sup>: Мова програмування, яка використовується для розробки динамічних веб-сайтів та веб-застосунків. Переваги: найпоширеніша мова для фронтенд-розробки. Має велику спільноту та екосистему, багато готових рішень, підтримується всіма сучасними браузерами.

React<sup>[4]</sup>: JavaScript-бібліотека для побудови інтерфейсів користувача, заснована на компонентах. Найвідомішим аналогом React є Vue.js. Використання JSX у React дозволяє вписувати HTML-подібний код безпосередньо в JavaScript, що спрощує створення компонентів. Vue.js дозволяє використовувати HTML-подібний код у файлах шаблонів, роблячи його більш зрозумілим та читабельним. React використовує віртуальний DOM для оптимізації та прискорення процесу оновлення інтерфейсу. Vue.js вбудовує реактивність безпосередньо в систему, що спрощує відслідковування змін у даних та автоматичне оновлення інтерфейсу. React надає більше гнучкості у виборі та налаштуванні інструментів та бібліотек. React має широку та активну спільноту розробників та безліч корисних бібліотек. Спільнота Vue.js менша, але вона стрімко зростає, а фреймворк має добру екосистему.

Firebase: Облачна платформа від Google для розробки веб-застосунків, яка надає ряд послуг, таких як база даних, аутентифікація, зберігання та інше. Переваги: Firebase надає повний спектр інструментів для роботи з базою даних, аутентифікацією, зберіганням файлів та іншими ключовими функціональностями. Вибір Firebase може прискорити розробку та спростити управління серверною частиною.

Visual Studio Code (VS Code): Легкий та потужний текстовий редактор від Microsoft з великою кількістю розширень для розробників. VS Code підтримує різні мови програмування та технології, що робить його універсальним інтерфейсом. Один із аналогів Visual Studio Code - Atom. VS Code використовує ефективний та швидкий двигун, що робить роботу з кодом ефективною та безперервною. Atom використовує JavaScript для свого інтерфейсу, що може призводити до меншої швидкодії та продуктивності, особливо в обробці великих файлів. VS Code має широкий магазин розширень, який дозволяє легко

встановлювати та управляти розширеннями. Atom також має можливості розширення, але його екосистема може бути менш розвиненою порівняно із VS Code. VS Code добре працює навіть на менш потужних комп'ютерах і не вимагає великої кількості ресурсів. Atom може використовувати більше ресурсів, особливо при роботі з великими проектами та файлами. VS Code має вбудовану підтримку Git, що полегшує ведення версій та співпрацю з репозиторіями. В Atom для повноцінної підтримки Git може бути потрібно встановлювати деякі додаткові плагіни.

Vite<sup>[6]</sup>: Швидкий та мінімалістичний інструмент для розробки веб-застосунків на базі JavaScript. Переваги: Дозволяє швидко розгортати проекти та надає широкий спектр можливостей. Один із аналогів - Webpack. Vite швидше за рахунок використання ESM, гарантує швидке перезавантаження браузера при внесенні змін, має простий конфігураційний файл, що робить його більше зрозумілим та легким.

Redux<sup>[7]</sup>: Управління станом застосунків JavaScript, часто використовується з React. Переваги: Допмагає підтримувати прозорий та простий стан додатку, зручний для великих застосунків. Одним з аналогів Redux є MobX. Обидві бібліотеки застосовуються для управління станом додатку в JS-проектах, але кожна має свої особливості. Redux базується на централізованому збереженні стану, що забезпечує структурованість та передбачуваність управління даними в додатку, в той же час MobX дозволяє кожному об'єкту в додатку мати свій власний стан, що може призводити до менш очевидного контролю та більш складного управління станом у великих проектах. Redux використовує чисті функції (reducers) для зміни стану, що робить код більш декларативним та легким для тестування. Централізована структура Redux робить його більш підходящим для великих проектів, де важлива чіткість та контроль над станом. MobX може бути зручним для менших проектів або тих, де не потрібна така велика централізована контрольованість.

### 1.3.3 Аналіз відомих програмних продуктів

Money24 – веб-застосунок для підтримки роботи власних пунктів обміну валют по всій Україні, який має великий перелік доступних для обміну валют. Сайт містить інформацію про актуальні курси, калькулятор валют, можливість створювати запити на обмін у фізичному пункті без потреби реєстрації, а просто вказуючи номер телефону та ім'я.

Для порівняння курсової роботи з аналогом можна скористатись таблицею 1.3.3.1

Таблиця 1.3.3.1 – Порівняння з аналогом

Функціонал	Курсовий проект (CashFlow Exchange)	Money24	Пояснення
Можливість перегляду актуальних курсів онлайн	Так	Так	Обидва застосунки надають актуальні курси валют
Можливість розрахунку обміну валют в калькуляторі	Так	Так	Обидва застосунки мають калькулятор, який надає можливість обраховувати обмін за актуальним курсом
Можливість створення акаунта користувача	Так	Ні	Money24 не передбачає створення акаунтів, що несе за собою певні наслідки
Можливість створення запитів на обмін	Так	Так	Функціонал реалізовано по-різному, але він наявний в обох випадках. CashFlow Exchange потребує користувацького

			акаунту та верифікації пошти для цього, в той час як Money24 тільки номер телефона
Можливість переглядати історію обмінів	Так	Ні	Money24 не має можливості перегляду історії обмінів через відсутність особистих акаунтів
Можливість налаштування особистого акаунту	Так	Ні	Через відсутність можливості створення акаунтів, даний функціонал неможливий
Можливість відстежувати певні валюти та отримувати сповіщення про їх зміни	Так	Ні	Курсовий проект передбачає вибір порогового значення, при зміні курсу на яке, на пошту користувачу приходить сповіщення, Money24 не надає подібного функціоналу

#### 1.4 Аналіз вимог до програмного забезпечення

Головною функцією програмного забезпечення є створення запитів на обмін валют.

Програмне забезпечення повинно забезпечувати виконання наступних основних функцій:

Користувач повинен мати можливість зареєструватися використовуючи внутрішню систему реєстрації. В результаті натиску кнопки “Sign Up” буде створено нового користувача, якщо всі дані валідні. Після процедури реєстрації користувача переадресовує на “Home” сторінку.

Користувач повинен мати можливість авторизуватись використовуючи систему. Кнопка Sign In відповідає за підтвердження введених даних і у випадку їх правильного вводу, користувач успішно авторизується та переадресовується на сторінку “Home”.

Користувач повинен мати можливість переглядати актуальні курси валют на сторінці “Home”.

Користувач повинен мати можливість обирати з якої та в яку валюту відбувається конвертація в калькуляторі натискаючи на кнопку вибору валют.

Неавторизований користувач повинен мати можливість перейти на сторінку реєстрації або на сторінку авторизації, якщо має існуючий обліковий запис.

Авторизований користувач повинен мати можливість перейти в налаштування власного облікового запису шляхом натискання “Profile”.

Авторизований користувач повинен мати можливість перейти на сторінку створення угоди з обміну валют по кнопці “Start Exchange”.

Користувач повинен мати можливість змінювати персональні дані (“First name”, “Last name”, “Email”, “Password”).

Користувач повинен мати можливість обирати значення зміни цікавого йому курсу, при якому про цю зміну буде йти сповіщення на пошту користувачу.

Користувач повинен мати можливість обирати валюти між якими буде проводитись обмін шляхом.

Користувач повинен мати можливість переглядати історію своїх обмінів (валюти, між якими було проведено, дату та час, статус заявки).

Користувач повинен мати можливість змінювати статус заявки на обмін з “In review” або “Take away” на “Rejected”, якщо з якихось причин передумав проводити операцію.

Адміністратор системи повинен мати можливість переглядати усі створені запити та змінювати їх статус. Адміністратор може ставити 4 статуси (“In review” - статус ставиться по стандарту після створення та значить, що заявка очікує перевірки адміністратором. “Take away” - статус ставиться, коли адміністратор одобрих заявку та клієнту треба прийти на фізичний пункт та виконати обмін. “Rejected” - статус ставиться за різних причин, коли обмін не

може бути скоєним. “Done” - ставиться після успішного обміну вже на фізичному пункті.)

Більше функцій можна побачити на рисунку 1.4.1.

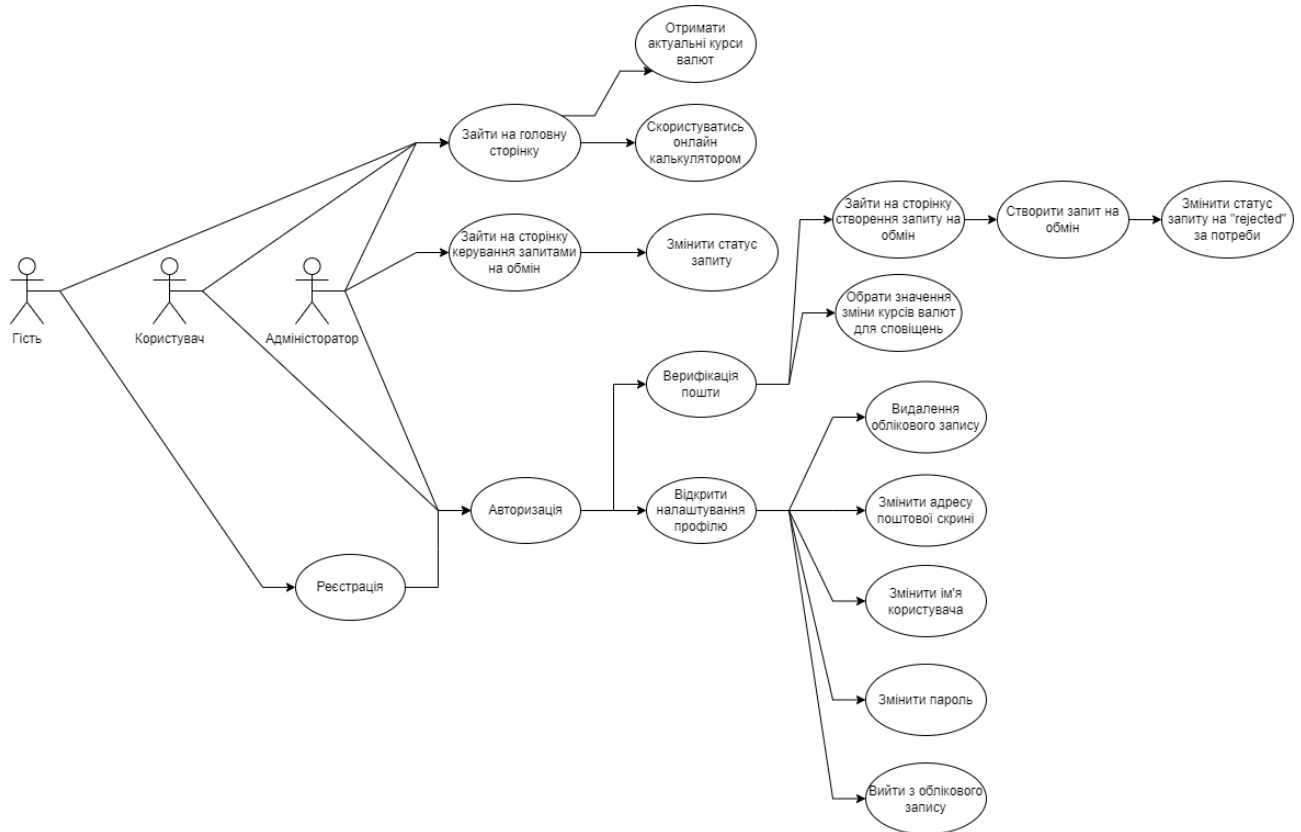


Рисунок 1.4.1 – Діаграма варіантів використання

В таблицях 1.4.1 – 1.4.11 наведені варіанти використання програмного забезпечення.

Таблиця 1.4.1 - Варіант використання UC-1

Use case name	Реєстрація користувача
Use case ID	UC-01
Goals	Реєстрація нового користувача в системі
Actors	Гість (неzareєстрований користувач)
Trigger	Користувач бажає зареєструватися
Pre-conditions	-
Flow of Events	Користувач переходить на сторінку реєстрації. В поля для реєстрації вводяться відповідні дані: ім'я, прізвище, пошта користувача, пароль в системі. Після заповнення даних користувач натискає кнопку реєстрації. Після цього користувача

	авторизує та перенаправляє на домашню сторінку
Extension	У випадку введення не коректних даних, з'являються помилки біля відповідних полів та форма потребує виправлення помилок, щоб бути підтвердженою
Post-Condition	Створення сторінки користувача, користувач авторизований, перехід на домашню сторінку

Таблиця 1.4.2 - Варіант використання UC-2

Use case name	Авторизація користувача
Use case ID	UC-02
Goals	Авторизація користувача в існуючий обліковий запис
Actors	Неавторизований користувач
Trigger	Користувач бажає авторизуватись
Pre-conditions	Обліковий запис користувача існує
Flow of Events	Користувач переходить на сторінку авторизації. В поля для авторизації вводяться відповідні дані: пошта, пароль. Після заповнення даних користувач натискає кнопку авторизації. Після цього користувача авторизує та перенаправляє на домашню сторінку
Extension	У випадку введення не коректних даних, з'являються помилки та форма потребує виправлення помилок, щоб бути підтвердженою.
Post-Condition	Користувач авторизований у власний обліковий запис, перехід на домашню сторінку

Таблиця 1.4.3 - Варіант використання UC-3

Use case name	Вихід з облікового запису
Use case ID	UC-03
Goals	Вийти з облікового запису
Actors	Авторизований користувач
Trigger	Користувач бажає вийти з облікового запису
Pre-conditions	Обліковий запис користувача існує, користувач авторизований
Flow of Events	Користувач переходить на сторінку "Profile". Користувач натискає кнопку "Sign Out". Після цього користувача викидає з облікового запису та перенаправляє на домашню сторінку
Extension	-
Post-Condition	Користувач не авторизований у власний обліковий запис,

	перехід на домашню сторінку
--	-----------------------------

Таблиця 1.4.4 - Варіант використання UC-4

Use case name	Розрахунок валюти
Use case ID	UC-04
Goals	Обрахувати кількість валюти при можливому обміні
Actors	Гість або авторизований користувач
Trigger	Гість бажає порахувати кількість грошей, які він отримає у випадку обміну за актуальним курсом.
Pre-conditions	Гість на сторінці “Home”
Flow of Events	Гість обирає валюту, з якої буде проводити конвертація, та валюту, в яку буде проводитись. В полі для вводу “Enter amount” ввести потрібну кількість грошей. Поле “You receive” виводить кількість грошей після конвертації
Extension	Користувач не може ввести не коректні дані, поле приймає тільки цифри без жодних інших символів
Post-Condition	Користувач отримав інформацію про кількість грошей, які може отримати при можливій конвертації

Таблиця 1.4.5 - Варіант використання UC-5

Use case name	Створення запиту на обмін
Use case ID	UC-05
Goals	Створити запит на обмін валюти
Actors	Авторизований користувач
Trigger	Користувач бажає обміняти валюту та створити запит для цього
Pre-conditions	Обліковий запис користувача існує, користувач авторизований
Flow of Events	Користувач вводить необхідну кількість грошей для обміну в калькулятор та обирає валюти для операції. Далі натискає кнопку підтвердження та запит створюється та додається в історію запитів
Extension	Користувач не може ввести не коректні дані в поля вводу валюти. Якщо користувач залишив поля пустими, запит не створиться
Post-Condition	Запит створено та додано в історію запитів



Таблиця 1.4.6 - Варіант використання UC-6

Use case name	Зміна статусу обміну користувачем
Use case ID	UC-06
Goals	Відмінити створений запит на обмін
Actors	Авторизований користувач
Trigger	Користувач бажає відмінити виконання свого запиту на обмін
Pre-conditions	Обліковий запис користувача існує, користувач авторизований, користувач має створений запит зі статусом “pending” або “take away”
Flow of Events	На сторінці “Start Exchange” в історії власних запитів користувач обирає потрібний запит, який хоче відмінити. Запит має статус “pending” або “take away”, користувач змінює статус на “rejected”
Extension	Користувач може змінити статус тільки з “pending” або “take away” і тільки на “rejected”, інші варіанти не приймаються
Post-Condition	Статус запиту змінено

Таблиця 1.4.7 - Варіант використання UC-7

Use case name	Зміна статусу обміну адміністратором
Use case ID	UC-07
Goals	Змінити створений користувачем запит на обмін
Actors	Адміністратор застосунку
Trigger	Адміністратор бажає розгледіти заявки на обмін
Pre-conditions	Обліковий запис користувача існує, користувач авторизований, користувач має створений запит зі статусом “pending” або “take away”
Flow of Events	На сторінці “Exchange Requests” в списку всіх запитів адміністратор обирає потрібний запит, який хоче розгледіти. Запит має статус “pending” або “take away”, адміністратор змінює на “completed”, “rejected”, “take away” в залежності від ситуації
Extension	Адміністратор не може міняти значення, “completed” та “rejected”, оскільки це вже відпрацьовані запити
Post-Condition	Статус запиту змінено

Таблиця 1.4.8 - Варіант використання UC-8

Use case name	Зміна імені користувача
Use case ID	UC-08
Goals	Зміна персональних даних облікового запису користувача
Actors	Авторизований користувач
Trigger	Користувач бажає змінити персональні дані (username)
Pre-conditions	Обліковий запис користувача існує, користувач авторизований
Flow of Events	Користувач переходить на сторінку “Profile”, де змінює потрібні йому дані (username). Натискає кнопку підтвердження
Extension	У випадку введення не коректних даних, з’являються помилки та форма потребує виправлення помилок, щоб бути підтвердженою.
Post-Condition	Персональні дані змінені

Таблиця 1.4.9 - Варіант використання UC-9

Use case name	Зміна пошти
Use case ID	UC-09
Goals	Зміна даних облікового запису користувача (пошта)
Actors	Авторизований користувач
Trigger	Користувач бажає змінити пошту
Pre-conditions	Авторизований користувач
Flow of Events	Користувач переходить на сторінку “Profile”, де змінює потрібне йому поле (пошта). Для зміни пошти потрібно ввести дійсний пароль для підтвердження зміни. Користувач натискає кнопку підтвердження. Після чого його викидає з акаунта та переадресовує на сторінку авторизації. Нову пошту потрібно верифікувати, щоб авторизуватись з неї.
Extension	У випадку введення не коректних даних, з’являються помилки та форма потребує виправлення помилок, щоб бути підтвердженою.
Post-Condition	Користувач не авторизований у власний обліковий запис, перехід на сторінку авторизації, нова пошта підтверджена

Таблиця 1.4.10 - Варіант використання UC-10

Use case name	Зміна пароллю
Use case ID	UC-10
Goals	Зміна чутливої інформації (пароллю)
Actors	Авторизований користувач
Trigger	Користувач бажає змінити пароль
Pre-conditions	Авторизований користувач
Flow of Events	Користувач переходить на сторінку “Profile”, де змінює потрібне йому поле (пароль). Для зміни пароллю потрібно ввести дійсний пароль для підтвердження зміни та новий пароль. Користувач натискає кнопку підтвердження
Extension	У випадку введення не коректних даних, з’являються помилки та форма потребує виправлення помилок, щоб бути підтвердженою
Post-Condition	Користувач авторизований у власний обліковий запис з новим паролем, перехід на сторінку авторизації

Таблиця 1.4.11 - Варіант використання UC-11

Use case name	Зміна значення валют для сповіщення
Use case ID	UC-11
Goals	Зміна мінімальних значень валют, при зміні на які буде надсилатись сповіщення при оновленні курсу
Actors	Авторизований користувач
Trigger	Користувач бажає отримувати сповіщення про зміну курсів
Pre-conditions	Користувач авторизований, має верифіковану пошту.
Flow of Events	Користувач переходить на сторінку “Profile”, де значення біля потрібних йому валют. Якщо при оновленні курсу, він зміниться на таке значення чи більше по модулю, користувач отримає сповіщення на верифіковану пошту. Користувач натискає кнопку підтвердження
Extension	У випадку введення не коректних даних (коректні від 0 до 10), з’являються помилки та форма потребує виправлення помилок, щоб бути підтвердженою. Якщо пошта користувача не верифікована, він не матиме можливість ставити якісь значення
Post-Condition	При оновленні курсу на пошту прийшло сповіщення відповідно до значень, які вказані в полях.

### 1.4.1 Розроблення функціональних вимог

Програмне забезпечення розділене на модулі. Кожен модуль має свій певний набір функцій. На рисунку 1.4.1.1 наведено загальну модель вимог, а в таблицях 1.4.1.1 – 1.4.1.17 наведений опис функціональних вимог до програмного забезпечення. Матрицю трасування вимог можна побачити на рисунку 1.4.1.2.

Опис	Назва	Пріоритет	Ризик
1. Реєстрація	FR-1	2	Високий
2. Авторизація	FR-2	2	Високий
3. Перегляд головної сторінки	FR-3	1	Високий
3.1. Перегляд актуальних курсів валют	FR-4		
3.2. Обрахунок обміну в калькуляторі валют	FR-5		
4. Створення запиту на обмін	FR-6	1	Середній
4.1. Обрахунок обміну в калькуляторі валют	FR-5		
4.2. Перегляд історії обмінів	FR-7		
4.3. Відміна запиту на обмін	FR-8		
5. Керування запитами на обмін з боку адміністратора	FR-9	2	Високий
5.1. Перегляд усіх створених запитів.	FR-10		
5.2. Зміна статусу запиту.	FR-11		
6. Зміна персональних даних	FR-12	2	Високий
6.1 Зміна імені та прізвища	FR-13		
6.2 Зміна пошти	FR-14		
6.3 Зміна паролю	FR-15		
7. Вибір порогу зміни валют для сповіщення	FR-16	3	Низький

8. Вихід з облікового запису	FR-17	2	Низький
------------------------------	-------	---	---------

Рисунок 1.4.1.1 – Модель вимог у загальному вигляді

Таблиця 1.4.1.1 – Функціональна вимога FR-1

Назва	Реєстрація
Опис	Система повинна надавати можливість реєстрації користувачеві шляхом введення імені, прізвища, пошти, пароллю.

Таблиця 1.4.1.2 – Функціональна вимога FR-2

Назва	Авторизація
Опис	Система повинна надавати можливість авторизуватись користувачеві шляхом введення пошти, пароллю.

Таблиця 1.4.1.3 – Функціональна вимога FR-3

Назва	Перегляд головної сторінки
Опис	Система повинна надавати можливість зображати інформацію головної сторінки усім користувачам, які заходять на сайт.

Таблиця 1.4.1.4 – Функціональна вимога FR-4

Назва	Перегляд актуальних курсів валют
Опис	Система повинна надавати можливість відображати актуальні курси валют на головній сторінці.

Таблиця 1.4.1.5 – Функціональна вимога FR-5

Назва	Обрахунок обміну в калькуляторі валют
Опис	Система повинна надавати можливість розраховувати конвертацію валюти згідно актуального курсу.

Таблиця 1.4.1.6 – Функціональна вимога FR-6

Назва	Створення запиту на обмін
Опис	Система повинна надавати можливість користувачам створювати

	запит на обмін.
--	-----------------

Таблиця 1.4.1.7 – Функціональна вимога FR-7

Назва	Перегляд історії обмінів
Опис	Система повинна надавати можливість користувачам переглядати історію власних обмінів.

Таблиця 1.4.1.8 – Функціональна вимога FR-8

Назва	Відміна запиту на обмін
Опис	Система повинна надавати можливість користувачам відмінити створені запити на обмін.

Таблиця 1.4.1.9 – Функціональна вимога FR-9

Назва	Керування запитами на обмін з боку адміністратора
Опис	Система повинна надавати можливість адміністраторам керувати усіма створеними запитами.

Таблиця 1.4.1.10 – Функціональна вимога FR-10

Назва	Перегляд усіх створених запитів
Опис	Система повинна надавати можливість адміністраторам переглядати усі створені запити.

Таблиця 1.4.1.11 – Функціональна вимога FR-11

Назва	Зміна статусу запиту
Опис	Система повинна надавати можливість адміністраторам змінювати статус запитів.

Таблиця 1.4.1.12 – Функціональна вимога FR-12

Назва	Зміна персональних даних
-------	--------------------------

Опис	Система повинна надавати можливість користувачам змінювати персональні дані.
------	--

Таблиця 1.4.1.13 – Функціональна вимога FR-13

Назва	Зміна імені та прізвища
Опис	Система повинна надавати можливість користувачам змінювати ім'я та прізвище.

Таблиця 1.4.1.14 – Функціональна вимога FR-14

Назва	Зміна пошти
Опис	Система повинна надавати можливість користувачам змінювати пошту.

Таблиця 1.4.1.15 – Функціональна вимога FR-15

Назва	Зміна паролю
Опис	Система повинна надавати можливість користувачам змінювати пароль.

Таблиця 1.4.1.16 – Функціональна вимога FR-16

Назва	Вибір порогу зміни валют для сповіщення
Опис	Система повинна надавати можливість користувачам обирати порогові значення для валют, про зміну яких вони хочуть отримувати сповіщення.

Таблиця 1.4.1.17 – Функціональна вимога FR-17

Назва	Вихід з облікового запису
Опис	Система повинна надавати можливість користувачам виходити з облікового запису.

	FR-1	FR-2	FR-3	FR-4	FR-5	FR-6	FR-7	FR-8	FR-9	FR-10	FR-11	FR-12	FR-13	FR-14	FR-15	FR-16	FR-17
UC-1	+																
UC-2		+															
UC-3																	+
UC-4			+	+	+												
UC-5					+	+											
UC-6							+	+									
UC-7									+	+	+						
UC-8												+	+				
UC-9												+		+			
UC-10												+			+		
UC-11																+	

Рисунок 1.4.1.2 – Матриця трасування вимог

## 1.4.2 Розроблення нефункціональних вимог

Нефункціональні вимоги є важливим аспектом продукту, які визначають його якість та характеристики.

Таблиця 1.4.2.1 – Нефункціональні вимоги

Номер	Назва	Опис
NFR-1	Сумісність	Забезпечення сумісності із браузерами Google Chrome (Версія 120.0.6099.130), Opera (Версія Opera 95), Edge (Версія 119.0.2151.58), Firefox (Верс 121.0) та пристроями для зручного використання користувачами.
NFR-2	Надійність	Система повинна бути стійкою до помилок та забезпечувати контроль введення інформації та захист від некоректних дій користувача. При виникненні помилок, система повинна надавати зрозумілі та інформативні повідомлення користувачеві.
NFR-3	Безпека	Система повинна гарантувати конфіденційність інформації та захищати від несанкціонованого доступу під час операцій.



## 1.5 Постановка задачі

В результаті розробки застосунку стоять цілі отримати гарно оптимізований продукт з мінімалістичним та інтуїтивним інтерфейсом, який забезпечить найкращий досвід користування. Веб-застосунок повинен оптимізувати процес комунікації між потенційним клієнтом та пунктом обміну. Основна задача застосунку – дати можливість людині передивлятись актуальні курси онлайн, створювати запити на обміни потрібних валют, відстежувати зміни потрібних курсів.

Створення запитів є головним завданням ПЗ, для виконання даного завдання першочергово потрібно отримувати актуальні курси валют, в ролі джерела даних буде використовуватись зовнішня API, з якої щоденно автоматично будуть діставатись актуальні курси. Далі користувач в калькуляторі обраховує потрібні значення для обміну та створює запит, який з'являється в історії запитів з урахуванням комісії за обмін. Однією з ключових функцій є отримання сповіщень при зміні курсів, це значно поліпшує користувацький досвід порівняно з аналогами, де немає можливості відстежувати зміну окремих курсів при заданих значеннях зміни. В даному проекті це можливо завдяки наявності власних облікових записів користувачів, де вони можуть виставляти значення, при зміні курсу на які, на верифіковану поштову скриньку прийде сповіщення про зміну потрібних валют. Весь цей функціонал допомагає користувачу першим отримувати важливу йому інформацію про зміну курсів валют та створювати запити з можливістю онлайн відстеження їх статусу.

Проект також має виконувати нефункціональні вимоги: сумісність, надійність, безпека, зручність інтерфейсу. Все це потрібно для забезпечення максимального задоволення користувача від процесу використання сайту.

## Висновки до розділу

В даному розділі було виконано детальний аналіз предметної області, визначено мету та ціль для розробки додатку, обрано технології, які найкраще відповідають вимогам проекту та оптимізують час виконання завдання. Проведено аналіз аналогічних додатків та їх особливостей, звідси було отримано причини для розробки власного застосунку. Проведено детальний аналіз допоміжних технологій та їх аналогів, де описано, чому було обрано ту чи іншу технологію. Проведено детальне порівняння з локально відомим застосунок Money24 та виявлено його недоліки.

Чітко визначено функціональні вимоги з допоміжними діаграмами та таблицями, розписано детальний процес виконання вимог. Також визначено нефункціональні вимоги, які гарно впливають на стабільність роботи застосунку та користувацький досвід.

## 2 МОДЕЛЮВАННЯ ТА КОНСТРУЮВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

### 2.1 Моделювання та аналіз програмного забезпечення

Для моделювання бізнес-процесів було вибрано BPMN <sup>[8]</sup>, який є визнаним стандартом у цій області. BPMN надає уніфікований інструмент для загального розуміння, аналізу та оптимізації бізнес-процесів у корпоративному середовищі. Використання цього стандарту дозволяє застосовувати символи та правила для створення графічних представлень процесів, полегшуючи взаємодію між бізнес-аналітиками та розробниками програмного забезпечення.

Переваги використання BPMN: визнаний як міжнародний стандарт для моделювання бізнес-процесів, що забезпечує єдиний набір термінів та концепцій. BPMN використовує графічні символи та правила, що полегшує створення інтуїтивно зрозумілих графічних представлень бізнес-процесів. BPMN є універсальним інструментом, придатним як для стратегічного моделювання, так і для деталей оперативних процесів. Застосування BPMN дозволяє легше управляти і співпрацювати між бізнес-аналітиками та розробниками. Це сприяє створенню спільного розуміння між учасниками проекту та узгодженню різних команд.

Отже, висунемо головні бізнес-процеси:

- реєстрація;
- авторизація;
- перегляд курсів валют та користування калькулятором валют;
- зміна персональних даних;
- створення запитів на обмін валют;
- керування запитами адміністратором;
- розсилання сповіщень про зміну курсу;

Для опису бізнес процесу програмного забезпечення використовуються BPMN моделі (рисунок 2.1.1 та рисунок 2.1.2).

Опис послідовності реєстрації користувача:

- користувач переходить на сторінку реєстрації;
- користувач заповнює поля реєстрації;
- якщо введені поля не відповідають шаблону заповнення на клієнтській стороні, відповідні поля підсвічуються помилкою.

Опис послідовності авторизації користувача:

- користувач переходить на сторінку авторизації;
- користувач заповнює поля реєстрації;
- якщо введені поля не відповідають шаблону заповнення на клієнтській стороні, відповідні поля підсвічуються помилкою.

Опис послідовності перегляду курсів валют та користування калькулятором валют:

- користувач переходить на головну сторінку;
- користувач отримує інформацію про актуальні курси з таблички курсів;
- користувач обирає потрібні валюти та вводить кількість грошей;
- система вираховує та показує кількість грошей після обміну.

Опис послідовності зміни персональних даних:

- користувач переходить на сторінку профіля;
- користувач змінює дані в потрібних йому полях;
- користувач бачить змінені дані;
- якщо введені поля не відповідають шаблону заповнення на клієнтській стороні, відповідні поля підсвічуються помилкою.

Опис послідовності створення запитів на обмін валют:

- користувач переходить на сторінку початку обміну;
- користувач обирає валюти для обміну та вводить кількість грошей;

- користувач бачить кількість грошей після обміну та підтверджує обмін;
- користувач бачить створений запит в історії своїх запитів.

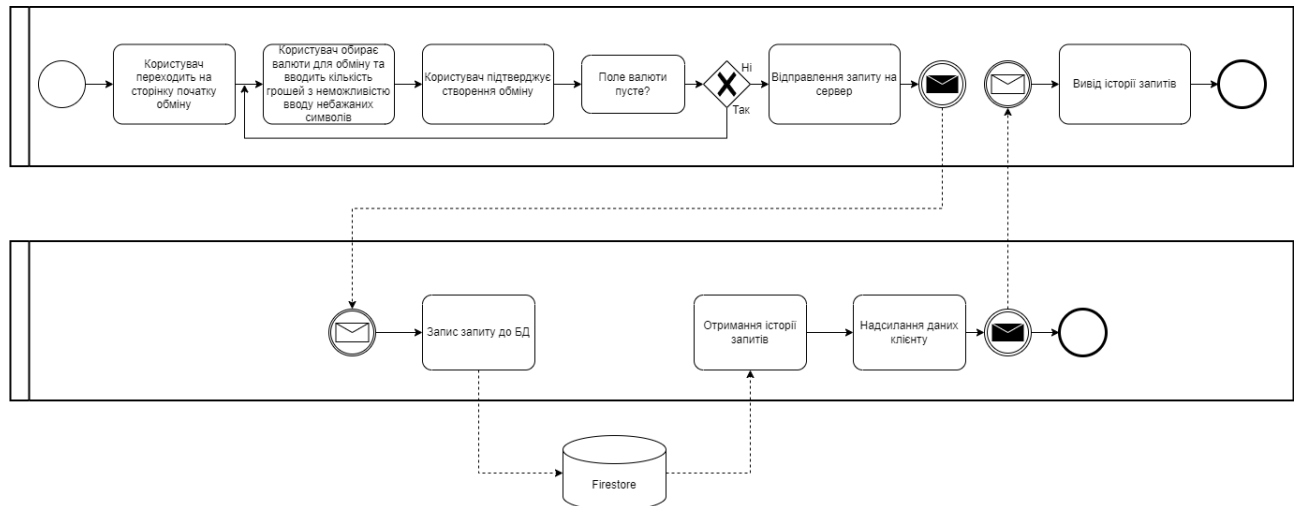


Рисунок 2.1.1 – Діаграма бізнес-процесу створення запитів на обмін.

Опис послідовності керування запитами адміністратором:

- адміністратор переходить на сторінку запитів на обмін;
- адміністратор бачить усі запити з їх детальною інформацією та станами;
- адміністратор змінює статус заявки на потрібний.
- адміністратор бачить заявку з оновленим статусом;

Опис послідовності розсилання сповіщень про зміну курсу:

- користувач переходить на сторінку профіля;
- користувач змінює мінімальне значення при якому прийде повідомлення на пошту по певним валютам;
- користувач чекає на оновлення курсу в системі;
- при оновленні курсу користувач отримує сповіщення, якщо обрані валюти змінились на заданий курс.

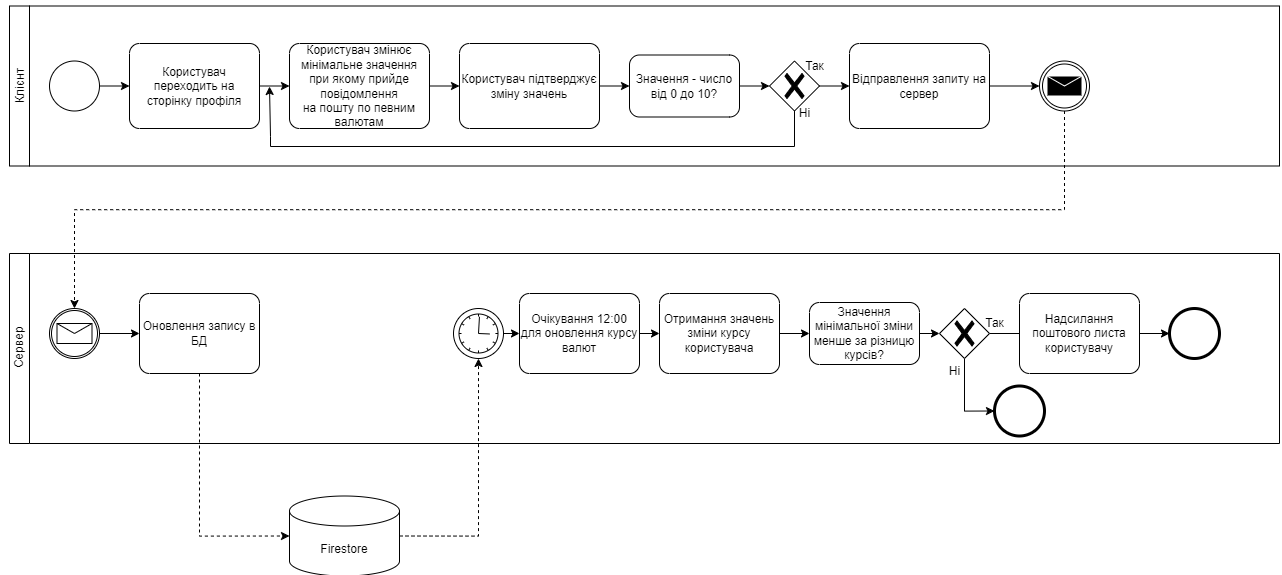


Рисунок 2.1.2 – Діаграма бізнес-процесу розсилання сповіщень про зміну курсу.

## 2.2 Архітектура програмного забезпечення

Клієнт-серверна архітектура<sup>[9]</sup> — це підхід до побудови програмних систем, при якому функціонал системи розподіляється між клієнтською стороною та серверною. Клієнтський сторона в свою чергу відповідає за користувацький інтерфейс, а серверний компонент виконує обробку даних та надає потрібні ресурси. Цей підхід сприяє зменшенню завантаження на клієнтські пристрої та централізованому управлінню ресурсами на сервері.

В нашому випадку клієнт-серверна архітектура представлена у вигляді клієнтської частини, яка використовує такі технології, як React, який є ефективною бібліотекою для побудови користувацьких інтерфейсів, його роль полягає в наданні інтерфейсу для взаємодії з додатком та відправленні запитів на серверну частину для отримання даних для оновлення інтерфейсу, а сама серверна частина представлена у вигляді Firebase сервісів та зовнішньої API для отримання курсів валют, тому даний веб-застосунок можна ще класифікувати як мікросервісний. Наприклад, клієнтська частина потребує відображення актуальних курсів валют, для чого посилається запит до зовнішньої API, яка вже повертає потрібні дані. Також як приклад, клієнт хоче створити запит на обмін валют, для цього після створення запиту, його дані мають записатись в БД, знову ж доводиться звертатись до серверу.

З переваг даного підходу можна відмітити можливість централізовано керувати та забезпечувати консистентність даних, також важливим пунктом є зниження навантаження на клієнт, тому клієнтська частина може фокусуватись на інтерфейсі та користувацькому досвіді.

Представлення архітектури даного проекту можна побачити на рис. 2.2.1.

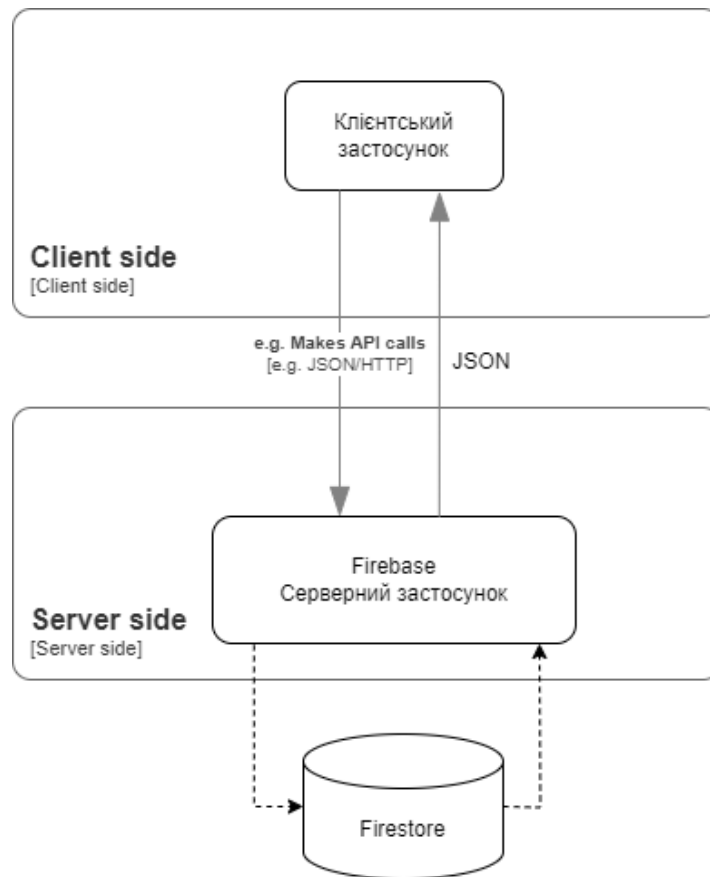


Рисунок 2.2.1 – Загальна архітектура веб-застосунку

Мікросервісна архітектура<sup>[10]</sup> – це підхід до розробки програмного забезпечення, при якому програмна система розбивається на невеликі і незалежні компоненти під назвою мікросервіси. Кожен мікросервіс виконує певний функціонал та взаємодіє з іншими мікросервісами через API. Такий підхід сприяє гнучкості, масштабованості та легкості розвитку програмних систем.

В даному застосунку мікросервісна архітектура представлена у вигляді незалежних сервісів всередині Firebase (Firestore, Cloud Functions, Authentication) та зовнішньої API у вигляді мікросервісу. Firestore використовується для зберігання даних у вигляді документо-орієнтованої БД,

Cloud Functions створює можливість створювати власні функції на Node.js<sup>[11]</sup>, завдяки яким реалізовано процес збору даних з зовнішньої API та розсилки сповіщень на пошту. Authentication використовується як сервіс для авторизації та збереження даних авторизації, які незалежні від даних користувача в Firestore. Клієнтська частина звертається окремо до різних мікросервісів за потреб, якщо потрібно оновити курси валют, потрібно звертатись до зовнішньої API, якщо потрібно авторизуватись, то потрібно звертатись до мікросервісу аутентифікації.

З переваг даної архітектури є гнучкість та незалежність, де кожен мікросервіс розглядається як окремий компонент, який працює незалежно від інших мікросервісів, також це спрощує розвиток та розгортання додатку через декомпозицію.

Бізнес-логіка реалізована в серверній частині, його компоненти можна побачити на рис. 2.2.2.

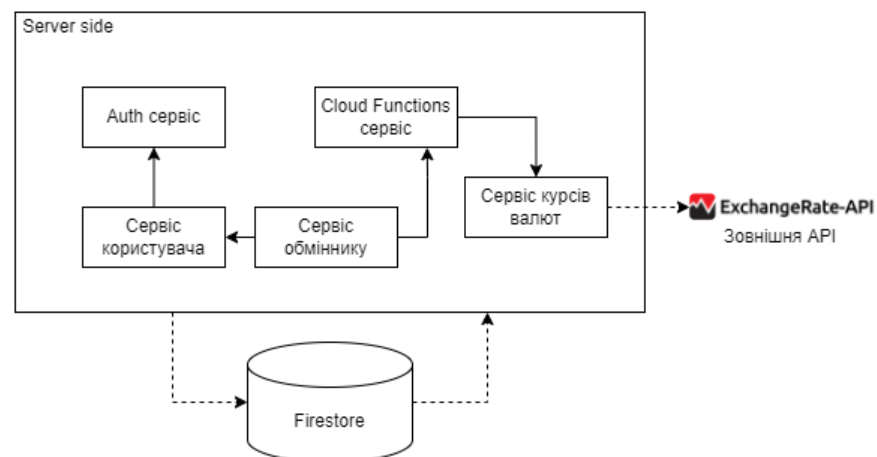


Рисунок 2.2.2 – діаграма бізнес логіки застосунку.

### 2.3 Конструювання програмного забезпечення

Веб-застосунок, який реалізується у межах даної курсової роботи, не вимагає впровадження конкретних алгоритмів чи специфічних технічних рішень. Було обрано NoSQL<sup>[12]</sup> базу даних, оскільки дані не потребують складної структури, вона реалізує простий дизайн схеми БД, значно спрощує горизонтальне масштабування на кластери машин і має тонкий контроль над доступністю, через відмінну структуру даних потрібні операції виконуються



швидше, ніж було б на реляційній БД. База даних призначена для зберігання користувачів, їх запитів на обмін та актуальних курсів валют.

Опис таблиць бази даних наведено у таблицях 2.3.1 – 2.3.3. ER модель бази даних наведена на рисунку 2.3.1.

Таблиця 2.3.1 – Опис колекції users

Таблиця	Назва поля	Тип даних	Опис
users	id	Serial	id документу, ідентифікаційний номер користувача
	firstName	String	ім'я користувача
	lastName	String	прізвище користувача
	email	String	електрона адреса користувача
	isAdmin	Boolean	наявність особливих прав у користувача
	currencyDiff	Object	об'єкт, який зберігає інформацію про зміну валют для триггеру сповіщень користувачу
	currencyDiff. EUR	Number	значення зміни курсу євро для приходу сповіщення на пошту
	currencyDiff. GBP	Number	значення зміни курсу фунта для приходу сповіщення на пошту
	currencyDiff. PLN	Number	значення зміни курсу злотих для приходу сповіщення на пошту
	currencyDiff. USD	Number	значення зміни курсу долара для приходу сповіщення на пошту

Таблиця 2.3.2 – Опис колекції rates

Таблиця	Назва поля	Тип даних	Опис
rates	EUR	Number	значення курсу євро
	GBP	Number	значення курсу фунта
	PLN	Number	значення курсу злотих
	USD	Number	значення курсу американського долара

Таблиця 2.3.3 – Опис колекції requests

Таблиця	Назва поля	Тип даних	Опис
requests	id	Serial	ідентифікаційний номер запиту, id документа
	userId	Serial	ідентифікаційний номер користувача, якому належить запит на обмін
	timestamp	Object	час створення запиту на обмін
	amount	Number	кількість валюти яку віддає користувач
	convertedAmount	Number	кількість валюти яку отримує користувач
	sourceCurrency	String	валюта з якої відбувається конвертація

### Продовження таблиці 2.3.3

	targetCurrency	String	валюта в яку відбувається конвертація
	status	String	статус обробки запиту на обмін

На рисунку 2.3.1 зображено модель NoSQL БД:

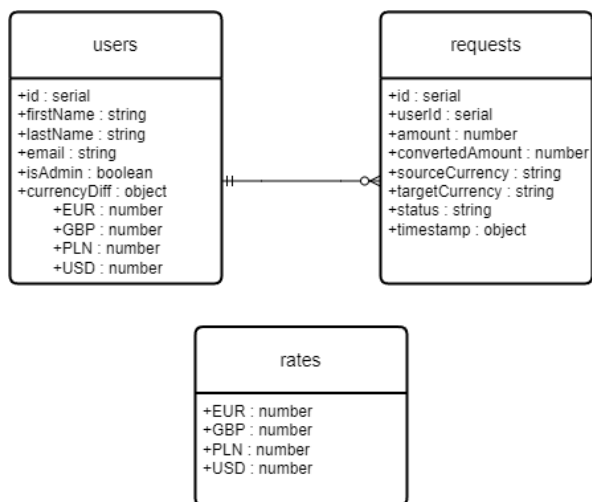


Рисунок 2.3.1 – модель нереляційної БД.

Опис утиліт, бібліотек та іншого стороннього програмного забезпечення, що використовується у розробці наведено в таблиці 2.3.4.

Таблиця 2.3.4 – Опис утиліт

№ п/п	Назва утиліти	Опис застосування
1	Visual Studio Code	Головне середовище розробки програмного забезпечення.
2	React	JavaScript бібліотека для створення інтерфейсів користувача.
3	Firebase	Платформа розробки мобільних та веб застосунків, що надає доступ до хмарного сховища.

#### Продовження таблиці 2.3.4

4	Git	Система контролю версій для зручного управління змінами проекту.
5	Redux	JavaScript бібліотека для керування станом застосунку.

#### 2.4 Аналіз безпеки даних

Аналіз безпеки включає в себе оцінку вразливостей ПЗ та застосування основних заходів безпеки ПЗ для запобігання подібних атак:

- Атака брутфорс<sup>[13]</sup>: спроба злому пароля шляхом перебору. Завдяки вбудованим заходам безпеки Firebase, спроба атаки брутфорс запобігається обмеженням кількості невдалих спроб авторизації.
- Міжсайтовий скриптинг (XSS)<sup>[14]</sup> – внесення зловмисного коду на веб-сторінки користувачів. Firebase використовує власні механізми для захисту від міжсайтового скриптингу. Це включає в себе валідацію та екранування введених даних перед їх збереженням та обробкою.
- Шифрування даних в Firestore забезпечує безпеку особистих даних користувача.
- Механізми видалення власного облікового запису та верифікації пошти забезпечують вимоги безпеки персональної інформації.

#### Висновки до розділу

У даному розділі було проведено детальний аналіз бізнес-процесів, використано BPMN моделі для моделювання бізнес-процесів. Для двох складних процесів було імплементовано модель.

Було обґрунтовано вибір паттерну архітектури для розробки ПЗ, перераховано переваги обраних фігур та наведено чіткі схеми архітектури. Застосунок має клієнт-серверну мікросервісну архітектуру, оскільки функціонал в системі має розділятися на клієнтську та сервісну частину для делегування функціоналу та покращення оптимізації, також Firebase .

Використання NoSQL Firestore надає гнучкість та масштабованість у зберіганні та отриманні даних. Модель даних ретельно розроблена, враховуючи специфіку додатку та його потреби.

В детальному аналізі безпеки пояснено способи запобігання загроз брутфорс, XSS, процес шифрування даних в БД та обробки конфіденційної персональної інформації. Для їх запобігання впроваджені відповідні заходи безпеки, включаючи використання тимчасового блокування облікового запису при багатьох невдалих спробах авторизації, валідацію та екранування даних.

### 3 АНАЛІЗ ЯКОСТІ ТА ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

#### 3.1 Аналіз якості ПЗ

Аналіз якості програмного забезпечення є ключовим етапом у життєвому циклі розробки та експлуатації. Це процес, що визначає, наскільки програмне забезпечення відповідає вимогам, стандартам та очікуванням користувачів. В якості інструменту статичного аналізу коду було під'єднано ESLint для виявлення проблемних шаблонів. Також було проведено аналіз коду за допомогою онлайн аналізатора Codacy<sup>[15]</sup>(рисунок 3.2.1).

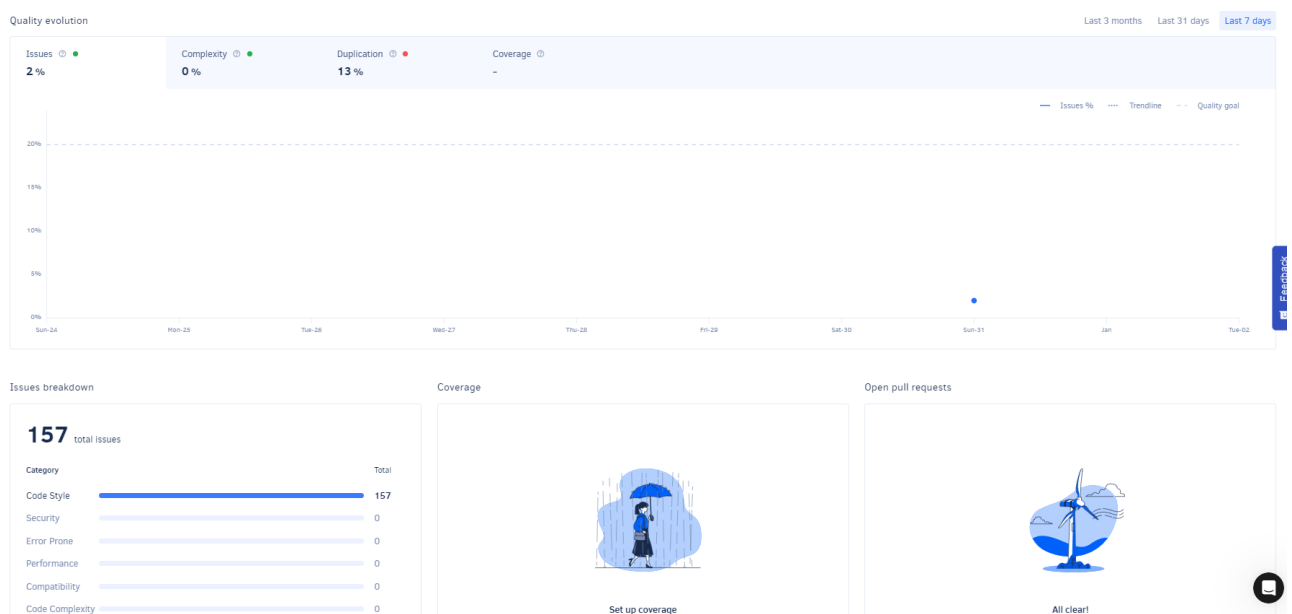


Рисунок 3.2.1 – результати онлайн аналізу коду

З виявлених проблем було піймано тільки мінорні проблеми, пов'язані зі стилем коду, які ніяк не впливають на ефективність та надійність коду.

Подальші процеси тестування функціональних вимог застосунку продемонструють якісну роботу необхідного функціоналу.

#### 3.2 Опис процесів тестування

Тестування є важливим аспектом в налагодженні роботи ПЗ, оскільки воно дозволяє оцінити якість готового продукту, перевірити виконання зазначених вимог та знайти можливі вразливості, що дає можливість виявити

серйозні проблеми перед випуском застосунку в маси, також дозволяє ефективно використовувати часові та людські ресурси, маючи повністю налагоджений процес тестування.

Було виконане мануальне тестування програмного забезпечення, опис відповідних тестів наведено у таблицях 3.2.1 – 3.2.10.

Таблиця 3.2.1 – Тест 1.1

Тест	Реєстрація користувача
Модуль	Реєстрація користувача
Номер тесту	1.1
Початковий стан системи	Користувач знаходиться на сторінці реєстрації
Вхідні данні	Ім'я облікового запису, електронна пошта, пароль
Опис проведення тесту	У відповідні поля вводяться: електронна пошта в правильному форматі, яка до цього не була зареєстрована в системі, пароль від 8 до 16 символів, який може містити тільки цифри, літери та спеціальні символи.
Очікуваний результат	Реєстрація проходить успішно, користувач додається у систему, автоматично авторизується і перенаправляється на домашню сторінку.
Фактичний результат	Реєстрація проходить успішно, користувач додається у систему, автоматично авторизується і перенаправляється на домашню сторінку.

Таблиця 3.2.2 – Тест 2.1

Тест	Авторизація користувача
Модуль	Авторизація користувача
Номер тесту	2.1
Початковий стан	Користувач знаходиться на сторінці авторизації

системи	
Вхідні данні	Електронна пошта, пароль
Опис проведення тесту	Користувач, який має активний обліковий запис, вводить свою електронну пошту, до якої прив'язаний обліковий запис, та пароль від даного облікового запису
Очікуваний результат	Авторизація проходить успішно, користувач авторизується в системі і перенаправляється на домашню сторінку.
Фактичний результат	Авторизація проходить успішно, користувач авторизується в системі і перенаправляється на домашню сторінку.

Таблиця 3.2.3 – Тест 3.1

Тест	Розрахунок валюти в калькуляторі
Модуль	Калькулятор валют
Номер тесту	3.1
Початковий стан системи	Користувач знаходиться на домашній сторінці
Вхідні данні	Валюти для конвертації, значення валюти
Опис проведення тесту	Користувач обирає валюти для конвертації та вводить коректне значення валюти в поле “Enter amount”
Очікуваний результат	Конвертація проходить успішно, користувач отримує результати конвертації.
Фактичний результат	Конвертація проходить успішно, користувач отримує результати конвертації.

Таблиця 3.2.4 – Тест 4.1

Тест	Створення запитну на обмін
------	----------------------------



Модуль	Запит на обмін
Номер тесту	4.1
Початковий стан системи	Користувач знаходиться на сторінці створення запиту
Вхідні данні	Валюти для конвертації, значення валюти
Опис проведення тесту	Користувач обирає валюти для конвертації та вводить коректне значення валюти в поле “Enter amount”. Натискає кнопку підтвердження
Очікуваний результат	Запит успішно створюється та відображається в історії запитів.
Фактичний результат	Запит успішно створюється та відображається в історії запитів.

Таблиця 3.2.5 – Тест 4.2

Тест	Відміна запиту на обмін
Модуль	Запит на обмін
Номер тесту	4.2
Початковий стан системи	Користувач знаходиться на сторінці створення запиту
Вхідні данні	Новий статус запиту
Опис проведення тесту	Користувач обирає потрібний запит та в полі статусу змінює статус з “pending” на “rejected”.
Очікуваний результат	Статус запиту успішно змінюється.
Фактичний результат	Статус запиту успішно змінюється.

Таблиця 3.2.6 – Тест 4.3

Тест	Зміна статусу запиту на обмін адміністратором
Модуль	Запит на обмін
Номер тесту	4.3
Початковий стан системи	Адміністратор знаходиться на сторінці всіх заявок на обмін
Вхідні данні	Новий статус запиту
Опис проведення тесту	Адміністратор обирає потрібний запит та в полі статусу змінює статус з “take away” на “completed”
Очікуваний результат	Статус запиту успішно змінюється.
Фактичний результат	Статус запиту успішно змінюється.

Таблиця 3.2.7 – Тест 5.1

Тест	Зміна мінімального значення зміни курсу, при зміні на яке користувач отримує сповіщення на пошту
Модуль	Зміна інформації профіля
Номер тесту	5.1
Початковий стан системи	Користувач знаходиться на сторінці профіля
Вхідні данні	Значення зміни курсу
Опис проведення тесту	Користувач вводить порогові значення зміни курсу, при зміні на які приходить лист на пошту, для USD = 0, всі інші поля залишає пустими.
Очікуваний результат	При оновленні курсу на пошту прийшов лист зі сповіщення про зміну курсу долара.
Фактичний результат	При оновленні курсу на пошту прийшов лист зі сповіщення про зміну курсу долара.

Таблиця 3.2.8 – Тест 5.2

Тест	Зміна імені облікового запису в інформації профіля
Модуль	Зміна інформації профіля
Номер тесту	5.2
Початковий стан системи	Користувач знаходиться на сторінці профіля
Вхідні данні	Username
Опис проведення тесту	Користувач вводить новий username замість старих. Нові дані починаються з великою літери, містять тільки літери без пробілів.
Очікуваний результат	Ім'я та прізвище користувача успішно змінені.
Фактичний результат	Ім'я та прізвище користувача успішно змінені.

Таблиця 3.2.9 – Тест 5.3

Тест	Зміна адреси пошти
Модуль	Зміна інформації профіля
Номер тесту	5.3
Початковий стан системи	Користувач знаходиться на сторінці профіля
Вхідні данні	Поштова адреса, пароль
Опис проведення тесту	Користувач вводить нову поштову адресу замість старої та актуальний пароль для підтвердження зміни. Підтверджує нову поштову адресу по посиланню на новій пошті. Авторизується з новою поштою
Очікуваний результат	Користувача успішно авторизовано з новою поштою
Фактичний	Користувача успішно авторизовано з новою поштою

результат	
-----------	--

Таблиця 3.2.10 – Тест 6.1

Тест	Блокування доступу до деяких можливостей
Модуль	Верифікація пошти
Номер тесту	6.1
Початковий стан системи	Користувач знаходиться на домашній сторінці
Вхідні данні	Пошта користувача не верифікована
Опис проведення тесту	Користувач намагається перейти на сторінку створення запитів.
Очікуваний результат	Система не пускає користувача на сторінку створення запитів та виводить помилку.
Фактичний результат	Система не пускає користувача на сторінку створення запитів та виводить помилку.

### 3.3 Опис контрольного прикладу

В даному розділі будемо відтворювати взаємодію користувача з веб-застосунком. Тестування було проведено на різних комп'ютерах, які відповідають рекомендованим вимогам в браузері Google Chrome, Firefox, Opera, Edge останніх версій.

Відтворимо сценарій, де користувач бажає зареєструватись або авторизуватись в системі. Введемо коректні дані та подивимось, як програма відпрацює.

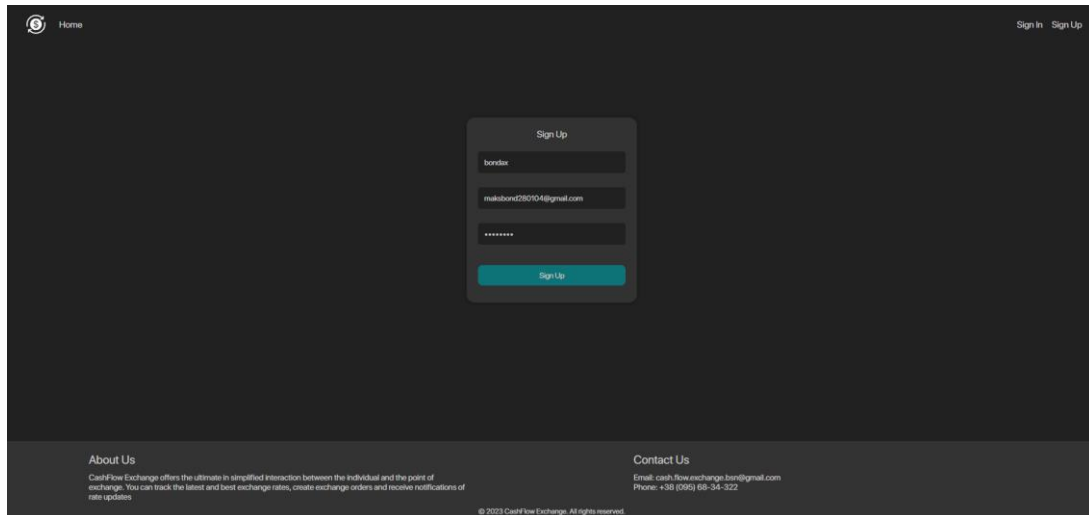


Рисунок 3.3.1 – Відтворення процесу реєстрації

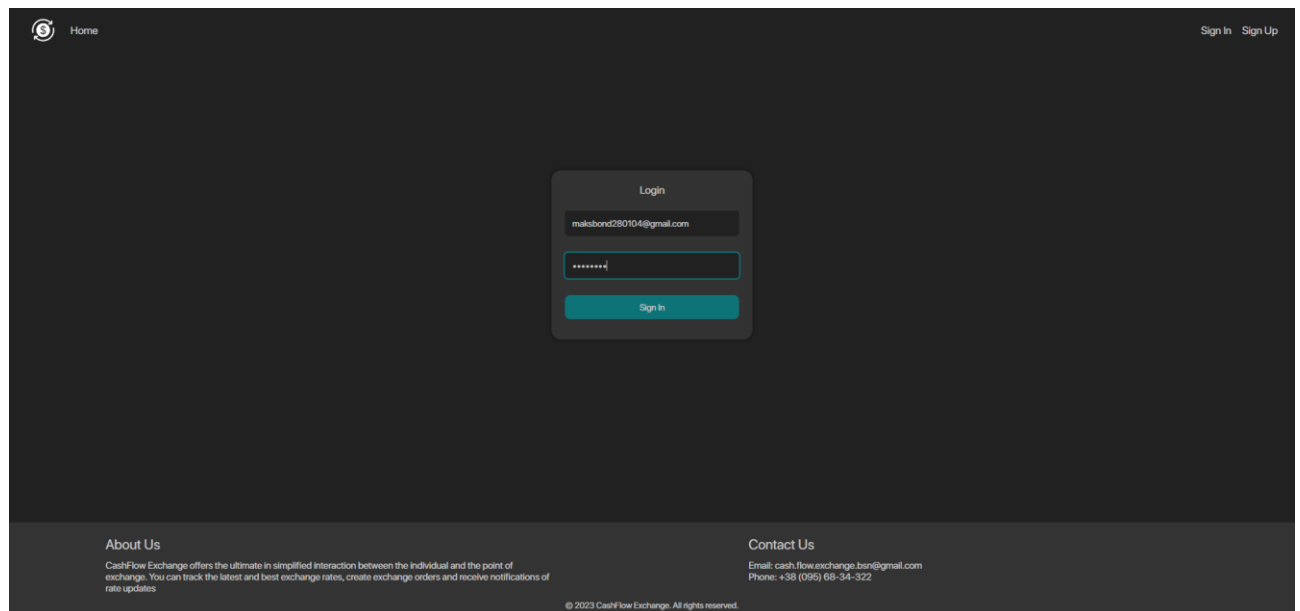


Рисунок 3.3.2 – Відтворення процесу авторизації

В обох випадках користувача буде перекинути на головну сторінку (рисунок 3.3.3) та авторизовано. З головної сторінки можна перевірити функціонал калькулятора валют, відтворимо ситуацію розрахунку в калькуляторі.

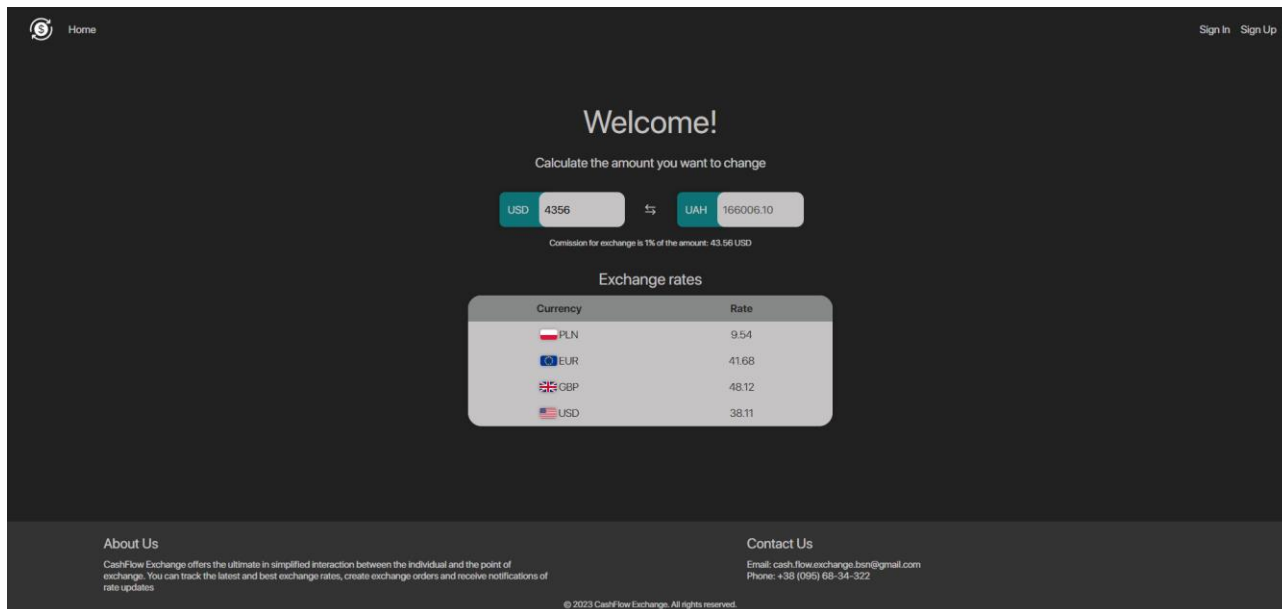


Рисунок 3.3.3 – Головна сторінка (для адміністратора)

Калькулятор відпрацював коректно, спробуємо перейти на сторінку створення запиту на обмін (рисунок 3.3.4) та спробуємо створити запит та подивитись на нього в історії

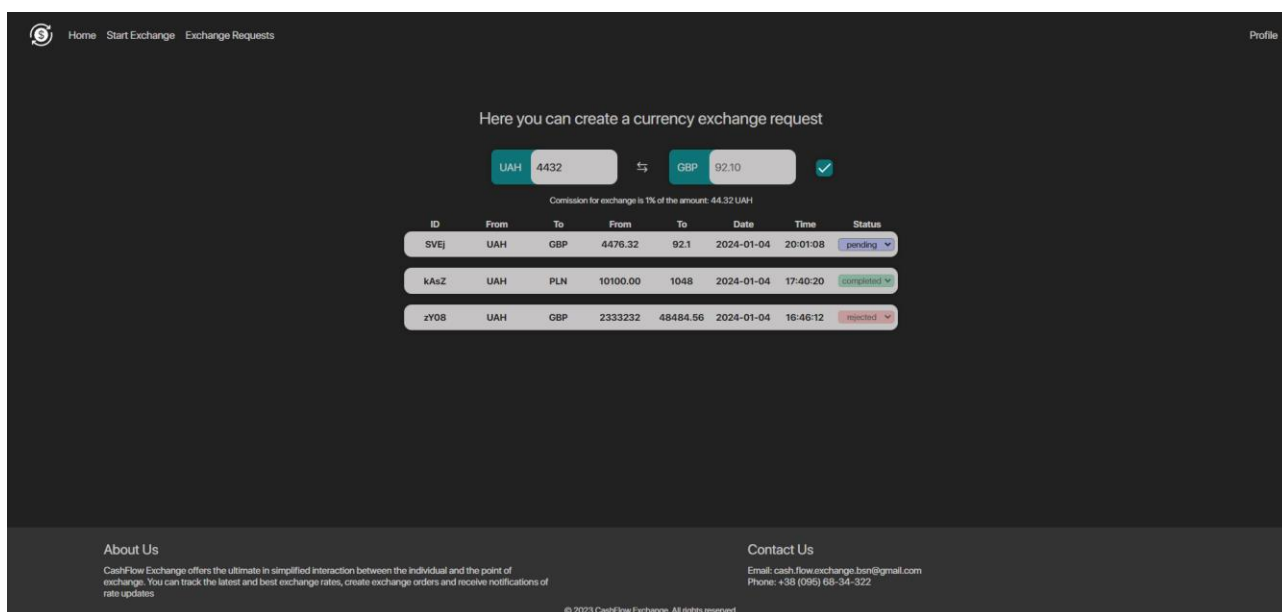


Рисунок 3.3.4 – Сторінка створення запиту на обмін

Спробуємо відмінити запит на створення обравши в пункті статус варіант “rejected”

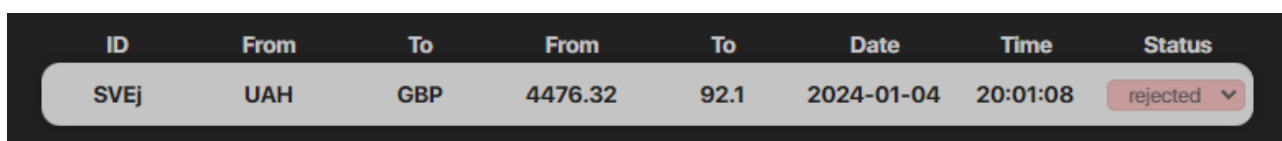


Рисунок 3.3.5 – Запит зі зміненим статусом.

Далі перейдемо на сторінку адміністратора для керування запитами на обмін (рисунок 3.3.6).

ID	From	To	From	To	Date	Time	Status
8H2r	UAH	EUR	4377.34	103.97	2024-01-04	20:02:00	pending
SVEJ	UAH	GBP	4476.32	92.1	2024-01-04	20:01:08	rejected
kAsZ	UAH	PLN	10100.00	1048	2024-01-04	17:40:20	completed
zY08	UAH	GBP	2333232	48484.56	2024-01-04	16:46:12	rejected
PRa7	UAH	USD	1008.99	26.21	2024-01-04	06:32:07	rejected
BYUK	EUR	GBP	1414.00	1213.18	2024-01-04	03:43:54	completed
2MkJ	UAH	USD	104.43	2.72	2024-01-04	03:02:48	rejected

About Us  
CashFlow Exchange offers the ultimate in simplified interaction between the individual and the point of exchange. You can track the latest and best exchange rates, create exchange orders and receive notifications of rate updates.

Contact Us  
Email: cash.flow.exchange.bsn@gmail.com  
Phone: +38 (095) 68-34-322

Рисунок 3.3.6 – Адміністраторська сторінка для керування створеними запитами.

Тепер спробуємо змінити статус запиту з UD 8H2r на “completed” (рисунок 3.3.7).

ID	From	To	From	To	Date	Time	Status
8H2r	UAH	EUR	4377.34	103.97	2024-01-04	20:02:00	completed

Рисунок 3.3.7 – Статус потрібного запиту змінено

Тепер перейдемо на сторінку профіля (рисунок 3.3.8)

Your Profile

Username: bondax

Email: maksbond280104@gmail.com

Current Password: Enter your current password

New Password: Enter your new password

Would you like to be notified when the exchange rate changes? Enter the minimum values next to the required currencies, when changing to which the email will be sent

USD: 0 EUR: 0 PLN: 0 GBP: 0

Save Changes

Sign Out Delete account

About Us  
CashFlow Exchange offers the ultimate in simplified interaction between the individual and the point of exchange. You can track the latest and best exchange rates, create exchange orders and receive notifications of rate updates.

Contact Us  
Email: cash.flow.exchange.bsn@gmail.com  
Phone: +38 (095) 68-34-322

© 2023 CashFlow Exchange. All rights reserved.

### Рисунок 3.3.8 – Сторінка профіля.

Тепер спробуємо змінити username та пароль (рисунок 3.3.9)

The screenshot displays the 'Your Profile' interface. At the top, navigation links include 'Home', 'Start Exchange', 'Exchange Requests', and a 'Profile' link. The profile form contains the following elements:

- Username:** A text input field containing 'bondax2'.
- Email:** A text input field containing 'maksbond280104@gmail.com'.
- Current Password:** A password input field with masked characters.
- New Password:** A password input field with masked characters.
- Notification Preferences:** A section asking 'Would you like to be notified when the exchange rate changes? Enter the minimum values next to the required currencies, when changing to which the email will be sent'. It includes input fields for USD, EUR, PLN, and GBP, each with a '0' value.
- Save Changes:** A prominent green button.
- Sign Out:** A red button.
- Delete account:** A red button.

Below the form, a green message states 'Profile info successfully updated'. The footer contains an 'About Us' section, a 'Contact Us' section with email and phone information, and a copyright notice: '© 2023 CashFlow Exchange. All rights reserved.'

### Рисунок 3.3.9 – username та пароль змінено.

Ну і вийдемо з акаунту шляхом натискання кнопки “Sign out” на сторінці профіля, після чого користувача переадресовує на головну сторінку.

### Висновки до розділу

У даному розділі було проведено детальний аналіз якості ПЗ за допомогою онлайн аналізаторів, який не виявив серйозних проблем в коді, тільки мінорні проблеми в стилі коду, також на якість коду при написанні позитивну вплинув інструмент статичного аналізу ESLint<sup>[16]</sup>. Було описано та виконано тестування застосунку, в результаті якого помилок в роботі ПЗ не було помічено. Було повністю описано та проведено імітацію використання функціоналу ПЗ на комп’ютері, що повністю відповідає вимогам, тестування дало позитивні результати. Отже, застосунок був повністю протестований та готовий до використання.



## 4 ВПРОВАДЖЕННЯ ТА СУПРОВІД ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

### 4.1 Розгортання програмного забезпечення

Даний проект буде розгорнуто на платформі Vercel<sup>[17]</sup>. Потрібно буде розгорнути тільки клієнтську частину, оскільки серверна частина за допомогою Firebase вже розгорнута у хмарі та функціонує без додаткових для цього зусиль.

Покрокова інструкція:

1. створити обліковий запис Vercel;
2. розмістити свій проект у гілці `deploy` в GitHub репозиторії;
3. у вкладці “Import Git Repository” потрібно під’єднати репозиторій з проектом;
4. в конфігурації проекту потрібно задати ім’я проекту, значення “Vite” у виборі “Framework Preset”, в “Root Directory” вказати `./frontend`;
5. у вкладці “Environment Variables” треба задати всі змінні середовища з `.env.local` файлу проекту;
6. натиснути кнопку “Deploy”;
7. якщо з’явилися якісь проблеми, то в створеному розгортанні треба вказати гілку “`deploy`” в налаштуваннях та перевірити пункт “Root Directory”.
8. перейти по посиланню на розгорнутий проект (рисунки 4.1.1)

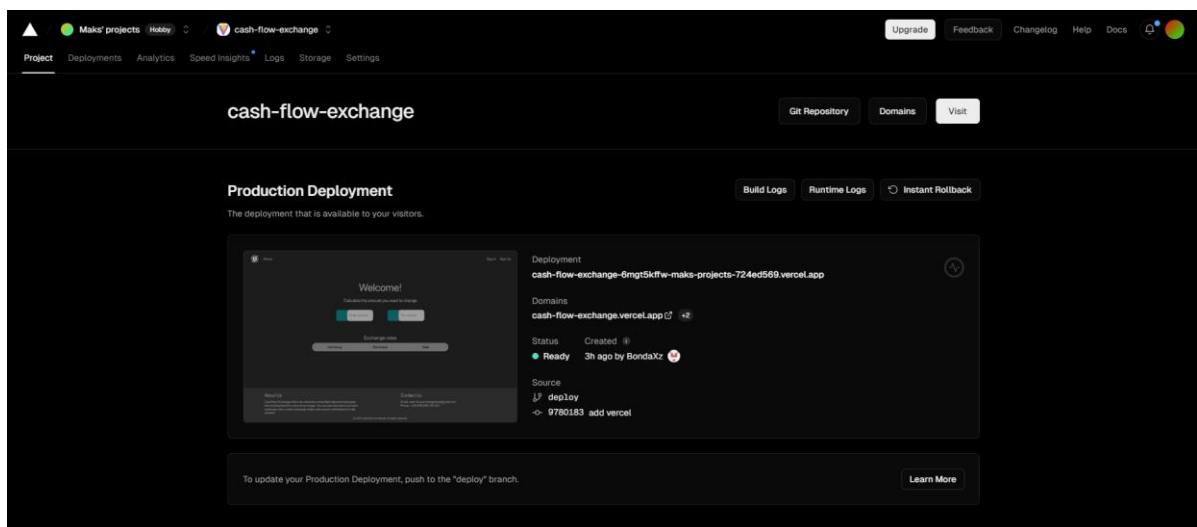


Рисунок 4.1.1 – Вигляд проекту після розгортання в інтерфейсі Vercel

Для подальшої розробки зручніше використовувати локальне розгортання, тож в терміналі потрібно перейти до папки з проектом і прописати по чергово: `“npm install”` та `“npm run dev”`. Після чого з’явиться посилання на локально розгорнутий проект.

## 4.2 Підтримка програмного забезпечення

Остання версія застосунку зберігається на GitHub репозиторії застосунку, там можна отримувати всі оновлення програми, для цього достатньо клонувати застосунок собі на комп’ютер.

### Висновки до розділу

В даному розділі було детально описано повний процес розгортання додатку, щоб кожен бажаючий міг розгорнути даний застосунок на платформі Vercel або локально на своєму комп’ютері, щоб можна було вносити зміни в код та дивитись ці зміни моментально, що дозволяє робити Vite.

## ВИСНОВКИ

В ході виконання курсової роботи була проведена інтенсивна робота над проектуванням та розробкою веб-застосунку для підтримки роботи пункту обміну валют. Застосовані сучасні технології та архітектурні підходи дозволили досягти високого рівня ефективності та гнучкості додатку.

Проект доступний для впровадження будь-яким фізичним пунктом обміну валют, який бажає оптимізувати свою роботу завдяки сучасним технологіям. Даний додаток був би корисним для будь-якого обмінника, який веде активну роботу на ринку та має велику кількість клієнтів.

Отримані результати курсової роботи вважаються задовільними та відповідають сучасному рівню наукових і технічних знань в області веб-розробки та інформаційних технологій. Використання сучасних технологій, таких як React для клієнтської частини та Firebase для серверної, свідчить про актуальність обраного стеку та відповідність його сучасним тенденціям у розробці веб-додатків.

Використання BPMN для моделювання бізнес-процесів дозволило стандартизувати термінологію та концепції, спрощуючи спільне розуміння команди розробників та бізнес-аналітиків.

Досліджувати дану тематику більш широко вважаю важливим, оскільки обмін валют ще стосується цифрових валют (криптовалют), що досі є новим напрямком у сфері фінансів.

В якості середовища розробки обрано Visual Studio Code, який давно зарекомендував себе, як якісний продукт для розробки подібних застосунків.

Огляд безпекових питань, таких як запобігання брутфорсу, XSS, демонструє уважний підхід до аспектів безпеки та відповідність сучасним стандартам забезпечення інформаційної безпеки.

Використання NoSQL бази даних Firestore вказує на усвідомлення сучасних підходів до зберігання та обробки даних у розподілених системах. Використання зовнішньої API для отримання актуальних курсів валют

відображає вміння інтегрувати зовнішні ресурси для забезпечення актуальності та коректності інформації.

Усі поставлені задачі було успішно виконано та реалізовано у веб-застосунку. Після реалізації застосунок було протестовано на різних пристроях з різними технічними та програмними характеристиками, щоб переконатись в коректній роботі застосунку незалежно від різних умов.

## СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

- 1) Currency Exchange: Definition, How It Works, and Where to Find It [Електронний ресурс] – <https://www.investopedia.com/terms/c/currency-exchange.asp>.
- 2) Гроші як сміття: 6 світових прикладів гіперінфляції [Електронний ресурс] – <https://loyer.com.ua/uk/groshi-yak-smitty-a-6-svitovyh-prykladiv-giperinfluyacziyi/>
- 3) Firebase Documentation [Електронний ресурс] – <https://firebase.google.com/docs>.
- 4) React [Електронний ресурс] – <https://react.dev/>
- 5) JavaScript [Електронний ресурс] – <https://developer.mozilla.org/ru/docs/Web/JavaScript>
- 6) Vite [Електронний ресурс] – <https://vitejs.dev/guide/why>
- 7) React Redux [Електронний ресурс] – <https://react-redux.js.org/>
- 8) What is Business Process Modeling Notation [Електронний ресурс] – <https://www.lucidchart.com/pages/bpmn>
- 9) What is Client Server Architecture? [Електронний ресурс] – <https://intellipaat.com/blog/what-is-client-server-architecture/>
- 10) Microservice architecture style [Електронний ресурс] – <https://learn.microsoft.com/en-us/azure/architecture/guide/architecture-styles/microservices>
- 11) Node.js Guides [Електронний ресурс] – <https://nodejs.org/en/guides>
- 12) What is a NoSQL database? [Електронний ресурс] – <https://www.ibm.com/topics/nosql-databases>
- 13) What Is A Brute Force Attack? [Електронний ресурс] – <https://www.fortinet.com/resources/cyberglossary/brute-force-attack>
- 14) Cross Site Scripting (XSS) [Електронний ресурс] – <https://owasp.org/www-community/attacks/xss/>
- 15) Codacy [Електронний ресурс] – <https://www.codacy.com/>
- 16) ESLint [Електронний ресурс] – <https://eslint.org/>

17) Vercel [Электронный ресурс] —  
<https://vercel.com/docs/deployments/overview>

Факультет інформатики та обчислювальної техніки

Кафедра інформатики та програмної інженерії

“ЗАТВЕРДЖЕНО”

Керівник роботи

\_\_\_\_\_ Ілля АХАЛАДЗЕ

“    ” \_\_\_\_\_ 2023 р.

## **ВЕБ-ЗАСТОСУНОК ПІДТРИМКИ РОБОТИ ПУНКТУ ОБМІНУ ВАЛЮТ**

**Технічне завдання**

КП.ІП-1304.045440.01.91

“ПОГОДЖЕНО”

Керівник роботи:

\_\_\_\_\_ Ілля АХАЛАДЗЕ

Виконавець:

\_\_\_\_\_ Максим БОНДАРЕНКО

## ЗМІСТ

1 НАЙМЕНУВАННЯ ТА ГАЛУЗЬ ЗАСТОСУВАННЯ	3
2 ПІДСТАВА ДЛЯ РОЗРОБКИ	4
3 ПРИЗНАЧЕННЯ РОЗРОБКИ	5
4 ВИМОГИ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ	6
4.1 Вимоги до функціональних характеристик	6
4.1.1 Користувацького інтерфейсу	6
4.1.2 Для користувача:	10
4.1.3 Для адміністратора системи (якщо він передбачений):	11
4.1.4 Додаткові вимоги:	11
4.2 Вимоги до надійності	11
4.3 Умови експлуатації	11
4.3.1 Вид обслуговування	11
4.3.2 Обслуговуючий персонал	12
4.4 Вимоги до складу і параметрів технічних засобів	12
4.5 Вимоги до інформаційної та програмної сумісності	12
4.5.1 Вимоги до вхідних даних	12
4.5.2 Вимоги до вихідних даних	12
4.5.3 Вимоги до мови розробки	12
4.5.4 Вимоги до середовища розробки	13
4.5.5 Вимоги до представленню вихідних кодів	13
4.6 Вимоги до маркування та пакування	13
4.7 Вимоги до транспортування та зберігання	13
4.8 Спеціальні вимоги	13
5 ВИМОГИ ДО ПРОГРАМНОЇ ДОКУМЕНТАЦІЇ	14
5.1 Попередній склад програмної документації	14
5.2 Спеціальні вимоги до програмної документації	14
6 СТАДІЇ І ЕТАПИ РОЗРОБКИ	15
7 ПОРЯДОК КОНТРОЛЮ ТА ПРИЙМАННЯ	16



## **1 НАЙМЕНУВАННЯ ТА ГАЛУЗЬ ЗАСТОСУВАННЯ**

Назва розробки: Веб-застосунок підтримки роботи пункту обміну валют.

Галузь застосування:

Наведене технічне завдання поширюється на розробку веб-застосунку CashFlow Exchange, котре використовується для налагодження процедури контакту між фізичною особою та пунктом обміну валют та призначена для користувачів, які хочуть максимально спростити процес обміну валют та отримувати корисну інформацію щодо зміни поточного курсу.

## **2 ПІДСТАВА ДЛЯ РОЗРОБКИ**

Підставою для розробки веб-застосунку CashFlow Exchange є оптимізація процесу комунікації між клієнтом та фізичним пунктом обміну валют, що є завданням курсової роботи.

### **3 ПРИЗНАЧЕННЯ РОЗРОБКИ**

Розробка призначена для моніторингу актуальних курсів валют, отримання сповіщень про певні зміни в курсах обраних валют, онлайн розрахунку обміну з урахуванням актуальних курсів, створення онлайн запитів на обмін з подальшим контактом клієнта з фізичним пунктом.

Метою розробки є спрощення процесу взаємодії має клієнтом та пунктом обміну валют.

## 4 ВИМОГИ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

### 4.1 Вимоги до функціональних характеристик

Програмне забезпечення повинно забезпечувати виконання наступних основних функцій:

#### 4.1.1 Користувацького інтерфейсу

- Користувач повинен мати можливість зареєструватися використовуючи внутрішню систему реєстрації. Елемент №1 (рис. 4.1) відповідає за підтвердження реєстрації, в результаті натиску кнопки буде створено нового користувача, якщо всі дані валідні. Після процедури реєстрації користувача переадресовую на “Home” сторінку. У випадку наявності зареєстрованого облікового запису та натискання на елемент №2 (рис. 4.1), користувача переадресовує на сторінку авторизації.

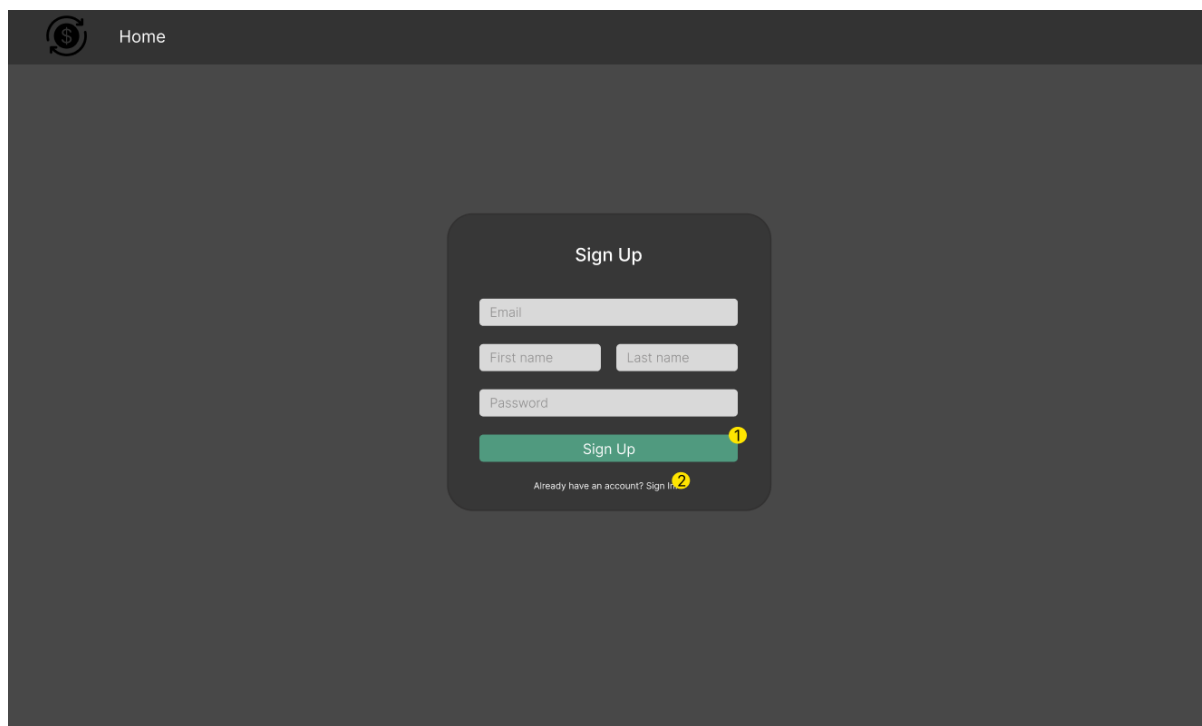
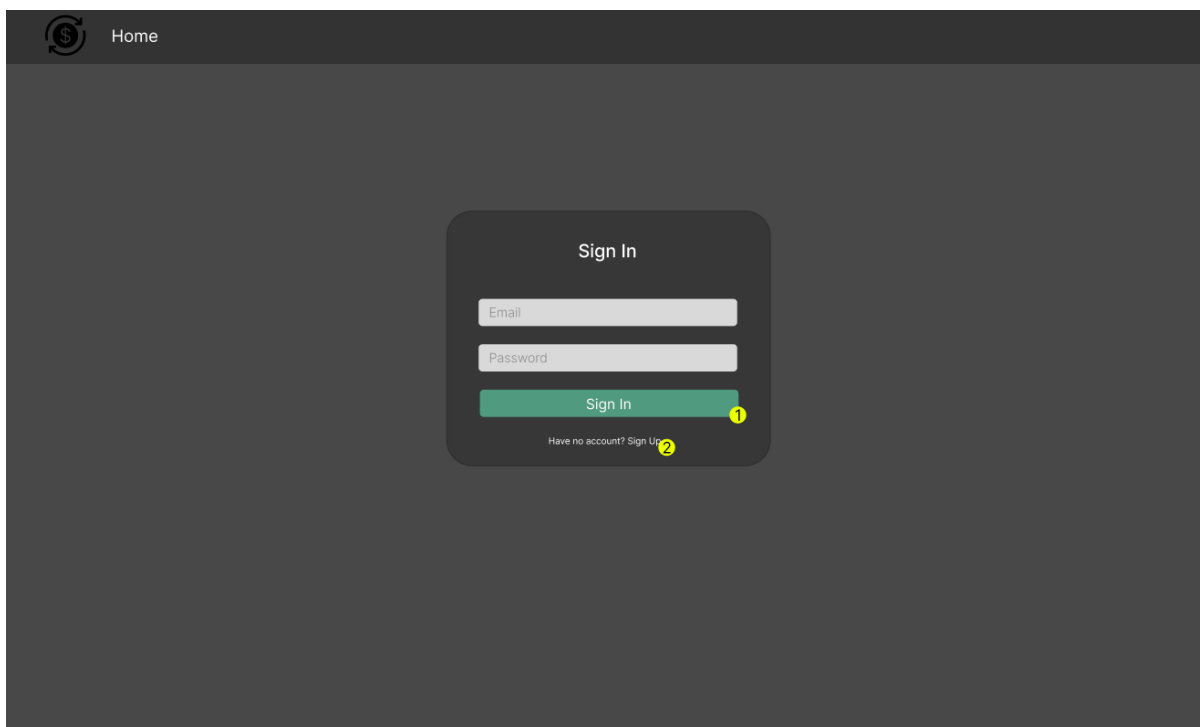


Рисунок 4.1 - прототип сторінки “Sign Up”

- Користувач повинен мати можливість авторизуватись використовуючи систему. Елемент №1 (рис. 4.2) відповідає за підтвердження введених даних і у випадку їх правильного вводу, користувач успішно авторизується та переадресовується на сторінку “Home”. У випадку відсутності облікового

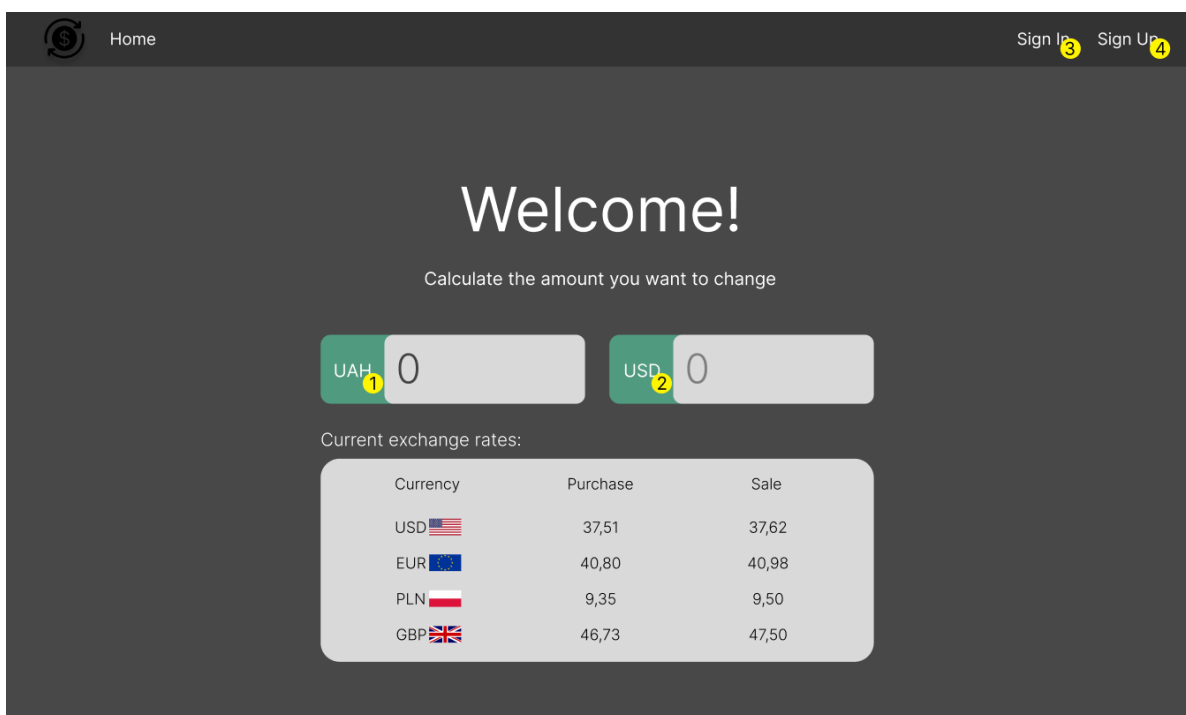
запису та натискання на елемент №2 (рис. 4.2) , користувача переадресовує на сторінку реєстрації.



The image shows a dark-themed 'Sign In' form. At the top left is a logo with a dollar sign and the word 'Home'. The form is centered and contains two input fields: 'Email' and 'Password'. Below these is a green 'Sign In' button with a yellow circle containing the number '1' next to it. At the bottom of the form, there is a link that says 'Have no account? Sign Up' with a yellow circle containing the number '2' next to it.

Рисунок 4.2 - прототип сторінки “Sign In”

- Користувач повинен мати можливість переглядати актуальні курси валют на сторінці “Home” (рис. 4.3).



The image shows a dark-themed 'Home' page. At the top left is a logo with a dollar sign and the word 'Home'. At the top right are links for 'Sign In' (with a yellow circle containing the number '3') and 'Sign Up' (with a yellow circle containing the number '4'). The main heading is 'Welcome!' followed by the text 'Calculate the amount you want to change'. Below this are two input fields: one for 'UAH' (with a yellow circle containing the number '1') and one for 'USD' (with a yellow circle containing the number '2'). Both fields currently show the value '0'. Below the input fields is a section titled 'Current exchange rates:' which contains a table.





Currency	Purchase	Sale
USD 	37,51	37,62
EUR 	40,80	40,98
PLN 	9,35	9,50
GBP 	46,73	47,50

Рисунок 4.3 - прототип сторінки “Home” неавторизованого користувача

- Користувач повинен мати можливість обирати з якої та в яку валюту відбувається конвертація в калькуляторі натискаючи на елемент №1 (рис. 4.3) та елемент №2 (рис. 4.3) відповідно.
- Неавторизований користувач повинен мати можливість перейти на сторінку реєстрації натисканням на елемент №4 (рис. 4.3) або на сторінку авторизації натисканням на елемент №3 (рис. 4.3), якщо має існуючий обліковий запис.
- Авторизований користувач повинен мати можливість перейти в налаштування власного облікового запису шляхом натискання елемента №1 (рис. 4.4).
- Авторизований користувач повинен мати можливість перейти на сторінку створення угоди з обміну валют натисканням на елемент №2 (рис. 4.4).

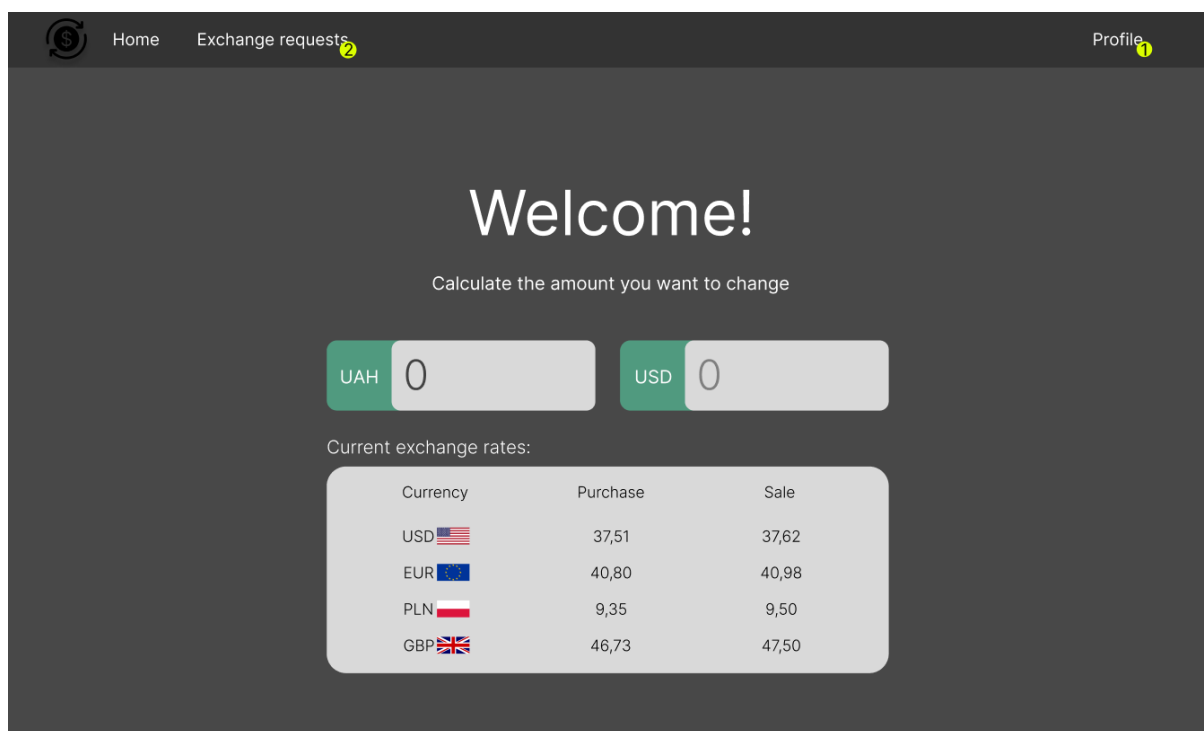


Рисунок 4.4 - прототип сторінки “Home” авторизованого користувача

- Користувач повинен мати можливість змінювати персональні дані (“First name”, “Last name”, “Email”, “Password”). Натискання елемента №1 (рис. 4.5) підтверджує та застосовує здійснені зміни.
- Користувач повинен мати можливість обирати значення зміни цікавого йому курсу, при якому про цю зміну буде йти сповіщення на пошту

користувачу. Натискання елемента №1 (рис. 4.5) підтверджує внесені зміни.

Home Start exchange Profile

Welcome to your profile information

First name: Maksym Last name: Bondarenko

Email: bondax28@gmail.com Password: \*\*\*\*\*

Would you like to be notified when the exchange rate changes?  
Enter the minimum values next to the required currencies, when  
changing to which the email will be sent

USD 0.5 EUR EUR GBP 1.0 PLN PLN

Submit changes 1

Рисунок 4.5 - прототип сторінки “Profile”

- Користувач повинен мати можливість обирати валюти між якими буде проводитись обмін шляхом натискання на елемент №1 (рис. 4.6), який відповідає за валюту, яку віддає користувач, та елемент №2 (рис. 4.6), який відповідає за валюту, яку користувач отримає. Натискання елемента №3 створює заявку на обмін валют.

Home Start exchange Profile

You can create your own currency exchange request

UAH 1 USD 2 Create request 3

Your exchange history:

Currency from	Currency to	Date and time	Status
UAH	USD	18/12/2023 12:51 PM	In review
UAH	USD	18/12/2023 08:45 AM	Take away
UAH	USD	15/12/2023 04:51 PM	Done
UAH	USD	14/12/2023 12:51 PM	Rejected

Рисунок 4.6 - прототип сторінки “Start exchange”

- Користувач повинен мати можливість переглядати історію своїх обмінів (валюти, між якими було проведено, дату та час, статус заявки).
- Користувач повинен мати можливість змінювати статус заявки на обмін з “In review” або “Take away” на “Rejected”, якщо з якихось причин передумав проводити операцію.
- Адміністратор системи повинен мати можливість переглядати усі створені запити та змінювати їх статус натисканням на елемент №1 (рис. 4.7). Адміністратор може ставити 4 статуси (“In review” - статус ставиться по стандарту після створення та значить, що заявка очікує перевірки адміністратором. “Take away” - статус ставиться, коли адміністратор одобрив заявку та клієнту треба прийти на фізичний пункт та виконати обмін. “Rejected” - статус ставиться за різних причин, коли обмін не може бути скоєним. “Done” - ставиться після успішного обміну вже на фізичному пункті.)

User ID	Full name	Currency from	Currency to	Date and time	Status
12321	Bondarenko Maksym	UAH	USD	18/12/2023 12:51 PM	In review <sup>1</sup>
12321	Bondarenko Maksym	UAH	USD	18/12/2023 08:45 AM	Take away
12321	Bondarenko Maksym	UAH	USD	15/12/2023 04:51 PM	Done
12321	Bondarenko Maksym	UAH	USD	14/12/2023 12:51 PM	Rejected

Рисунок 4.7 - прототип сторінки “Exchange requests” з приватним доступом для адміністратора

#### 4.1.2 Для користувача:

- реєстрація нового облікового запису;



- вхід в існуючий обліковий запис;
- зміна особистих даних (пароль, ім'я та прізвище, email);
- отримання сповіщень на пошту про зміну курсу;
- розраховувати валюту в онлайн калькуляторі;
- переглядати актуальний курс валют;
- створювати запити на обмін валют;
- передивлятись історію своїх запитів з детальною інформацією;
- змінювати статус своїх запитів з “In review” або “Take away” на “Rejected” за потреби.

#### 4.1.3 Для адміністратора системи (якщо він передбачений):

- усі функції адміністратора, окрім створення запитів на обмін валют;
- переглядати список усіх створених запитів різними користувачами;
- змінювати статус запитів на обмін.

#### 4.1.4 Додаткові вимоги:

- можливість розширення та оновлення функціоналу;
- забезпечення сумісності з різними браузерами;
- забезпечення валідації даних у полях вводу в калькуляторі обміну, формах авторизації, реєстрації та зміни персональних даних.

#### 4.2 Вимоги до надійності

Передбачити контроль введення інформації та захист від некоректних дій користувача. Забезпечити цілісність інформації в базі даних.

#### 4.3 Умови експлуатації

Умови експлуатації згідно СанПін 2.2.2.542 – 96.

##### 4.3.1 Вид обслуговування

Вимоги до виду обслуговування не висуваються.

#### 4.3.2 Обслуговуючий персонал

4.3.3 Адміністратор застосунку - основна задача полягає в розгляді запитів клієнтів на обмін валют.

#### 4.4 Вимоги до складу і параметрів технічних засобів

Програмне забезпечення повинно функціонувати на IBM-сумісних персональних комп'ютерах.

Мінімальна конфігурація технічних засобів:

- тип процесору: Intel Core i5;
- об'єм ОЗП: 4 Гб;
- підключення до мережі Інтернет зі швидкістю від 20 мегабіт;

Рекомендована конфігурація технічних засобів:

- тип процесору: Intel Core i5;
- об'єм ОЗП: 8 Гб;
- підключення до мережі Інтернет зі швидкістю від 100 мегабіт;

#### 4.5 Вимоги до інформаційної та програмної сумісності

Програмне забезпечення повинно працювати під управлінням операційних систем сімейства WIN32 (Windows'XP, Windows NT і т.д.) або Unix.

##### 4.5.1 Вимоги до вхідних даних

Вимоги до вхідних даних не висуваються.

##### 4.5.2 Вимоги до вихідних даних

Вимоги до результатів не висуваються.

##### 4.5.3 Вимоги до мови розробки

Розробку виконати на мові програмування JavaScript, платформа Firebase, бібліотека ReactJS.

#### 4.5.4 Вимоги до середовища розробки

Розробку виконати на платформі Visual Studio Code.

#### 4.5.5 Вимоги до представленню вихідних кодів

Вихідний код програми має бути представлений у вигляді завантаженого проекту на GitHub.

#### 4.6 Вимоги до маркування та пакування

Вимоги до маркування та пакування не висуваються.

#### 4.7 Вимоги до транспортування та зберігання

Вимоги до транспортування та зберігання не висуваються.

#### 4.8 Спеціальні вимоги

Застосунок повинен бути готовим до розгортання.

## **5 ВИМОГИ ДО ПРОГРАМНОЇ ДОКУМЕНТАЦІЇ**

### **5.1 Попередній склад програмної документації**

У склад супроводжувальної документації повинні входити наступні документи на аркушах формату А4:

- пояснювальна записка;
- технічне завдання;
- керівництво користувача;
- керівництво адміністратора;
- програма та методика тестування;
- текст програми.

Графічна частина повинна бути виконана на аркушах формату А3 та містити наступні документи:

- схема структурна програмного забезпечення;
- схема структурна варіантів використання;
- схема структурна бази даних;
- креслення вигляду екранних форм.

### **5.2 Спеціальні вимоги до програмної документації**

Програмні модулі, котрі розробляються, повинні бути задокументовані, тобто тексти програм повинні містити всі необхідні коментарі.

## 6 СТАДІЇ І ЕТАПИ РОЗРОБКИ

№	Назва етапу	Строк	Звітність
1.	Вивчення літератури за тематикою роботи	12.11	
2.	Розробка технічного завдання	15.11	Технічне завдання
3.	Аналіз вимог та уточнення специфікацій	17.11	Специфікації програмного забезпечення
4.	Проектування структури програмного забезпечення, проектування компонентів	19.11	Схема структурна програмного забезпечення та специфікація компонентів (діаграма класів, схема алгоритму)
5.	Програмна реалізація програмного забезпечення	21.11	Тексти програмного забезпечення
6.	Тестування програмного забезпечення	25.11	Тести, результати тестування
7.	Розробка матеріалів текстової частини роботи	21.11	Пояснювальна записка
8.	Розробка матеріалів графічної частини роботи	25.12	Графічний матеріал проекту
9.	Оформлення технічної документації роботи	30.12	Технічна документація

## **7 ПОРЯДОК КОНТРОЛЮ ТА ПРИЙМАННЯ**

Тестування розробленого програмного продукту виконується відповідно до “Програми та методики тестування”.

Факультет інформатики та обчислювальної техніки  
Кафедра інформатики та програмної інженерії

“ЗАТВЕРДЖЕНО”

Керівник роботи

\_\_\_\_\_ Ілля АХАЛАДЗЕ

“ \_\_\_\_ ” \_\_\_\_\_ 2024 р.

**ВЕБ-ЗАСТОСУНОК ПІДТРИМКИ РОБОТИ ПУНКТУ ОБМІНУ  
ВАЛЮТ**

**Текст програми**

КПІ.ПІ-1304.045440.05.13

“ПОГОДЖЕНО”

Керівник роботи:

\_\_\_\_\_ Ілля АХАЛАДЗЕ

Консультант:

\_\_\_\_\_ Максим ГОЛОВЧЕНКО

Виконавець:

\_\_\_\_\_ Максим БОНДАРЕНКО

Київ – 2024

### **Файл main.jsx**

```
import React from "react";
import ReactDOM from "react-dom/client";
import App from "./App.jsx";
import "./index.css";
import { BrowserRouter } from "react-router-dom";
import { Provider } from "react-redux";
import store from "./storage/store.js";

ReactDOM.createRoot(document.getElementById("root")).render(
  <Provider store={store}>
    <React.StrictMode>
      <BrowserRouter>
        <App />
      </BrowserRouter>
    </React.StrictMode>
  </Provider>
);
```

### **Файл App.jsx**

```
import { Routes, Route } from "react-router-dom";
import SignUp from "./pages/SignUp/SignUp";
import Header from "./components/Header/Header";
import Home from "./pages/Home/Home";
import SignIn from "./pages/SignIn/SignIn";
import StartExchange from "./pages/StartExchange/StartExchange";
import { useDispatch } from "react-redux";
import { setUser, removeUser } from "./reducers/userReducer";
import { useEffect } from "react";
import { auth } from "./firebase";
```



```

import { db } from "./firebase";
import { doc, onSnapshot } from "firebase/firestore";
import Profile from "./pages/Profile/Profile";
import AdminExchangeRequests from
"./pages/AdminExchangeRequests/AdminExchangeRequests";
import Footer from "./components/Footer/Footer";
import { useState } from "react";
import classes from "./App.module.css";

```

```

function App() {
  const dispatch = useDispatch();
  const [loading, setLoading] = useState(true);

  useEffect(() => {
    const unsubscribe = auth.onAuthStateChanged((user) => {
      if (user) {
        const userRef = doc(db, "users", user.uid);
        onSnapshot(userRef, (snapshot) => {
          if (snapshot.exists()) {
            const data = snapshot.data();
            dispatch(
              setUser({
                username: data.username,
                email: user.email,
                currencyDiff: data.currencyDiff,
                isAdmin: data.isAdmin,
                uid: user.uid,
              })
            );
          } else {

```

```

        console.log("User does not exist in Firestore.");
    }
});
setLoading(false);
} else {
    dispatch(removeUser());
    setLoading(false);
}
});
return () => {
    unsubscribe();
};
}, [dispatch]);

return (
    <>
    {loading ? (
        <div className={classes.ldsDualRing}></div>
    ) : (
        <>
        <Header />
        <Routes>
            <Route path="/signup" element={<SignUp />} />
            <Route path="/" element={<Home />} />
            <Route path="/signin" element={<SignIn />} />
            <Route path="/profile" element={<Profile />} />
            <Route path="/start-exchange" element={<StartExchange />} />
            <Route
                path="/exchange-requests"
                element={<AdminExchangeRequests />}
            />
        </Routes>
        </>
    )}
    </>
);

```

```

        />
      </Routes>
      <Footer />
    </>
  )}
</>
);
}

export default App;

```

### **Файл store.js**

```

import { configureStore } from "@reduxjs/toolkit";
import userReducer from "../reducers/userReducer";
import { combineReducers } from "redux";

```

```

const rootReducer = combineReducers({
  user: userReducer,
});

```

```

const store = configureStore({
  reducer: rootReducer,
  devTools: true,
});

```

```

export default store;

```

### **Файл userReducer.js**

```

import { createSlice } from "@reduxjs/toolkit";

```

```
const initialState = {  
  username: null,  
  email: null,  
  currencyDiff: {  
    USD: null,  
    EUR: null,  
    PLN: null,  
    GBP: null,  
  },  
  isAdmin: false,  
  uid: null,  
};
```

```
const authSlice = createSlice({  
  name: "user",  
  initialState,  
  reducers: {  
    setUser: (state, action) => {  
      state.username = action.payload.username;  
      state.email = action.payload.email;  
      state.currencyDiff = action.payload.currencyDiff;  
      state.isAdmin = action.payload.isAdmin;  
      state.uid = action.payload.uid;  
    },  
    removeUser: (state) => {  
      state.username = null;  
      state.email = null;  
      state.currencyDiff = {  
        USD: null,  
        EUR: null,
```

```

    PLN: null,
    GBP: null,
  };
  state.isAdmin = false;
  state.uid = null;
},
},
));

export const { setUser, removeUser } = authSlice.actions;
export default authSlice.reducer;

```

### **Файл useAuth.jsx**

```

import { useSelector } from "react-redux";

export function useAuth() {
  const { username, email, currencyDiff, isAdmin, uid } =
    useSelector((state) => state.user);

  return {
    isAuth: !!uid,
    username,
    email,
    currencyDiff,
    isAdmin,
    uid,
  };
}

```

### Файл **SignUpForm.jsx**

```
import { useState } from "react";
// import classes from './SignUpForm.module.css';
import { getAuth, createUserWithEmailAndPassword } from "firebase/auth";
import { db } from "../firebase";
// import { collection, addDoc } from 'firebase/firestore';
import { sendEmailVerification } from "firebase/auth";
import { useEffect } from "react";
import { doc, setDoc } from "firebase/firestore";
import classes from "../SignInForm/SignInForm.module.css";
import { useNavigate } from "react-router-dom";
```

```
const SignUpForm = () => {
  const [formData, setFormData] = useState({
    username: "",
    email: "",
    password: "",
  });

  const [errors, setErrors] = useState({
    username: "",
    email: "",
    password: "",
  });

  const navigate = useNavigate();

  const handleInputChange = (e) => {
    const { name, value } = e.target;
    setFormData((prevData) => ({ ...prevData, [name]: value }));
```

```
setErrors((prevErrors) => ({ ...prevErrors, [name]: "" }));  
};
```

```
const validateForm = () => {  
  let isValid = true;  
  const newErrors = {};
```

```
  const usernameRegex = /^[a-zA-Z0-9_]+$/  
  const emailRegex = /^[^\s@]+@[^\s@]+\.[^\s@]+$/;
```

```
  const validateName = (fieldName, regex) => {  
    if (!formData[fieldName].trim() || !regex.test(formData[fieldName])) {  
      newErrors[fieldName] = `Valid ${fieldName.toLowerCase()} is required`;  
      isValid = false;  
    }  
  };  
};
```

```
validateName("username", usernameRegex);  
validateName("email", emailRegex);
```

```
const passwordRegex = /^[a-zA-Z0-9!@#$$%^&*]{8,16}$/;
```

```
if (!passwordRegex.test(formData.password)) {  
  newErrors.password =  
    "Password must be 8-16 characters long, contain only numbers, letters and  
special characters";  
  isValid = false;  
}
```

```
setErrors(newErrors);
```

```

    return isValid;
  };

const handleSubmit = async (e) => {
  e.preventDefault();

  if (validateForm()) {
    try {
      const auth = getAuth();

      const userCredential = await createUserWithEmailAndPassword(
        auth,
        formData.email,
        formData.password
      );

      const user = userCredential.user;

      const userRef = doc(db, "users", user?.uid);

      await sendEmailVerification(auth.currentUser);

      await setDoc(
        userRef,
        {
          username: formData.username,
          email: formData.email,
          currencyDiff: {
            USD: null,
            EUR: null,

```



```

        PLN: null,
        GBP: null,
    },
    isAdmin: false,
},
{
    maxAttempts: 1,
    backoffMillis: 3000,
}
);

console.log("Verification email sent. Please verify your email.");
navigate("/");
} catch (error) {
    console.error("Error registering user:", error.message);
    if (error.code === "auth/email-already-in-use") {
        setErrors((prevErrors) => ({
            ...prevErrors,
            general: "Email is already in use",
        }));
    } else {
        setErrors((prevErrors) => ({
            ...prevErrors,
            general: error.message,
        }));
    }
}
}
};

```

```

return (
  <div className={classes.signInContainer}>
    <div className={classes.signInForm}>
      <h2 className={classes.text}>Sign Up</h2>
      <form onSubmit={handleSubmit}>
        <div className={classes.formGroup}>
          <input
            className={classes.input}
            type="text"
            name="username"
            value={formData.username}
            onChange={handleInputChange}
            placeholder="Username"
          />
          <div className={classes.error}>{errors.username}</div>
        </div>
        <div className={classes.formGroup}>
          <input
            className={classes.input}
            type="email"
            name="email"
            value={formData.email}
            onChange={handleInputChange}
            placeholder="Email"
            autoComplete="username"
          />
          <div className={classes.error}>{errors.email}</div>
        </div>
        <div className={classes.formGroup}>
          <input

```

```

        className={classes.input}
        type="password"
        name="password"
        value={formData.password}
        onChange={handleInputChange}
        placeholder="Password"
        autoComplete="current-password"
      />
      <div className={classes.error}>{errors.password}</div>
    </div>

    <div className={classes.myBtnCont}>
      <button type="submit" className={classes.myBtn}>
        Sign Up
      </button>
      <div className={classes.errorGeneral}>{errors.general}</div>
    </div>
  </form>
</div>
</div>
);
};

export default SignUpForm;

```

### **Файл SignInForm.jsx**

```

import { useState } from "react";
import classes from "../SignInForm.module.css";
import { getAuth, signInWithEmailAndPassword } from "firebase/auth";
import { useNavigate } from "react-router-dom";

```

```

import { db } from "../firebase";
import { doc, updateDoc } from "firebase/firestore";

const SignInForm = () => {
  const [formData, setFormData] = useState({
    email: "",
    password: "",
  });

  const [errors, setErrors] = useState("");

  const navigate = useNavigate();

  const handleChange = (e) => {
    const { name, value } = e.target;
    setFormData((prevData) => ({ ...prevData, [name]: value }));
  };

  const handleSubmit = async (e) => {
    e.preventDefault();

    try {
      const auth = getAuth();
      const { user } = await signInWithEmailAndPassword(
        auth,
        formData.email,
        formData.password
      );
      await updateDoc(doc(db, "users", auth.currentUser.uid), {
        email: auth.currentUser.email,

```

```

    });
    if (user) {
        navigate("/");
    }
} catch (error) {
    console.error("Error signing in:", error.message);
    switch (error.code) {
        case "auth/invalid-credential":
            setErrors("Invalid email or password");
            break;
        case "auth/too-many-requests":
            setErrors("Many failed login attempts. Please try again later");
            break;
        case "auth/invalid-email":
            setErrors("Invalid email");
            break;
        case "auth/missing-password":
            setErrors("Missing password");
            break;
        default:
            setErrors("An error occurred while signing in");
            break;
    }
}
};

return (
    <div className={classes.signInContainer}>
        <div className={classes.signInForm}>
            <h2 className={classes.text}>Login</h2>

```

```

<form onSubmit={handleSubmit}>
  <div className={classes.formGroup}>
    <input
      className={classes.input}
      type="email"
      name="email"
      value={formData.email}
      onChange={handleChange}
      placeholder="Email"
      autoComplete="username"
    />
  </div>
  <div className={classes.formGroup}>
    <input
      className={classes.input}
      type="password"
      name="password"
      value={formData.password}
      onChange={handleChange}
      placeholder="Password"
      autoComplete="current-password"
    />
  </div>

  <div className={classes.formGroup}>
    <button className={classes.signin} type="submit">
      Sign In
    </button>
    {errors && <div className={classes.error}>{errors}</div>}
  </div>

```

```

        </form>
      </div>
    </div>
  );
};

export default SignInForm;

```

### **Файл RequestsHistory.jsx**

```

import { useEffect, useState } from "react";
import { db } from "../../firebase";
import { collection } from "firebase/firestore";
import RequestRow from "../../RequestRow/RequestRow";
import classes from "../../RequestsHistory.module.css";
import { query, where, orderBy } from "firebase/firestore";
import { useAuth } from "../../hooks/useAuth";
import { format } from "date-fns";
import { onSnapshot } from "firebase/firestore";

const RequestsHistory = ({ showAdmin }) => {
  const [requestsHistory, setRequestsHistory] = useState([]);
  const user = useAuth();

  useEffect(() => {
    const fetchRequestHistory = async () => {
      try {
        const requestsCollection = collection(db, "requests");

        let requestsQuery = null;
        if (showAdmin) {

```

```

requestsQuery = query(
  requestsCollection,
  orderBy("timestamp", "desc")
);
} else {
  requestsQuery = query(
    requestsCollection,
    where("userId", "==", user.uid),
    orderBy("timestamp", "desc")
  );
}

const unsubscribe = onSnapshot(requestsQuery, (querySnapshot) => {
  const historyData = querySnapshot.docs.map((doc) => {
    const { timestamp, ...rest } = doc.data();

    const formattedDate = format(timestamp.toDate(), "yyyy-MM-dd");
    const formattedTime = format(timestamp.toDate(), "HH:mm:ss");

    const shortId = doc.id.slice(-4);

    return {
      id: doc.id,
      shortId: shortId,
      date: formattedDate,
      time: formattedTime,
      ...rest,
    };
  });
});

```



```

        setRequestsHistory(historyData);
    });
    return () => {
        unsubscribe();
    };
} catch (error) {
    console.error("Error fetching request history:", error.message);
}
};

fetchRequestHistory();
}, [user.uid]);

return (
    <div className={classes.requestsHistory}>
        <table className={classes.table}>
            <thead>
                {requestsHistory.length > 0 ? (
                    <tr className={classes.header}>
                        <th className={classes.label}>ID</th>
                        <th className={classes.label}>From</th>
                        <th className={classes.label}>To</th>
                        <th className={classes.label}>From</th>
                        <th className={classes.label}>To</th>
                        <th className={classes.label}>Date</th>
                        <th className={classes.label}>Time</th>
                        <th className={classes.status}>Status</th>
                    </tr>
                ) : null}
            </thead>

```

```

    <tbody className={classes.tbody}>
      {requestsHistory.map((request) => (
        <RequestRow key={request.id} request={request} />
      ))}
    </tbody>
  </table>
</div>
);
};

```

```
export default RequestsHistory;
```

### **Файл RequestRow.jsx**

```

import classes from "./RequestRow.module.css";
import { useState } from "react";
import { doc, updateDoc } from "firebase/firestore";
import { db } from "../../firebase";
import { useAuth } from "../../hooks/useAuth";

const RequestRow = ({ request }) => {
  const user = useAuth();
  const [status, setStatus] = useState(request.status);

  const handleStatusChange = async (event) => {
    const newStatus = event.target.value;

    console.log(user.isAdmin);
    if (!user.isAdmin && newStatus === "rejected") {
      const requestDocRef = doc(db, "requests", request.id);
      await updateDoc(requestDocRef, { status: newStatus });
    }
  };

```

```

        setStatus(newStatus);
    } else if (user.isAdmin) {
        const requestDocRef = doc(db, "requests", request.id);
        await updateDoc(requestDocRef, { status: newStatus });

        setStatus(newStatus);
    }
};

const getStatusColor = () => {
    switch (status) {
        case "completed":
            return "#6CAA8C";
        case "rejected":
            return "#c48a8a";
        case "take away":
            return "#cbba9c";
        case "pending":
            return "#9ca1cb";
        default:
            return "white";
    }
};

return (
    <tr className={classes.requestRow}>
        <td className={classes.info}>{request.shortId}</td>
        <td className={classes.info}>{request.sourceCurrency}</td>
        <td className={classes.info}>{request.targetCurrency}</td>

```

```

<td className={classes.info}>{request.amount}</td>
<td className={classes.info}>{request.convertedAmount}</td>
<td className={classes.info}>{request.date}</td>
<td className={classes.info}>{request.time}</td>
<td className={classes.select}>
  <select
    className={classes.select}
    value={status}
    onChange={handleStatusChange}
    disabled={status === "completed" || status === "rejected"}
    style={{ backgroundColor: getStatusColor() }}
  >
    <option className={classes.option} value="pending">
      pending
    </option>
    <option className={classes.option} value="completed">
      completed
    </option>
    <option className={classes.option} value="rejected">
      rejected
    </option>
    <option className={classes.option} value="take away">
      take away
    </option>
  </select>
</td>
</tr>
);
};

```

```
export default RequestRow;
```

### **Файл ProfileInfo.jsx**

```
import { useState } from "react";
import { useAuth } from "../../hooks/useAuth";
import classes from "./ProfileInfo.module.css";
import { useNavigate } from "react-router-dom";
import { db } from "../../firebase";
import { deleteDoc, doc } from "firebase/firestore";
import { updateDoc } from "firebase/firestore";
import { useEffect } from "react";
import {
  EmailAuthProvider,
  getAuth,
  updatePassword,
  verifyBeforeUpdateEmail,
} from "firebase/auth";
import { reauthenticateWithCredential } from "firebase/auth";

const ProfileInfo = () => {
  const navigate = useNavigate();
  const auth = getAuth();
  const user = auth.currentUser;
  const currentEmail = user.email;

  const { username, email, currencyDiff, uid, isAuth } = useAuth();
  const [formData, setFormData] = useState({
    username,
    email,
```

```
currencyDiff,  
curPassword: "",  
newPassword: "",  
});
```

```
const [errors, setErrors] = useState({  
  username: "",  
  email: "",  
  curPassword: "",  
  newPassword: "",  
  usd: "",  
  eur: "",  
  pln: "",  
  gbp: "",  
});
```

```
const [showPopup, setShowPopup] = useState(false);
```

```
useEffect(() => {  
  if (showPopup) {  
    const timeoutId = setTimeout(() => {  
      setShowPopup(false);  
    }, 5000);  
  
    return () => clearTimeout(timeoutId);  
  }  
}, [showPopup]);
```

```
useEffect(() => {  
  if (uid) {
```

```

    setFormData({
      username,
      email,
      currencyDiff,
    });
  }
}, [uid]);

const handleChange = (e) => {
  const { name, value } = e.target;

  if (name === "USD" || name === "EUR" || name === "PLN" || name ===
"GBP") {
    setFormData((prevData) => ({
      ...prevData,
      currencyDiff: { ...prevData.currencyDiff, [name]: value },
    }));
    return;
  } else if (name === "curPassword" || name === "newPassword") {
    setFormData((prevData) => ({ ...prevData, [name]: value }));
  }
  setFormData((prevData) => ({ ...prevData, [name]: value }));
  setErrors((prevErrors) => ({ ...prevErrors, [name]: "" }));
};

const validateForm = () => {
  let isValid = true;
  const newErrors = {};
  const usernameRegex = /^[a-zA-Z0-9_]+$;/;
  const emailRegex = /^[^\s@]+@[^\s@]+\.[^\s@]+$;/;

```

```
const passwordRegex = /^[a-zA-Z0-9!@#$$%^&*]{8,16}$/;
```

```
const validateName = (fieldName, regex) => {  
  if (!formData[fieldName].trim() || !regex.test(formData[fieldName])) {  
    newErrors[fieldName] = `Valid ${fieldName.toLowerCase()} is required`;  
    isValid = false;  
  }  
};
```

```
const validateCurrency = (currencyName) => {  
  if (  
    isNaN(formData.currencyDiff[currencyName]) ||  
    formData.currencyDiff[currencyName] < 0 ||  
    formData.currencyDiff[currencyName] > 10 ||  
    formData.currencyDiff[currencyName] === "-0"  
  ) {  
    newErrors[currencyName.toLowerCase()] = "Must be from 0 to 10.0";  
    isValid = false;  
  }  
};
```

```
const validatePassword = (fieldName, regex) => {  
  if (!regex.test(formData[fieldName]) && formData[fieldName].trim()) {  
    newErrors[fieldName] =  
      "Password must be 8-16 characters long, contain only numbers, letters and  
special characters";  
    isValid = false;  
  }  
};
```



```

validateName("username", usernameRegex);
validateName("email", emailRegex);

["USD", "EUR", "PLN", "GBP"].forEach((currency) =>
  validateCurrency(currency)
);

validatePassword("curPassword", passwordRegex);
validatePassword("newPassword", passwordRegex);

setErrors(newErrors);
return isValid;
};

const handleSubmit = async (e) => {
  e.preventDefault();

  if (validateForm()) {
    try {
      await updateDoc(doc(db, "users", uid), {
        username: formData.username,
        currencyDiff: {
          USD: formData.currencyDiff.USD || null,
          EUR: formData.currencyDiff.EUR || null,
          PLN: formData.currencyDiff.PLN || null,
          GBP: formData.currencyDiff.GBP || null,
        },
      });

      if (currentEmail !== formData.email) {

```

```

const credential = EmailAuthProvider.credential(
  currentEmail,
  formData.curPassword
);
await reauthenticateWithCredential(user, credential);
navigate("/signin");

await verifyBeforeUpdateEmail(user, formData.email);
await auth.signOut();
}

if (formData.newPassword) {
  const credential = EmailAuthProvider.credential(
    currentEmail,
    formData.curPassword
  );
  await reauthenticateWithCredential(user, credential);
  await updatePassword(user, formData.newPassword);
}

setShowPopup(true);
} catch (error) {
  console.error("Error updating profile:", error.message);
  if (
    error.code === "auth/invalid-credential"
  ) {
    setErrors((prevErrors) => ({
      ...prevErrors,
      curPassword: "Invalid password",
    }));
  }
}

```

```

    } else if (error.code === "auth/too-many-requests") {
      setErrors((prevErrors) => ({
        ...prevErrors,
        curPassword: "Too many requests. Try again later",
      }));
    } else if (error.code === "auth/missing-password")
      setErrors((prevErrors) => ({
        ...prevErrors,
        curPassword: "Enter your password",
      }));
  }
}
};

```

```

const handleSignOut = async () => {
  try {
    navigate("/");
    await auth.signOut();
  } catch (error) {
    console.error("Error signing out:", error.message);
  }
};

```

```

const handleDeleteAccount = async () => {
  try {
    const credential = EmailAuthProvider.credential(
      currentEmail,
      formData.curPassword
    );
    await reauthenticateWithCredential(user, credential);
  }
};

```

```

    await deleteDoc(doc(db, "users", uid));
    navigate("/");
    await user.delete();
  } catch (error) {
    console.error("Error deleting account:", error.message);
    if (
      error.code === "auth/invalid-credential"
    ) {
      setErrors((prevErrors) => ({
        ...prevErrors,
        curPassword: "Invalid password",
      }));
    } else if (error.code === "auth/missing-password") {
      setErrors((prevErrors) => ({
        ...prevErrors,
        curPassword: "Enter your password",
      }));
    } else if (error.code === "auth/too-many-requests") {
      setErrors((prevErrors) => ({
        ...prevErrors,
        curPassword: "Too many requests. Try again later",
      }));
    }
  }
}

```

```

return (
  <div className={classes.profileInfoContainer}>
    {!auth.currentUser.emailVerified ? (
      <div className={classes.verifyEmail}>

```

```

    <p>
      Verify your email to get all features
    </p>
  </div>
) : null}
<div className={classes.profileInfoHeader}>
  <h2>Your Profile</h2>
</div>
<form className={classes.profileInfoForm} onSubmit={handleSubmit}>
  <label className={classes.profileInfoLabel}>
    Username:
    <input
      className={classes.profileInfoInput}
      type="text"
      name="username"
      value={formData.username}
      onChange={handleChange}
      placeholder={username}
      autoComplete="username"
    />
    <p className={classes.error}>{errors.username}</p>
  </label>

  <label className={classes.profileInfoLabel}>
    Email:
    <input
      className={classes.profileInfoInput}
      type="email"
      name="email"
      value={formData.email}

```

```

    onChange={handleChange}
    placeholder={email}
    autoComplete="username"
  />
  <p className={classes.error}>{errors.email}</p>
</label>

<div className={classes.password}>
  <label className={classes.passwordInfoLabel}>
    Current Password:
    <input
      className={classes.passwordInput}
      type="password"
      name="curPassword"
      value={formData.curPassword}
      onChange={handleChange}
      placeholder={"Enter your current password"}
      autoComplete="current-password"
    />
    <p className={classes.error}>{errors.curPassword}</p>
  </label>
  <label className={classes.passwordInfoLabel}>
    New Password:
    <input
      className={classes.passwordInput}
      type="password"
      name="newPassword"
      value={formData.newPassword}
      onChange={handleChange}
      placeholder={"Enter your new password"}

```

```

        autoComplete="new-password"
    />
    <p className={classes.error}>{errors.newPassword}</p>
</label>
</div>

<p className={classes.paragraph}>
    Would you like to be notified when the exchange rate changes? Enter
    the minimum values next to the required currencies, when changing to
    which the email will be sent
</p>
<div className={classes.currBlock}>
    <label className={classes.currencyInfoLabel}>
        USD:
        <input
            className={classes.currency}
            type="text"
            name="USD"
            value={formData.currencyDiff.USD}
            onChange={handleChange}
            disabled={!user.emailVerified}
        />
        <p className={classes.errorCur}>{errors.usd}</p>
    </label>

    <label className={classes.currencyInfoLabel}>
        EUR:
        <input
            className={classes.currency}
            type="text"

```

```
        name="EUR"
        value={formData.currencyDiff.EUR}
        onChange={handleChange}
        disabled={!user.emailVerified}
      />
      <p className={classes.errorCur}>{errors.eur}</p>
    </label>
```

```
    <label className={classes.currencyInfoLabel}>
      PLN:
      <input
        className={classes.currency}
        type="text"
        name="PLN"
        value={formData.currencyDiff.PLN}
        onChange={handleChange}
        disabled={!user.emailVerified}
      />
      <p className={classes.errorCur}>{errors.pln}</p>
    </label>
```

```
    <label className={classes.currencyInfoLabel}>
      GBP:
      <input
        className={classes.currency}
        type="text"
        name="GBP"
        value={formData.currencyDiff.GBP}
        onChange={handleChange}
        disabled={!user.emailVerified}
```



```

    />
    <p className={classes.errorCur}>{errors.gbp}</p>
  </label>
</div>

<div className={classes.btnDiv}>
  <button className={classes.profileInfoSubmitBtn} type="submit">
    Save Changes
  </button>

  { showPopup && (
    <div className={classes.popup}>
      <p>Profile info successfully updated.</p>
    </div>
  )}
</div>

<div className={classes.profileInfoFooter}>
  <button
    className={classes.profileInfoSignoutBtn}
    onClick={handleSignOut}
    type="button"
  >
    Sign Out
  </button>
  <button
    className={classes.profileInfoDeleteBtn}
    onClick={handleDeleteAccount}
    type="button"
  >

```

```

        Delete account
      </button>
    </div>

    </form>
  </div>
);
};

export default ProfileInfo;

```

### **Файл Header.jsx**

```

import { Link } from "react-router-dom";
import classes from "./Header.module.css";
import logo from "../../assets/icons/logo.svg";
import { useAuth } from "../../hooks/useAuth";
import { getAuth } from "firebase/auth";
import { useState } from "react";
import { useEffect } from "react";
import { useNavigate } from "react-router";

const Header = () => {
  const user = useAuth();
  const auth = getAuth();
  const userAuth = auth.currentUser;
  const navigate = useNavigate();

  const [error, setError] = useState("");
  const [showPopup, setShowPopup] = useState(false);

```

```

useEffect(() => {
  if (showPopup) {
    const timeoutId = setTimeout(() => {
      setShowPopup(false);
    }, 5000);

    return () => clearTimeout(timeoutId);
  }
}, [showPopup]);

const handleStartExchange = (e) => {
  e.preventDefault();

  if (!userAuth.emailVerified) {
    setError("Please verify your email first!");
    setShowPopup(true);
  } else {
    setError("");
    navigate("/start-exchange");
  }
};

return (
  <header className={classes.header}>
    {showPopup && <div className={classes.popup}>{error}</div>}

    {user.isAuth ? (
      user.isAdmin ? (
        <div className={classes.leftSide}>
          <Link to="/">

```

```

        <img className={classes.logo} src={logo} />
      </Link>
      <Link to="/">Home</Link>
      <Link to="/start-exchange" onClick={handleStartExchange}>
        Start Exchange
      </Link>
      <Link to="/exchange-requests">Exchange Requests</Link>
    </div>
  ) : (
    <div className={classes.leftSide}>
      <Link to="/">
        <img className={classes.logo} src={logo} />
      </Link>
      <Link to="/">Home</Link>
      <Link to="/start-exchange" onClick={handleStartExchange}>
        Start Exchange
      </Link>
    </div>
  )
) : (
  <div className={classes.leftSide}>
    <Link to="/">
      <img className={classes.logo} src={logo} />
    </Link>
    <Link to="/">Home</Link>
  </div>
)}

{user.isAuthenticated ? (
  <div className={classes.rightSide}>

```

```

    <Link to="/profile">Profile</Link>
  </div>
): (
  <div className={classes.rightSide}>
    <Link to="/signin">Sign In</Link>
    <Link to="/signup">Sign Up</Link>
  </div>
)}
</header>
);
};

```

```
export default Header;
```

### **Файл Footer.jsx**

```
import classes from "../Footer.module.css";
```

```

const Footer = () => {
  return (
    <footer className={classes.footer}>
      <div className={classes.footerContent}>
        <div className={classes.footerSection}>
          <h3>About Us</h3>
          <p>
            CashFlow Exchange offers the ultimate in simplified interaction
            between the individual and the point of exchange. You can track the
            latest and best exchange rates, create exchange orders and receive
            notifications of rate updates
          </p>
        </div>
      </div>
    </footer>
  )
}

```

```

    <div className={classes.footerSection}>
      <h3>Contact Us</h3>
      <p>Email: cash.flow.exchange.bsn@gmail.com</p>
      <p>Phone: +38 (095) 68-34-322</p>
    </div>
  </div>

  <div className={classes.footerBottom}>
    <p>&copy; 2023 CashFlow Exchange. All rights reserved.</p>
  </div>
</footer>
);
};

```

export default Footer;

### Файл CurrencyRates.jsx

```

import { useEffect, useState } from "react";

import classes from "../CurrencyRates.module.css";
import { db } from "../../firebase";
import { doc } from "firebase/firestore";
import { getDoc } from "firebase/firestore";

import usdIcon from "../../assets/icons/US.svg";
import eurIcon from "../../assets/icons/EU.svg";
import plnIcon from "../../assets/icons/PL.svg";
import gbpIcon from "../../assets/icons/UK.svg";

```

```

const currencyIcons = {
  USD: usdIcon,
  EUR: eurIcon,
  PLN: plnIcon,
  GBP: gbpIcon,
};

const CurrencyRates = () => {
  const [rates, setRates] = useState("");

  useEffect(() => {
    const fetchRates = async () => {
      try {
        const ratesDoc = doc(db, "rates", "latestRates");
        const docSnapshot = await getDoc(ratesDoc);

        if (docSnapshot.exists()) {
          const ratesData = docSnapshot.data();
          setRates(ratesData);
        }
      } catch (error) {
        console.log(`Error fetching currency rates: ${error.message}`);
      }
    };

    fetchRates();
  }, []);

  return (
    <section className={classes.mainCon}>

```

```

<h2 className={classes.header}>Exchange rates</h2>
<table className={classes.table}>
  <thead>
    <tr className={classes.tableRow}>
      <th className={classes.tableHeader}>Currency</th>
      <th className={classes.tableHeader}>Rate</th>
    </tr>
  </thead>
  <tbody>
    {Object.entries(rates).map(([currency, rate]) => (
      <tr className={classes.tableRow} key={currency}>
        <td className={classes.tableFields}>
          <div className={classes.currencyInfo}>
            <img
              className={classes.flag}
              src={currencyIcons[currency]}
              alt={currency}
            />
            <span>{currency}</span>
          </div>
        </td>
        <td className={classes.tableFields}>{rate.toFixed(2)}</td>
      </tr>
    ))}
  </tbody>
</table>
</section>
);
};

```



```
export default CurrencyRates;
```

### **Файл CurrencyCalculator.jsx**

```
import { useEffect, useState } from "react";
import { db } from "../firebase";
import { doc, getDoc } from "firebase/firestore";
import classes from "./CurrencyCalculator.module.css";
import { serverTimestamp } from "firebase/firestore";
import { useAuth } from "../hooks/useAuth";
import { collection, addDoc } from "firebase/firestore";
import exchangeSvg from "../assets/icons/exchange.svg";

const CurrencyCalculator = ({ showCreateButton }) => {
  const [currencies, setCurrencies] = useState([]);
  const [sourceCurrency, setSourceCurrency] = useState("");
  const [targetCurrency, setTargetCurrency] = useState("");
  const [amount, setAmount] = useState("");
  const [convertedAmount, setConvertedAmount] = useState("");
  const [rates, setRates] = useState({ });

  const user = useAuth();

  useEffect(() => {
    const fetchCurrencies = async () => {
      const ratesDoc = doc(db, "rates", "latestRates");
      const docSnapshot = await getDoc(ratesDoc);

      if (docSnapshot.exists()) {
        const ratesData = docSnapshot.data();
        const availableCurrencies = Object.keys(ratesData);
```

```

    if (!availableCurrencies.includes("UAH")) {
      availableCurrencies.unshift("UAH");
      ratesData["UAH"] = 1;
    }

    setCurrencies(availableCurrencies);
    setSourceCurrency(availableCurrencies[0]);
    setTargetCurrency(availableCurrencies[1]);
    setRates(ratesData);
  }
};

fetchCurrencies();
}, []);

useEffect(() => {
  if (sourceCurrency && targetCurrency && amount !== "") {
    const sourceRate = rates[sourceCurrency];
    const targetRate = rates[targetCurrency];
    const convertedValue = ((amount * sourceRate) / targetRate).toFixed(2);
    setConvertedAmount(convertedValue);
  } else {
    setConvertedAmount("");
  }
}, [sourceCurrency, targetCurrency, amount, rates]);

const handleAmountChange = (e) => {
  const inputAmount = e.target.value;

```

```

    if (inputAmount === "" || /^[1-9]\d*\.\?\d*$/.test(inputAmount)) {
      setAmount(inputAmount);
    }
  };

```

```

const handleCurrencyChange = (e, type) => {
  const selected = e.target.value;
  if (type === "source") {
    setSourceCurrency(selected);
  } else {
    setTargetCurrency(selected);
  }
};

```

```

const handleExchangeClick = () => {
  setSourceCurrency(targetCurrency);
  setTargetCurrency(sourceCurrency);
};

```

```

const handleCreateRequest = async () => {
  if (amount !== "" && sourceCurrency !== targetCurrency) {
    try {
      const requestsCollection = collection(db, "requests");

      await addDoc(requestsCollection, {
        sourceCurrency,
        targetCurrency,
        amount: (parseFloat(amount) + parseFloat(amount / 100)).toFixed(2),
        convertedAmount: parseFloat(convertedAmount),
        userId: user.uid,

```

```

        timestamp: serverTimestamp(),
        status: "pending",
    });

    console.log("Request created successfully!");
  } catch (error) {
    console.error("Error creating request:", error.message);
  }
}
};

return (
  <div className={classes.currencyCalculator}>
    <div className={classes.calc}>
      <div className={classes.currencyBlock}>
        <select
          className={classes.currencySelectLeft}
          value={sourceCurrency}
          onChange={(e) => handleCurrencyChange(e, "source")}
        >
          {currencies.map((currency) => (
            <option key={currency} value={currency}>
              {currency}
            </option>
          ))}
        </select>
        <input
          className={classes.currencyInput}
          type="text"
          value={amount}

```

```

        onChange={handleAmountChange}
        placeholder="Enter amount"
      />
    </div>
    <img
      src={exchangeSvg}
      className={classes.exchangeBtn}
      alt="Exchange"
      onClick={handleExchangeClick}
    />
    <div className={classes.currencyBlock}>
      <select
        className={classes.currencySelectRight}
        value={targetCurrency}
        onChange={(e) => handleCurrencyChange(e, "target")}
      >
        {currencies.map((currency) => (
          <option className={classes.option} key={currency} value={currency}>
            {currency}
          </option>
        ))}
      </select>
      <input
        className={classes.currencyResult}
        type="text"
        value={convertedAmount}
        readOnly
        placeholder="You receive"
      />
    </div>

```

```

    <div className={classes.currencyBlock}>
      {showCreateButton ? (
        <button
          className={classes.createRequestButton}
          onClick={handleCreateRequest}
        ></button>
      ) : null}
    </div>
  </div>
  <div>
    <p className={classes.info}>
      Comission for exchange is 1% of the amount: {(amount / 100).toFixed(2)}
    {sourceCurrency}
    </p>
  </div>
</div>
);
};

```

```
export default CurrencyCalculator;
```

### **Файл index.js**

```

require("dotenv").config();
const functions = require("firebase-functions");
const admin = require("firebase-admin");
const axios = require("axios");
const nodemailer = require("nodemailer");
const smtpTransport = require("nodemailer-smtp-transport");

admin.initializeApp();

```

```

exports.fetchRatesAndWriteToFirestore = functions.pubsub
  .schedule("0 12 * * *")
  .timeZone("UTC")
  .onRun(async (context) => {
    try {
      const endpoint = "latest";
      const baseCurrency = "UAH";
      const response = await axios.get(
        `https://v6.exchangerate-api.com/v6/${endpoint}/${baseCurrency}`,
        {
          headers: {
            Authorization: `Bearer
${process.env.VITE_EXCHANGE_RATES_API_KEY}`,
          },
        }
      );
      const data = response.data;

      const requiredCurrencies = ["USD", "EUR", "PLN", "GBP"];

      const filteredRates = Object.fromEntries(
        Object.entries(data.conversion_rates).filter(([currency]) =>
          requiredCurrencies.includes(currency)
        )
      );

      const reciprocalRates = Object.fromEntries(
        Object.entries(filteredRates).map(([currency, rate]) => [
          currency,

```

```

    1 / rate,
  ])
);

console.log(reciprocalRates);
console.log(reciprocalRates);
console.log(reciprocalRates);

// reciprocalRates = fetchTest();

const ratesDoc = admin.firestore().collection("rates").doc("latestRates");
const docSnapshot = await ratesDoc.get();

const {
  USD = null,
  EUR = null,
  PLN = null,
  GBP = null,
} = docSnapshot.data() || {};

const usersSnapshot = await admin.firestore().collection("users").get();

const sendEmailPromises = [];

usersSnapshot.forEach(async (userDoc) => {
  const userData = userDoc.data();
  const currencyDiff = userData.currencyDiff || {};

  console.log(currencyDiff);
});

```



```

const newRates = {
  USD: reciprocalRates.USD || null,
  EUR: reciprocalRates.EUR || null,
  PLN: reciprocalRates.PLN || null,
  GBP: reciprocalRates.GBP || null,
};

const rateDifferences = {
  USD: newRates.USD - USD,
  EUR: newRates.EUR - EUR,
  PLN: newRates.PLN - PLN,
  GBP: newRates.GBP - GBP,
};

console.log(rateDifferences);

const emailMessages = Object.entries(rateDifferences)
  .filter(
    ([currency, diff]) =>
      Math.abs(diff) >= currencyDiff[currency] &&
      currencyDiff[currency] !== null
  )
  .map(([currency, diff]) => {
    return `${currency} rate changed by ${diff.toFixed(
      2
    )}\nOld rate: ${docSnapshot
      .data()
      [currency].toFixed(2)}\nNew rate: ${newRates[currency].toFixed(
        2
      )}\n`;
  });

```

```

    });

    const message = emailMessages.join("");

    console.log(message);

    if (message) {
        sendEmailPromises.push(
            sendEmailNotification(userData.email, message)
        );
    }
});

await Promise.all(sendEmailPromises);

const ratesRef = admin.firestore().collection("rates").doc("latestRates");
await ratesRef.set(reciprocalRates);

console.log("Rates fetched and written to Firestore successfully.");
return null;
} catch (error) {
    console.error("Error fetching and writing rates:", error.message);
    return null;
}
});

async function sendEmailNotification(email, message) {
    console.log(process.env.GMAIL_ADDRESS,
process.env.GMAIL_PASSWORD);
    const transporter = nodemailer.createTransport(

```

```

smtpTransport({
  host: "smtp.gmail.com",
  port: 465,
  secure: true,
  service: "gmail",
  auth: {
    user: process.env.GMAIL_ADDRESS,
    pass: process.env.GMAIL_PASSWORD,
  },
})
);

const mailOptions = {
  from: process.env.GMAIL_ADDRESS,
  to: email,
  subject: "Exchange Rate Notification",
  text: message,
};

await transporter.sendMail(mailOptions, (error, info) => {
  if (error) {
    return console.log(error);
  }
  console.log("Message sent: " + info.response);
});
}

// function fetchTest()
// {
//   return {

```

```
// USD: 38,  
// EUR: 42,  
// PLN: 9,  
// GBP: 46,  
// }  
// }
```

Факультет інформатики та обчислювальної техніки

Кафедра інформатики та програмної інженерії

“ЗАТВЕРДЖЕНО”

Керівник роботи

\_\_\_\_\_ Ілля АХАЛАДЗЕ

“ \_\_\_\_ ” \_\_\_\_\_ 2024 р.

**ВЕБ-ЗАСТОСУНОК ПІДТРИМКИ РОБОТИ ПУНКТУ ОБМІНУ ВАЛЮТ**

**Програма та методика тестування**

КП.ІП-1304.045440.04.51

“ПОГОДЖЕНО”

Керівник роботи:

\_\_\_\_\_ Ілля АХАЛАДЗЕ

Виконавець:

\_\_\_\_\_ Максим БОНДАРЕНКО

## ЗМІСТ

1	ОБ’ЄКТ ВИПРОБУВАНЬ.....	3
2	МЕТА ТЕСТУВАННЯ .....	4
3	МЕТОДИ ТЕСТУВАННЯ.....	5
4	ЗАСОБИ ТА ПОРЯДОК ТЕСТУВАННЯ.....	6

## **1 ОБ’ЄКТ ВИПРОБУВАНЬ**

Об’єктом випробування є веб-застосунок підтримки роботи пункту обміну валют “CashFlow Exchange”. Застосунок розроблено для використання в веб-середовищі. Доступ має надаватись через різні веб-браузери на їх останніх версіях (Opera, Google Chrome, Firefox, Edge), а також з різних пристроїв з різними технічними та програмними характеристиками.

## **2 МЕТА ТЕСТУВАННЯ**

Метою тестування є наступне:

- перевірка правильності роботи програмного забезпечення відповідно до функціональних вимог;
- перевірка збереження даних;
- перевірка сумісності веб-додатку з останніми версіями сучасних браузерів (Chrome, Opera, Firefox, Edge);
- перевірка сумісності застосунку з різними операційними системами (Windows);
- знаходження проблем, помилок і недоліків з метою їх усунення;
- перевірка зручності графічного інтерфейсу.



### **3 МЕТОДИ ТЕСТУВАННЯ**

Для тестування програмного забезпечення використовуються такі методи:

- статичне тестування – перевіряється програма разом з усією документацією, яка аналізується на предмет дотримання стандартів програмування;
- функціональне тестування – полягає у перевірці відповідності реальної поведінки програмного забезпечення очікуваній;
- мануальне тестування – тестування без використання автоматизації, тест-кейси пише особа, що тестує програмне забезпечення;

## **4 ЗАСОБИ ТА ПОРЯДОК ТЕСТУВАННЯ**

Тестування виконується мануально з використанням наскрізного тестування, з метою знаходження помилок та недоліків як у функціональній частині програмного забезпечення так і в зручності користування. Для того, щоб перевірити працездатність та відмовостійкість застосунку, необхідно провести наступні тестування:

- тестування на мобільних пристроях з різною роздільною здатністю екрану;
- тестування на виведення повідомлень про помилку, коли це необхідно;
- тестування інтерфейсу користувача;
- тестування зручності використання;
- тестування на відповідність функціональним вимогам.

Факультет інформатики та обчислювальної техніки

Кафедра інформатики та програмної інженерії

“ЗАТВЕРДЖЕНО”

Керівник роботи

\_\_\_\_\_ Ілля Ахаладзе

“ \_\_\_\_ ” \_\_\_\_\_ 2024 р.

**ВЕБ-ЗАСТОСУНОК ПІДТРИМКИ РОБОТИ ПУНКТУ ОБМІНУ ВАЛЮТ**

**Керівництво користувача**

КПІ.ІП-1304.045440.05.34

“ПОГОДЖЕНО”

Керівник роботи:

\_\_\_\_\_ Ілля АХАЛАДЗЕ

Виконавець:

\_\_\_\_\_ Максим БОНДАРЕНКО

## ЗМІСТ

1	ПРИЗНАЧЕННЯ ПРОГРАМИ .....	3
2	ПІДГОТОВКА ДО РОБОТИ З ПРОГРАМНИМ ЗАБЕЗПЕЧЕННЯМ.....	4
2.1	Системні вимоги для коректної роботи.....	4
2.2	Завантаження застосунку .....	4
2.3	Перевірка коректної роботи.....	4
3	ВИКОНАННЯ ПРОГРАМИ .....	5

## **1 ПРИЗНАЧЕННЯ ПРОГРАМИ**

“CashFlow Exchange” – це веб-застосунок для підтримки роботи фізичного пункту обміну валют, який оптимізує комунікацію між клієнтом та фізичним пунктом обміну. Застосунок дозволяє відстежувати актуальні курси, створювати запити на обмін валют, розраховувати кількість грошей в калькуляторі, отримувати сповіщення про зміни курсів валют, надає мінімалістичний, простий, інтуїтивний інтерфейс.

## **2 ПІДГОТОВКА ДО РОБОТИ З ПРОГРАМНИМ ЗАБЕЗПЕЧЕННЯМ**

### **2.1 Системні вимоги для коректної роботи**

Мінімальна конфігурація технічних засобів:

- тип процесору: Intel Core i3;
- об'єм ОЗП: 4 Гб;
- підключення до мережі Інтернет зі швидкістю від 20 мегабіт;

Рекомендована конфігурація технічних засобів:

- тип процесору: Intel Core i5;
- об'єм ОЗП: 8 Гб;
- підключення до мережі Інтернет зі швидкістю від 100 мегабіт;

### **2.2 Завантаження застосунку**

Сайт доступний за посиланням: <https://cash-flow-exchange.vercel.app/>

### **2.3 Перевірка коректної роботи**

Для перевірки коректності розгортання ПЗ потрібно:

- перейти на веб-сайт за посиланням вище;
- переконайтесь в завантаженні застосунку і коректній роботі його функціоналу;

Виконання кроків вище дозволяє переконатись в коректності роботи застосунку.

### 3 ВИКОНАННЯ ПРОГРАМИ

При першому заході на сайт перед гостем постає головна сторінка (рисунки 3.1), яка містить інформацію про актуальні курси валют, які оновлюються щоденно та калькулятор валют, в якому можна прорахувати обмін та подивитись комісію за потенційний обмін, обравши потрібні валюти зі списку та ввівши потрібну суму в поле “Enter amount”.

Currency	Rate
PLN	9.54
GBP	48.12
EUR	41.68
USD	38.11

Рисунок 3.1 – Головна сторінка

В правому верхньому куті неавторизований користувач має кнопки для реєстрації та авторизації, тож для доступу до повного функціоналу потрібно пройти процедуру реєстрації, натиснувши на кнопку “Sign Up”. Після відкриття вікна реєстрації (рисунки 3.2). У вікно реєстрації потрібно ввести валідні дані, за якими можна зареєструвати акаунт та натиснути кнопку “Sign Up”.

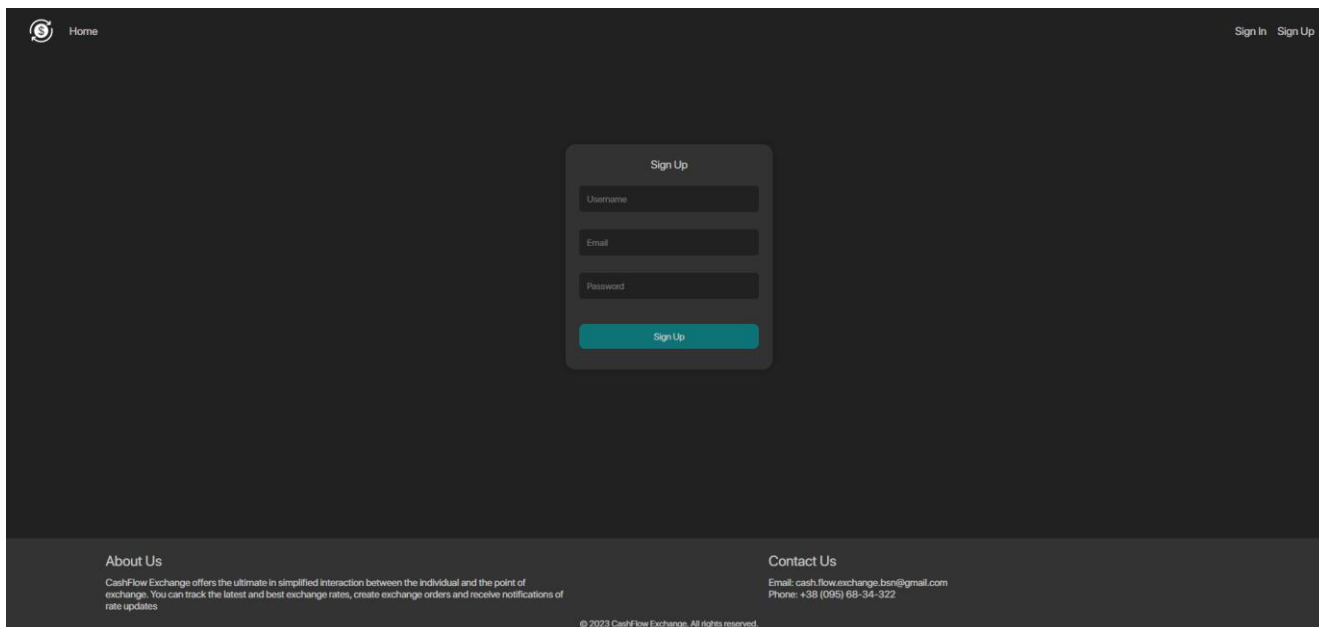


Рисунок 3.2 – Вікно реєстрації

Після реєстрації, користувача автоматично авторизує в акаунт і йому відкриваються нові можливості (рисунок 3.3), але потрібно підтвердити електронну адресу для отримання всіх можливостей.



Рисунок 3.3 – «шапка», яка містить доступ до нових можливостей

Авторизований користувач з верифікованою поштовою адресою має можливість перейти до сторінки створення запиту на обмін (рисунок 3.4).

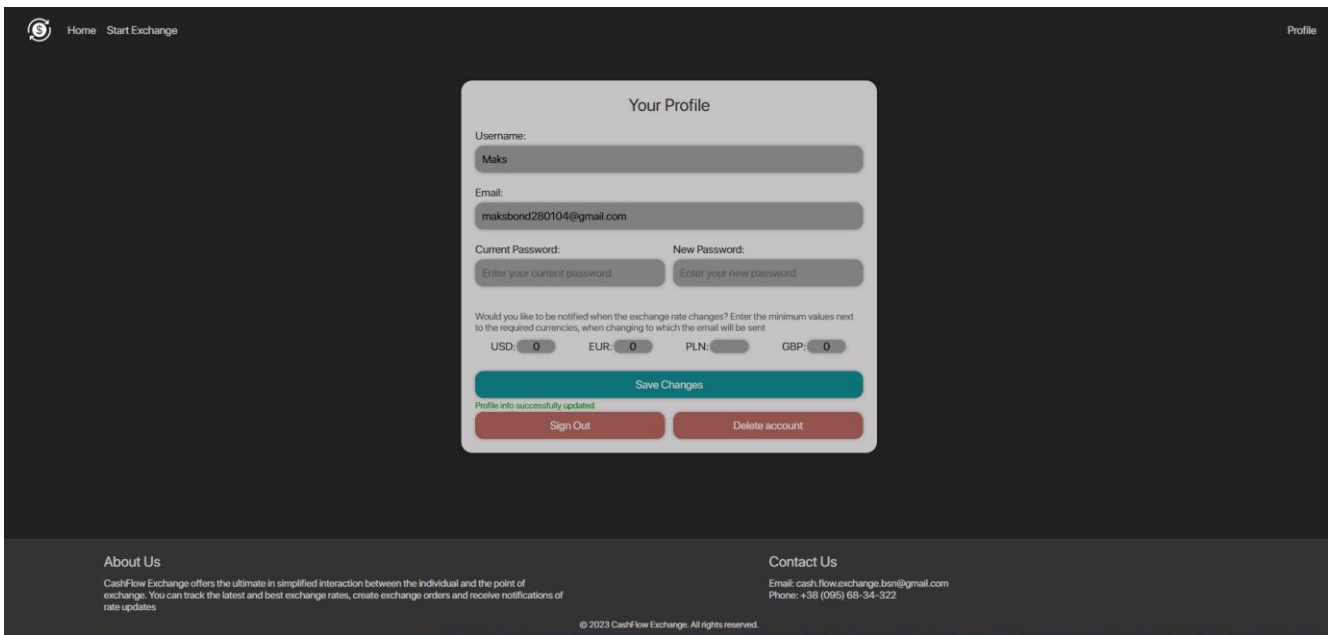


Рисунок 3.4 – Сторінка створення запиту на обмін



На даній сторінці користувач може створити запит на обмін, скористувавшись таким же калькулятором, як і на головній сторінці та натиснувши кнопку підтвердження, тоді новий запит з'явиться у користувача в історії його запитів зі статусом “pending”. Користувач може змінити статус свого запиту з “pending” або “take away” на “rejected”, якщо хоче відмовитись від створеного запиту.

В правому верхньому куті користувач має можливість перейти на сторінку налаштування профілю (рисунок 3.5).



The screenshot displays the 'Your Profile' page of the CashFlow Exchange application. The page has a dark background with a light gray form in the center. The form contains the following elements:

- Username:** A text input field with the value 'Maks'.
- Email:** A text input field with the value 'maksbond280104@gmail.com'.
- Current Password:** A text input field with the placeholder 'Enter your current password'.
- New Password:** A text input field with the placeholder 'Enter your new password'.
- Notifications:** A section with the text 'Would you like to be notified when the exchange rate changes? Enter the minimum values next to the required currencies, when changing to which the email will be sent'.
- Currency Selection:** Four radio buttons for 'USD', 'EUR', 'PLN', and 'GBP', each followed by a '0'.
- Buttons:** A green 'Save Changes' button, a red 'Sign Out' button, and a red 'Delete account' button.

At the bottom of the page, there is a footer with 'About Us' and 'Contact Us' information, including the email 'cash.flow.exchange.bon@gmail.com' and the phone number '+38 (096) 68-34-322'.

Рисунок 3.5 – сторінка профілю користувача

На сторінці профілю користувач має можливість змінювати персональні дані (ім'я користувача, пароль, пошту), для зміни пароля та пошти потрібно підтвердити свій актуальний пароль. Також користувач з верифікованою поштою може виставити межові значення для кожної валюти, при зміні курсу на які, користувач буде отримувати повідомлення на поштову скриньку з інформацією про обрані валюти. Також користувач може вийти з облікового запису або видалити власний обліковий запис натиснувши червону кнопку “Sign Out” внизу або “Delete account” відповідно. Тоді його буде переадресовано на домашню сторінку, а для повторної аутентифікації потрібно авторизуватись натиснувши кнопку “Sign In” (рисунок 3.6).

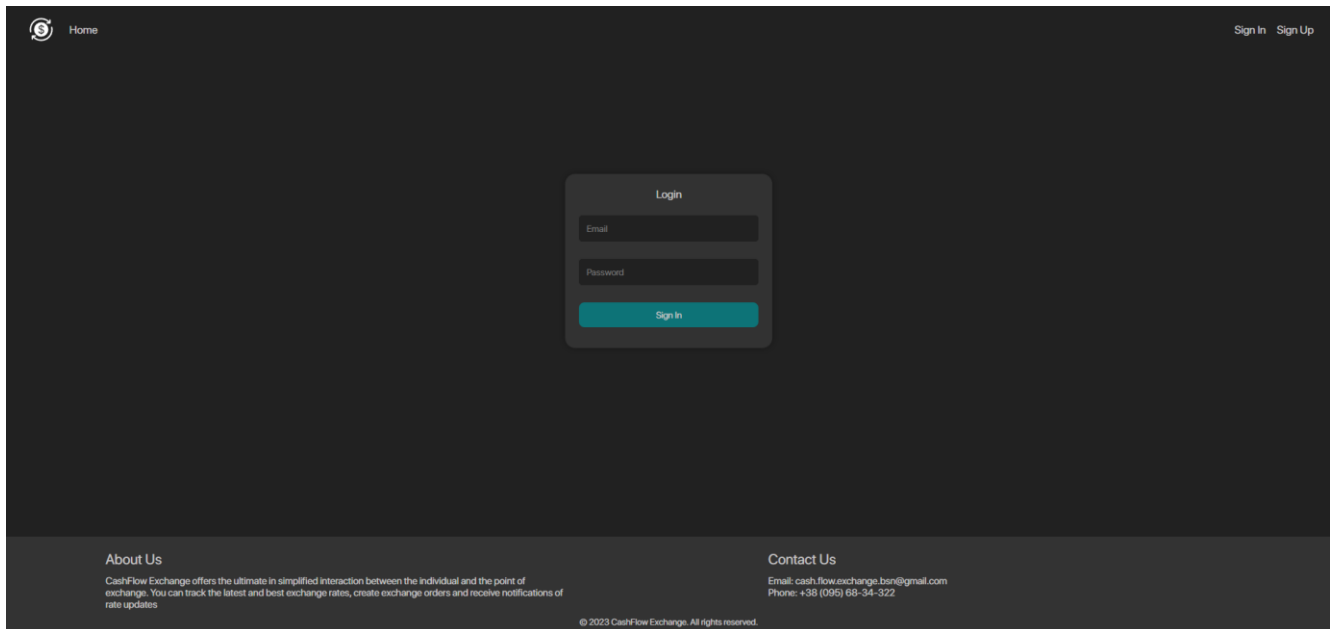


Рисунок 3.6 – Сторінка авторизації

На сторінці авторизації користувач повинен ввести дані від свого дійсного облікового запису, щоб авторизуватись в нього та мати всі ті ж можливості, що раніше.

Факультет інформатики та обчислювальної техніки

Кафедра інформатики та програмної інженерії

“ЗАТВЕРДЖЕНО”

Керівник роботи

\_\_\_\_\_ Ілля Ахаладзе

“ \_\_\_\_ ” \_\_\_\_\_ 2024 р.

**ВЕБ-ЗАСТОСУНОК ПІДТРИМКИ РОБОТИ ПУНКТУ ОБМІНУ ВАЛЮТ**

**Керівництво адміністратора**

КП.ІП-1304.045440.05.34

“ПОГОДЖЕНО”

Керівник роботи:

\_\_\_\_\_ Ілля АХАЛАДЗЕ

Виконавець:

\_\_\_\_\_ Максим БОНДАРЕНКО

## ЗМІСТ

1	ПРИЗНАЧЕННЯ ПРОГРАМИ .....	3
2	ПІДГОТОВКА ДО РОБОТИ З ПРОГРАМНИМ ЗАБЕЗПЕЧЕННЯМ.....	4
2.1	Системні вимоги для коректної роботи.....	4
2.2	Завантаження застосунку .....	4
2.3	Перевірка коректної роботи.....	4
3	ВИКОНАННЯ ПРОГРАМИ .....	5

## **1 ПРИЗНАЧЕННЯ ПРОГРАМИ**

“CashFlow Exchange” – це веб-застосунок для підтримки роботи фізичного пункту обміну валют, який оптимізує комунікацію між клієнтом та фізичним пунктом обміну. Застосунок дозволяє відстежувати актуальні курси, створювати запити на обмін валют, розраховувати кількість грошей в калькуляторі, отримувати сповіщення про зміни курсів валют, надає мінімалістичний, простий, інтуїтивний інтерфейс.

## **2 ПІДГОТОВКА ДО РОБОТИ З ПРОГРАМНИМ ЗАБЕЗПЕЧЕННЯМ**

### **2.1 Системні вимоги для коректної роботи**

Мінімальна конфігурація технічних засобів:

- тип процесору: Intel Core i3;
- об'єм ОЗП: 4 Гб;
- підключення до мережі Інтернет зі швидкістю від 20 мегабіт;

Рекомендована конфігурація технічних засобів:

- тип процесору: Intel Core i5;
- об'єм ОЗП: 8 Гб;
- підключення до мережі Інтернет зі швидкістю від 100 мегабіт;

### **2.2 Завантаження застосунку**

Сайт доступний за посиланням: <https://cash-flow-exchange.vercel.app/>

### **2.3 Перевірка коректної роботи**

Для перевірки коректності розгортання ПЗ потрібно:

- перейти на веб-сайт за посиланням вище;
- переконайтесь в завантаженні застосунку і коректній роботі його функціоналу;

Виконання кроків вище дозволяє переконатись в коректності роботи застосунку.

### 3 ВИКОНАННЯ ПРОГРАМИ

При першому заході на сайт перед гостем постає головна сторінка (рисунок 3.1), яка містить інформацію про актуальні курси валют, які оновлюються щоденно та калькулятор валют, в якому можна прорахувати обмін та подивитись комісію за потенційний обмін, обравши потрібні валюти зі списку та ввівши потрібну суму в поле “Enter amount”.

Home Sign In Sign Up

## Welcome!

Calculate the amount you want to change

UAH Enter amount ⇄ PLN You receive

Commission for exchange is 1% of the amount: 0.00 UAH

### Exchange rates

Currency	Rate
PLN	9.54
GBP	48.12
EUR	41.68
USD	38.11

**About Us**  
CashFlow Exchange offers the ultimate in simplified interaction between the individual and the point of exchange. You can track the latest and best exchange rates, create exchange orders and receive notifications of rate updates.

**Contact Us**  
Email: cash.flow.exchange.hs@gmail.com  
Phone: +38 (098) 68-34-322

© 2023 CashFlow Exchange. All rights reserved.

Рисунок 3.1 – Головна сторінка

В правому верхньому куті неавторизований користувач має кнопки для реєстрації та авторизації, тож для доступу до повного функціоналу потрібно пройти процедуру реєстрації, натиснувши на кнопку “Sign Up”. Після відкриття вікна реєстрації потрібно ввести валідні дані, за якими можна зареєструвати акаунт та натиснути кнопку “Sign Up”.

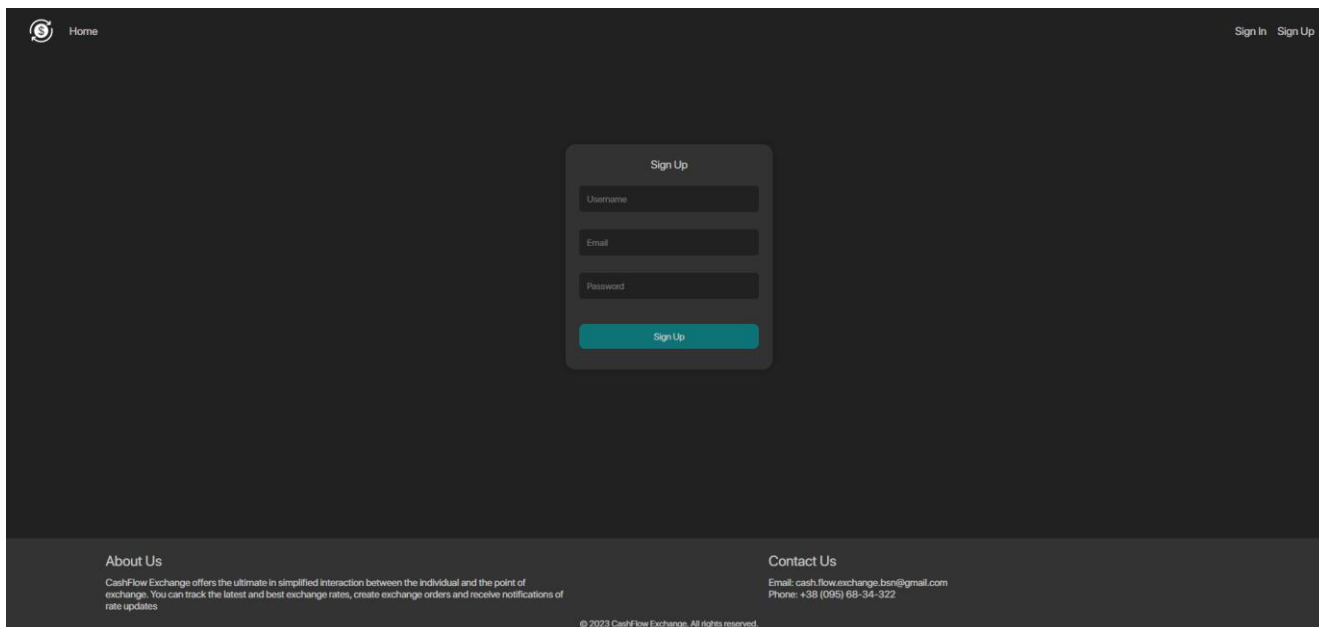


Рисунок 3.2 – Вікно реєстрації

Після реєстрації, користувача автоматично авторизує в акаунт і йому відкриваються нові можливості (рисунок 3.3), але потрібно підтвердити електронну адресу для отримання всіх можливостей.



Рисунок 3.3 – «шапка», яка містить доступ до нових можливостей

Авторизований користувач з верифікованою поштовою адресою має можливість перейти до сторінки створення запиту на обмін (рисунок 3.4).

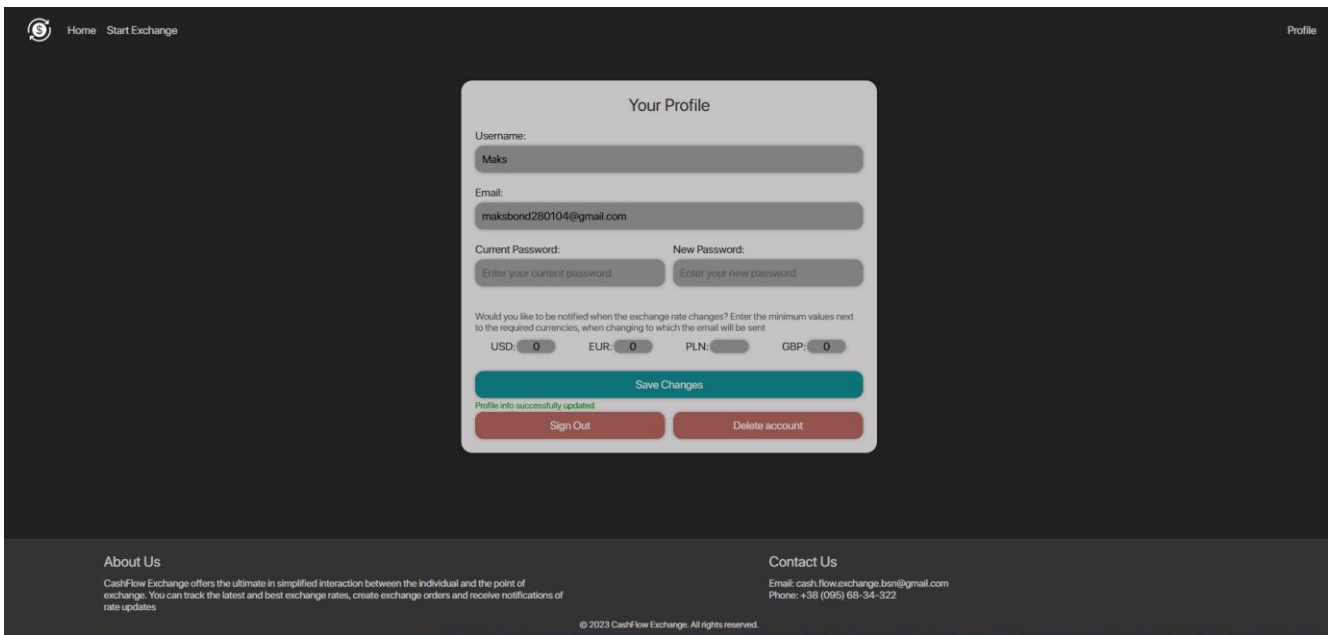


Рисунок 3.4 – Сторінка створення запиту на обмін



На даній сторінці користувач може створити запит на обмін, скористувавшись таким же калькулятором, як і на головній сторінці та натиснувши кнопку підтвердження, тоді новий запит з'явиться у користувача в історії його запитів зі статусом “pending”. Користувач може змінити статус свого запиту з “pending” або “take away” на “rejected”, якщо хоче відмовитись від створеного запиту.

В правому верхньому куті користувач має можливість перейти на сторінку налаштування профілю (рисунок 3.5).



The screenshot displays the 'Your Profile' page of the CashFlow Exchange application. The page has a dark background with a light gray form in the center. The form contains the following elements:

- Username:** A text input field with the value 'Maks'.
- Email:** A text input field with the value 'maksbond280104@gmail.com'.
- Current Password:** A text input field with the placeholder 'Enter your current password'.
- New Password:** A text input field with the placeholder 'Enter your new password'.
- Notifications:** A section with the text 'Would you like to be notified when the exchange rate changes? Enter the minimum values next to the required currencies, when changing to which the email will be sent'.
- Currency Selection:** Four radio buttons for USD, EUR, PLN, and GBP, each with a value of 0.
- Save Changes:** A green button.
- Sign Out:** A red button.
- Delete account:** A red button.

At the bottom of the page, there is a footer with the following information:

- About Us:** CashFlow Exchange offers the ultimate in simplified interaction between the individual and the point of exchange. You can track the latest and best exchange rates, create exchange orders and receive notifications of rate updates.
- Contact Us:** Email: cash.flow.exchange.bon@gmail.com, Phone: +38 (096) 68-34-322.
- Copyright:** © 2023 CashFlow Exchange. All rights reserved.

Рисунок 3.5 – сторінка профілю користувача

На сторінці профілю користувач має можливість змінювати персональні дані (ім'я користувача, пароль, пошту), для зміни пароля та пошти потрібно підтвердити свій актуальний пароль. Також користувач з верифікованою поштою може виставити межові значення для кожної валюти, при зміні курсу на які, користувач буде отримувати повідомлення на поштову скриньку з інформацією про обрані валюти. Також користувач може вийти з облікового запису або видалити власний обліковий запис натиснувши червону кнопку “Sign Out” внизу або “Delete account” відповідно. Тоді його буде переадресовано на домашню сторінку, а для повторної аутентифікації потрібно авторизуватись натиснувши кнопку “Sign In” (рисунок 3.6).

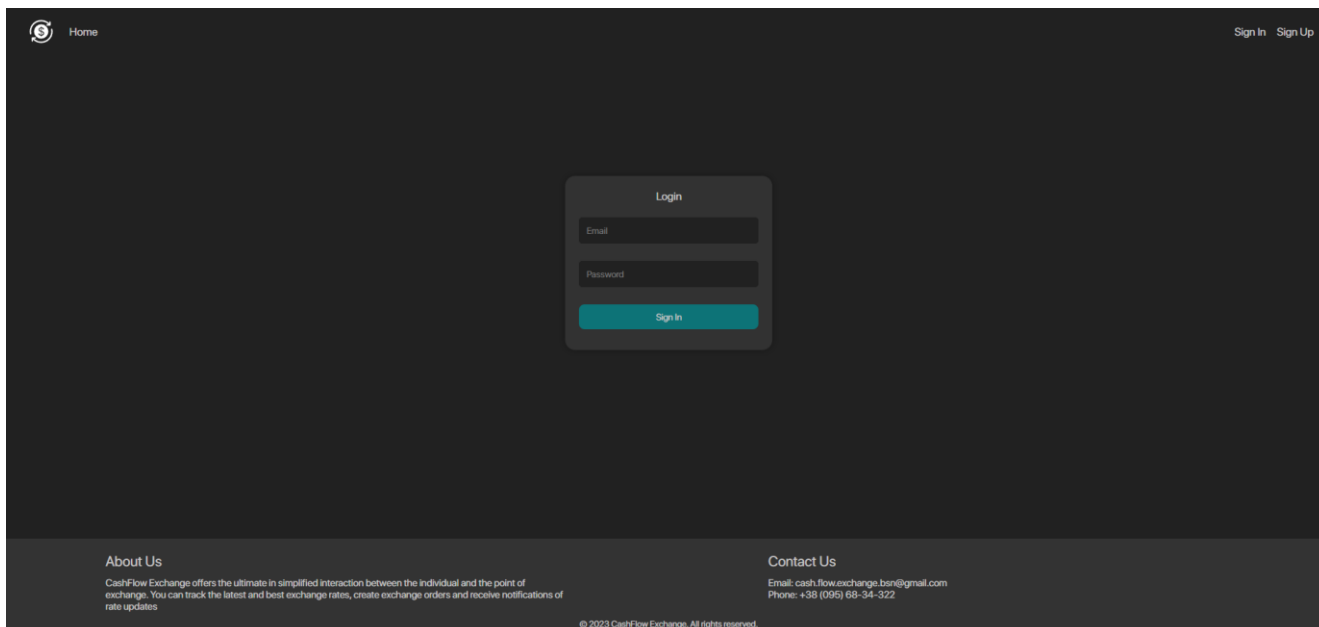


Рисунок 3.6 – Сторінка авторизації

На сторінці авторизації користувач повинен ввести дані від свого дійсного облікового запису, щоб авторизуватись в нього та мати всі ті ж можливості, що раніше.

Якщо користувач володіє привілеями адміністратора, йому відкривається нові можливості. В «шапці» у нього доступна сторінка “Exchange Requests” (рисунок 3.7)



Рисунок 3.7 – Сторінка керування запитів на обмін для адміністратора

Адміністратор може переглядати список з усіма створеними заявками на обмін та змінювати їх статус з “pending” або “take review” на “take review”,

“completed” або “rejected”, тим самим даючи іншим користувачам сигнал, що вони можуть приходити у фізичний пункт та проводити обмін валют, або їх запит відхилено.

[illegible]