Київський національний університет імені Тараса Шевченка
Факультет інформаційних технологій
Кафедра програмних систем і технологій

УДК 004.9

На правах рукопису

ВИПУСКНА КВАЛІФІКАЦІЙНА БАКАЛАВРСЬКА РОБОТА

Тема: “ Розробка SaaS для менеджменту інвестицій”

Спеціальність – 121 “Інженерія програмного забезпечення”

ПОЯСНЮВАЛЬНА ЗАПИСКА

Студент
ІПЗ-41_____ /Володимир ТКАЧ/
Науковий керівник
к.ф.м.н.,доц._____/ Сергій ПОЛЯКОВ/
Консультант
з питань нормоконтролю
фахівець_____/ Тамара ЧАПОВСЬКА/

Допускається до захисту
Завідувач кафедри
д.т.н.,доц._____/Олексій БИЧКОВ/

Київ – 2021

Рішенням Екзаменаційної комісії

випускна робота студента

Володимира ТКАЧА

захищена з оцінкою

_____

Голова екзаменаційної комісії

професор, д.т.н. Андрій БОНДАРЧУК

Додаток Г

Форма завдання на випускну кваліфікаційну бакалаврську роботу

Київський національний університет імені Тараса Шевченка
Факультет інформаційних технологій
Кафедра програмних систем і технологій
Спеціальність 121 “Інженерія програмного забезпечення”

ЗАТВЕРДЖУЮ:
Завідувач кафедри програмнихсистем і технологій
_____ (О.С.Бичков)
„___”_____20___р.

ЗАВДАННЯ
НА ВИПУСКНУ КВАЛІФІКАЦІЙНУ БАКАЛАВРСЬКУ РОБОТУ
СТУДЕНТУ
Ткачу Володимиру Володимировичу
(прізвище, ім’я, по батькові)

1. Тема випускної кваліфікаційної бакалаврської роботи
“Розробка SaaS для менеджменту інвестицій”
керівник проекту (роботи) Поляков Сергій Анатолійович, к.ф.-м.н.
затверджена наказом вищого навчального закладу від „__”_____20__р. №_____

2. Строк здачі студентом закінченої роботи _____

3. Вихідні дані до роботи Теоретичні концепції та формальні моделі побудови та функціонування інформаційних та програмних технологій певного класу

4. Зміст пояснювальної записки (перелік питань, що їх належить розробити)
1. Аналіз предметної області
2. Аналіз призначення й мета створення системи
2. Постановка задачі та методи дослідження
3. Моделювання системи
4. Розробка системи
5. Тестування системи

5. Перелік графічного матеріалу (з точним забезпеченням обов’язкових креслень)
1. Інвест менеджмент – загальні поняття (рис. 1.1, ст. 13)
2. Investing.com – трекінг одного із індексів (рис. 1.2, ст. 15)
3. портфель створений власноруч. Можливо додати тільки акції (рис. 1.3, ст. 16)
4. The Wall Street Journal – основна сторінка (рис. 1.4, ст. 17)
5. The Wall Street Journal – сторінка трекінга акцій (рис. 1.5, ст. 17)
6. Apple Inc. chart на сайті. Має фундаментальну дату але не має можливості додати це у свій портфель. (рис. 1.6, ст. 18)
7. фундаментальні дані про компанію – базовий аналіз компанії без деталей (рис. 1.7, ст. 19)

8. рейтинг мов програмування 2021 за DOU.UA (рис. 1.8, ст. 23)

9. дизайну вікна створення нового аккаунту (рис. 3.1, ст. 30)

10. Схематичне зображення домашнього екрану (рис. 3.2, ст. 30)

11. вікно створення позицій (рис. 3.3, ст. 31)

Форма завдання на випускну кваліфікаційну бакалаврську роботу (на звороті першого аркуша)

6. Консультанти з роботи із зазначенням розділів роботи, що їх стосуються

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв
1	Сергій ПОЛЯКОВ	22.02.21	23.02.21
2	Сергій ПОЛЯКОВ	08.03.21	09.03.21
3	Сергій ПОЛЯКОВ	15.03.21	15.03.21
4	Сергій ПОЛЯКОВ	08.04.21	09.04.21
5	Сергій ПОЛЯКОВ	29.04.21	29.04.21
6	Сергій ПОЛЯКОВ	07.05.21	07.05.21

7. Дата видачі завдання _____

Керівник _____ _____
(підпис) (розшифровка підпису)

Завдання прийняв до виконання _____ _____
(підпис) (розшифровка підпису)

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назви етапів бакалаврської роботи	Строк виконання етапів роботи	Примітка
1	Дослідження предметної області	22.02.21	виконано
2	Аналіз функціональних вимог	08.03.21	виконано
3	Аналіз нефункціональних вимог	15.03.21	виконано
4	Розробка алгоритмічної моделі	08.04.21	виконано
5	Розробка мобільного додатку	29.04.21	виконано
6	Тестування	07.05.21	виконано
7	Затвердження пояснювальної записки роботи завідувачем кафедри	??.06.21	виконано

Студент – бакалавр _____
(підпис) (розшифровка підпису)

Керівник роботи _____
(підпис) (розшифровка підпису)

АНОТАЦІЯ (на трьох мовах)

Випускна кваліфікаційна бакалаврська робота: 66 с., 22 рис., 5 табл., 6 джерела.

Тема: Розробка SaaS для менеджменту інвестицій

Об'єкт дослідження: інвестиційна діяльність, web-застосунки

Мета роботи: Аналіз предметної області, цільової аудиторії проекту та проектування програмного забезпечення.

Предмет дослідження: фінанси та інвестиції, менеджмент інвестицій, керування портфелем.

Результати дослідження: Досліджено теорію управління портфелем інструментів та застосування концепції на практиці. Розроблено рішення для використання системи.

Висновок: В результаті досліджень було отримано дані, щодо цільової аудиторії системи, теорії для розробки програмного забезпечення та використання на практиці.

ANNOTATION (in three languages)

Final qualifying bachelor's thesis: 66 p., 22 pics., 5 tables., 6 sources.

Topic: SaaS development for investment management

Object of research: investment activities, web-applications.

Purpose: Analysis of the subject area, target audience of the project and software design.

Subject of research: finance and investment, investment management, portfolio management.

Results of the research: The theory of tool portfolio management and application of the concept in practice is investigated. Solutions for using the system have been developed.

Conclusion: As a result of the research, data were obtained on the target audience of the system, theory for software development and use in practice.

АННОТАЦИЯ (на трех языках)

Выпускная квалификационная бакалаврская работа: 66 с., 22 рис., 6 табл., 5 источники.

Тема: Разработка SaaS для менеджмента инвестиций

Объект исследования: инвестиционная деятельность, web-приложения.

Цель работы: Анализ предметной области, целевой аудитории проекта и проектирования программного обеспечения.

Предмет исследования: финансы и инвестиции, менеджмент инвестиций, управления портфелем.

Результаты исследования: Исследована теорию управления портфелем инструментов и применение концепции на практике. Разработано решение для использования системы.

Заключение: В результате исследований было получено данные, по целевой аудитории системы, теории для разработки программного обеспечения и использования на практике.

Зміст

Завдання на дипломну роботу

В даному дипломному проекті була поставлена задача розробити SaaS для менеджменту інвестицій.

Метою розробки даної системи є полегшення менеджменту власних фінансів користувачів, а саме трекінг стану портфеля. Умови роботи даного SaaS – робоча веб сторінка для користувачів, на якому можливо побачити необхідні дані. Системні вимоги для роботи системи; для комп'ютера - наявність Web-browser, краще Google-chrome; так само як і для користування на мобільному телефоні.

Дипломна робота містить аналіз предметної області, аналіз функціональних та не функціональних вимог. Було схематично розроблено прототип додатку та користувацький інтерфейс. За допомогою мокапів було розроблено дизайн веб сторінки. Готовий продукт було протестовано, тільки Desktop версия була перевірена за допомогою браузера від Google. Мобільна адаптація не перевірялась.

Пояснювальна записка складається зі вступу, 5 розділів, висновків, списку використаних джерел із 14 найменувань, додатків. Загальний обсяг 82 сторінок, у тому числі 52 сторінок основного тексту, 2 сторінки списку використаних джерел, 27 сторінок додатків.

Вступ

В сучасному світі все частіше встає питання фінансової незалежності. Все більше людей сьогодні інтересуються інвестиціями з метою отримання пасивного доходу, котрий в ідеалі, навіть буде перевищувати розходи суб’єкта та зможе допомогти побудувати безбідне життя, або навіть розбагатіти, вклавшись в якогось нового єдиноріг.

Представлена мною тема являється дуже актуальною в Сполучених Штатах вже на протязі багатьох років, як і в Європі.

В нашому регіоні, нажаль, ця тема не так широко популярна. Люди чують головну думку: о формуванні власного капіталу, «подушки» безпеки та ін. Але дуже мало людей займаються цим серьезно.

Коли я почав інтересуватися цією темою, я швидко помітив що на ринку не так багато систем, де можна зробити власний портфель, наприклад акцій. А системи де

можна було б трекати інші інструменти: нерухомість: квартири, зем. ділянки; облігації, крипта, тощо, можна сказати немає.

Метою проекту є створення такої системи, на сам перед для власного користування: як система котра буде економити час на підрахунок всього в Excel: аналіз стану портфеля та диверсифікацію, тощо.

В ході проходження робіт було встановлено і зафіксовано: мета проекту, завдання, які буде виконувати додаток, Вибір технологічних методів, за якими буде здійснюватися розробка і планування робіт – обрані методи планування за методом АБС. В рамках моделювання додатку були проаналізовані цільова аудиторія, сценарій взаємодії клієнт-додаток. Схематично побудований прототип і структура проходження основних екранів. Описує архітектуру додатку. Наступним етапом була фаза впровадження на рівні програмного забезпечення, де була розглянута основна функціональність програми. Після створення програми була проведена фаза тестування, яка перевірила всі можливі взаємодії користувача з програмним продуктом.

1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Актуальність проблеми

Investment Management – управління інвестиціями відноситься до управління фінансовими активами та іншими інвестиціями, а не тільки до їх купівлі та продажу. Управління включає розробку короткостроковій або довгостроковій стратегії придбання та продажу портфельних активів. Сюди також можуть входити банківські, бюджетні і податкові пільги і збори.

Цей термін найчастіше відноситься до управління активами в інвестиційному портфелі і торгівлі ними для досягнення конкретної інвестиційної мети. Управління інвестиціями також відомо як управління капіталом, управління портфелем або управління капіталом.

Професійне управління інвестиціями направлено на досягнення конкретних інвестиційних цілей в інтересах клієнтів, за гроші яких вони несуть відповідальність. Ці клієнти можуть бути індивідуальними інвесторами або інституційними інвесторами, такими як пенсійні фонди, пенсійні плани, уряду, освітні установи та страхові компанії.

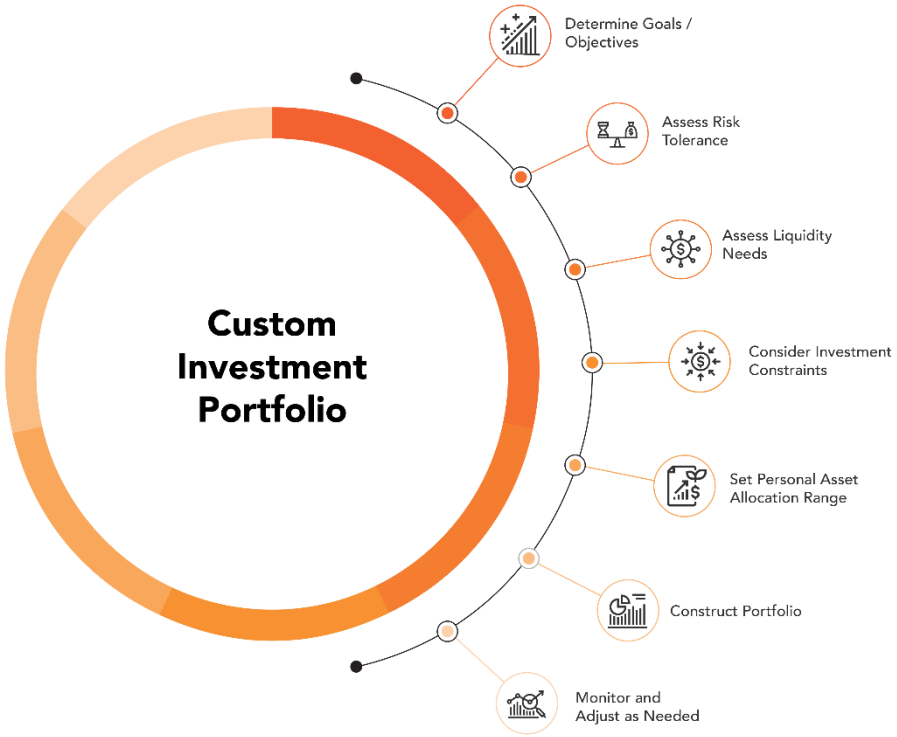


Рисунок 1.1 – Інвест менеджмент – загальні поняття

Послуги з управління інвестиціями включають розміщення активів, аналіз фінансової звітності, вибір акцій, моніторинг існуючих інвестицій, а також портфельную стратегію і реалізацію. Управління інвестиціями може також включати послуги фінансового планування і консультування, не тільки контролюючи портфель клієнта, а й координуючи його з іншими активами і життєвими цілями. Професійні менеджери мають справу з безліччю різних цінних паперів і фінансових активів, включаючи облігації, акції, товари і нерухомість. Менеджер також може управляти реальними активами, такими як дорогоцінні метали, товари і твори мистецтва. Менеджери можуть допомогти узгодити

інвестиції відповідно до пенсійного та майновим плануванням, а також розподілом активів.

Хоча індустрія управління інвестиціями може забезпечити прибуток, є також ключові проблеми, пов'язані з управлінням такою фірмою. Доходи фірм з управління інвестиціями безпосередньо пов'язані з поведінкою ринку. Ця пряма зв'язок означає, що прибуток компанії залежить від ринкової оцінки. Істотне зниження цін на активи може викликати зниження доходів фірми, особливо якщо зниження ціни значно в порівнянні з постійними і стабільними операційними витратами компанії. Крім того, клієнти можуть бути нетерплячими в важкі часи і на ведмежих ринках, і навіть прибутковість фонду вище середнього може бути не в змозі підтримувати портфель клієнта.

Asset allocation – Розподіл активів - це інвестиційна стратегія, яка проводиться з урахуванням балансу ризику, а інший спосіб розподіляє портфель активів пропорційно цілям інвестуваної особи. Три основні класи активів - акції, з’являються з появою основних та сучасних активів та їх еквівалентів - завдають шкоди грошам і стають різними, тому інших немає.

Тут йдеться про розподіл активів - немає простої формули, яку можна знайти у правильному розподілі активів для однієї особи. Переважна більшість фінансових фахівців сходяться на думці, що розподіл активів є одним з найкращих продажів інвесторів. Іншими словами, наприклад, бум і ціна є другим по відношенню до розподілу активів у формі дій, зобов'язань, оборотних активів та еквівалента

Інвестори можуть використовувати різні розподіли активів для різних цілей. Наприклад, кожен, хто оголошує про нові дні автомобілів наступного року, може інвестувати свій фонд заощаджень автомобілів у дуже консервативне засідання компакт-дисків для пошуку обладнання Інші окремі пенсіонери, якщо їм не потрібно виходити на пенсію, тому інвестуйте значну частину любовної пенсії особи (IPA) у запасах, які мають трохи більшу толерантність до ризику, знову відіграє важливу роль. Той, хто не задоволений інвестиціями в можливий крок,

змінить день і доповнить його більш консервативним розподілом, незалежно від горизонту в довгостроковій перспективі.

С того що було зазначено вище, слідує що після того як деякі позиції появляються у нашому портфелі, за ними слід наблюдати адже можливо буде потрібно вийти із них. Але не за всіма позиціями можливо наблюдати, деякі просто не можуть бути знайденими.

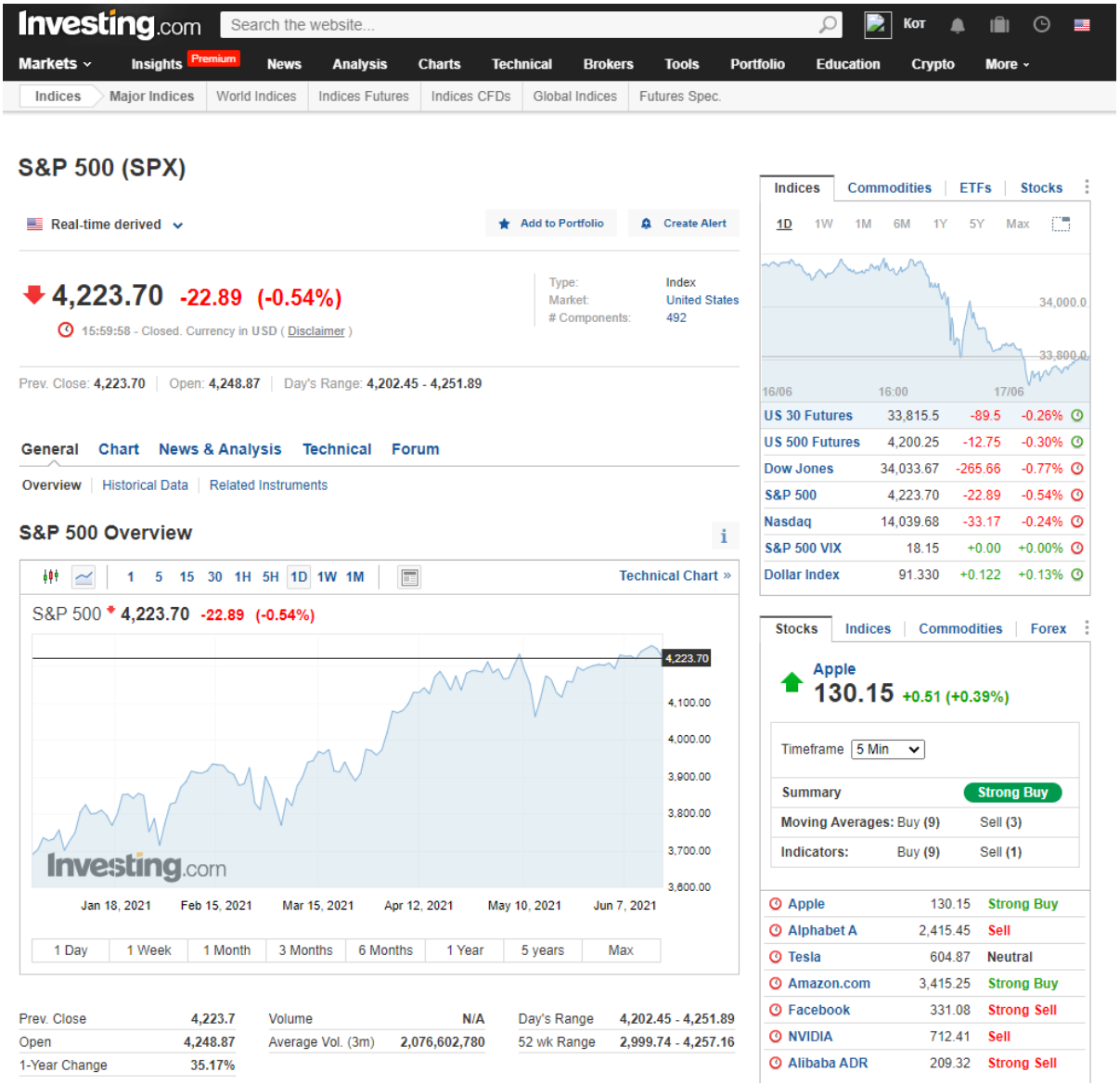


Рисунок 1.2 - Investing.com – трекінг одного із індексів – S&P 500

1.2 Аналіз ринку. Конкуренти

Під час планування проекту було розглянуто декілька популярних та подібних сайтів з метою аналізу функціоналу продукту, користувацького інтерфейсу, виокремлення сильних факторів та знаходження недоліків.

Схожі додатки бувають з наявністю ринку акцій з підтягуванням котировок у режимі ріалтайм, але на цьому все і закінчується. Бонди, нерухомість, метали, тощо – відсутні. Неможливо додати їх до списку вашого портфелю.

Перший приклад один з популярних сайтів для людей котрі інтересуються фінансами та інвестиціями – Investing.com, зображений на рисунку 1.3, на тему фінансів, має велику базу даних та новин. Підтягує усі дані напряму із біржі та користується великою популярністю для інвесторів у фондовий ринок. Користувачі можуть створювати власний портфель із будь-яких присутніх акцій або індексів, одни чи декілька. Недоліки додатку полягають у відсутності, можливості додати позиції інших типів, що не відповідає потребам інвесторів які їх мають, присутність реклами, яку можна відключити тільки купляючи підписку.

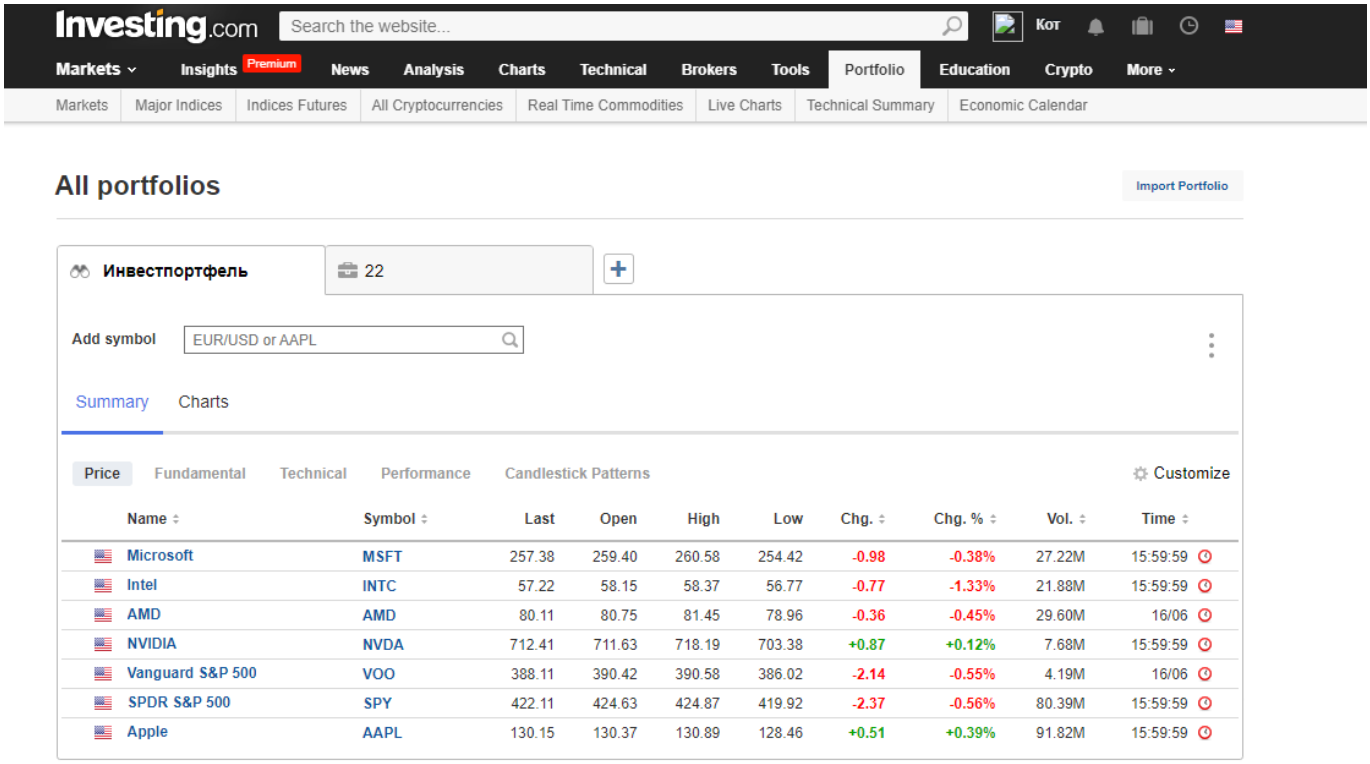


Рисунок 1.3 - портфель створений власноруч. Можливо додати тільки акції

Наступний сайт The Wall Street Journal? зображений на рисунку 1.4, влаштований більше як журнал для перегляду новин, але такоє має частину функціоналу зазначеному у попердьому кандадті. Сильні сторони: авторитетне видання котре має багато цікавої інформації в світі фінансів. Нажаль, тільки англомовна версія додатку присутня, тому хто не володіє буде не комфортно користуватися.

Рисунок 1.4 – The Wall Street Journal – основна сторінка

Як і у попереднього кандидата, в цьому видані є можливість також створити свій портфель, але без додаткового аналізу.

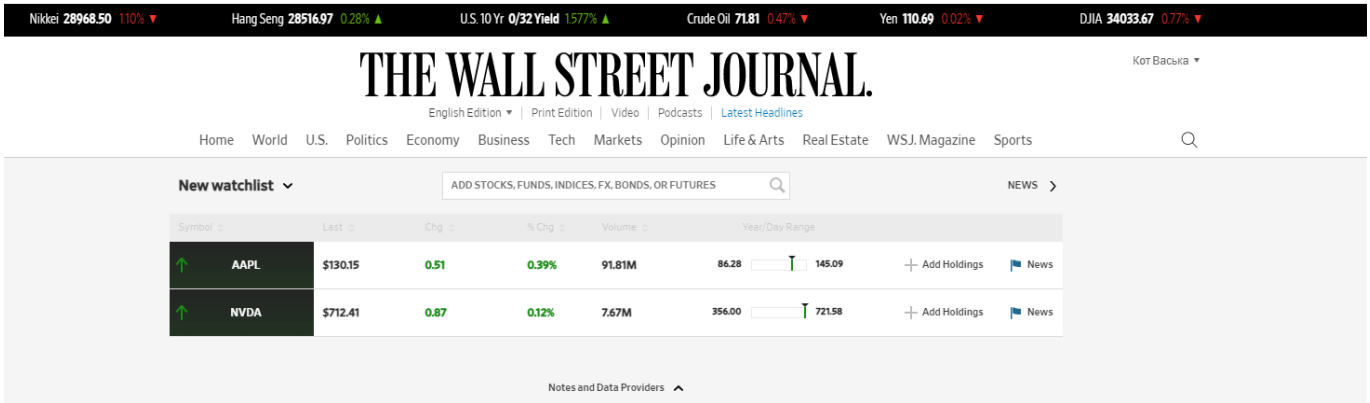
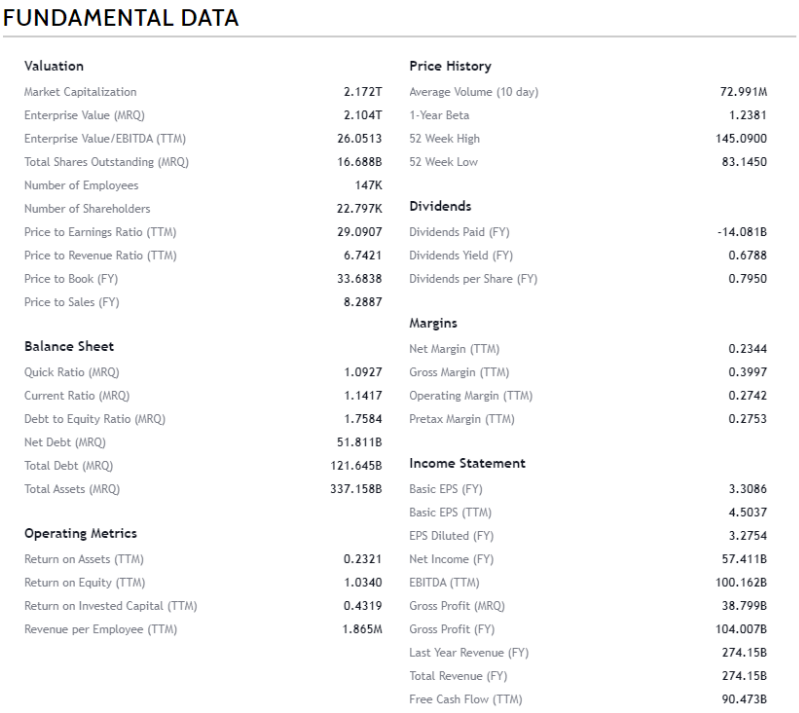


Рисунок 1.5 – The Wall Street Journal – сторінка трекінга акцій

Investopedia.com – найпопулярніша бібліотека для інвесторів. Як і попердні кандадти, має можливість слідкувати за цінами акцій, але не має можливості створити власний портвель, взагалі.

Рисунок 1.6 – Apple Inc. chart на сайті. Має фундаментальну дату але не має можливості додати це у свій портфель.



FUNDAMENTAL DATA

Valuation	
Market Capitalization	2.172T
Enterprise Value (MRQ)	2.104T
Enterprise Value/EBITDA (TTM)	26.0513
Total Shares Outstanding (MRQ)	16.688B
Number of Employees	147K
Number of Shareholders	22.797K
Price to Earnings Ratio (TTM)	29.0907
Price to Revenue Ratio (TTM)	6.7421
Price to Book (FY)	33.6838
Price to Sales (FY)	8.2887

Balance Sheet	
Quick Ratio (MRQ)	1.0927
Current Ratio (MRQ)	1.1417
Debt to Equity Ratio (MRQ)	1.7584
Net Debt (MRQ)	51.811B
Total Debt (MRQ)	121.645B
Total Assets (MRQ)	337.158B

Operating Metrics	
Return on Assets (TTM)	0.2321
Return on Equity (TTM)	1.0340
Return on Invested Capital (TTM)	0.4319
Revenue per Employee (TTM)	1.865M

Price History	
Average Volume (10 day)	72.991M
1-Year Beta	1.2381
52 Week High	145.0900
52 Week Low	83.1450

Dividends	
Dividends Paid (FY)	-14.081B
Dividends Yield (FY)	0.6788
Dividends per Share (FY)	0.7950

Margins	
Net Margin (TTM)	0.2344
Gross Margin (TTM)	0.3997
Operating Margin (TTM)	0.2742
Pretax Margin (TTM)	0.2753

Income Statement	
Basic EPS (FY)	3.3086
Basic EPS (TTM)	4.5037
EPS Diluted (FY)	3.2754
Net Income (FY)	57.411B
EBITDA (TTM)	100.162B
Gross Profit (MRQ)	38.799B
Gross Profit (FY)	104.007B
Last Year Revenue (FY)	274.15B
Total Revenue (FY)	274.15B
Free Cash Flow (TTM)	90.473B

Рисунок 1.7 – фундаментальні дані про компанію – базовий аналіз компанії без деталей.

Нижче, можно знайти підсумок переваг і недоліків найпопулярніших ресурсів для інвесторів. Зробивши відповідні підсумки на таблиці 1.1 продемонстровано основні переваги додатків. Схожі додатки мають свою аудиторію, мету, задачі тощо, тому вони можуть відрізнятися, але мета порівняння це знаходження недоліків у формуванні власного портфеля для інвестора. Саме та проблема котра буде вирішуватися розробляємою програмою.

Характеристика	Investing	The Wall Street Journal	Investopedia
Зручний інтерфейс	-	-	+
Формування власного портфелю	+	+	-
Підтягування даних реал-тайм	+	+	+

Можливість додання будь-яких типів інструментів	-	-	-
Цільове призначення	Інвестування – корисна інформація	Фінанси – новини фінансового світу	Фінанси – Wikipedia для інвесторів котру пишуть тільки обрані інвестори

Таблиця 1.1 – Порівняння найпопулярніших додатків для фінансів/інвестицій

2. ПОСТАНОВКА ЗАДАЧІ ТА МЕТОДИ ДОСЛІДЖЕННЯ

2.1 Задачі програмного продукту

Як було проілюстровано вище, дані приклади сайтів мають необідні дані та можуть бути корисними, але вони не надають змогу створювати власний портфель за бажанням. Цю задачу і слід буде виконати розробляємому програмному забезпеченню. Створення веб додатку зі зручним інтерфейсом, з можливістю створювати власний портфель котрий буде підтягувати ті дані, котрі можна знайти в мережі для трекінгу власних інструментів є можливістю стати конкурентним продуктом на ринку.

Software as a Service (програмне забезпечення як послуга) – це одна з форм хмарних обчислень, модель обслуговування, при якій передплатникам надається готове прикладне програмне забезпечення, повністю обслуговує провайдером. Постачальник в цій моделі самостійно управляє додатком, надаючи замовникам доступ до функцій з клієнтських пристроїв, як правило через мобільний додаток або веб-браузер.

Основна перевага моделі SaaS для споживача послуги полягає у відсутності витрат, пов'язаних з установкою, оновленням і підтримкою працездатності обладнання і працюючого на ньому програмного забезпечення.

У моделі SaaS:

- додаток пристосоване для віддаленого використання;
- одним додатком користується декілька клієнтів (додаток комунально);
- оплата стягується або у вигляді щомісячної абонентської плати, або на основі обсягу операцій;
- технічна підтримка програми включена в оплату;
- модернізація та оновлення додатка відбувається оперативно і прозоро для клієнтів.

Як і у всіх формах хмарних обчислень, замовники платять не за володіння програмним забезпеченням як таким, а за його оренду (тобто за його використання через мобільний додаток або веб-інтерфейс).

Таким чином, на відміну від класичної схеми ліцензування програмного забезпечення, замовник несе порівняно невеликі періодичні витрати, і йому не потрібно інвестувати значні кошти в придбання прикладної програми і необхідних програмно-платформних і апаратних засобів для його розгортання, а потім підтримувати його працездатність. Схема періодичної оплати передбачає, що якщо необхідність в програмному забезпеченні тимчасово відсутня, то замовник може призупинити його використання і заморозити виплати розробнику.

З точки зору розробника деякого пропрієтарного програмного забезпечення модель SaaS дозволяє ефективно боротися з неліцензійним програмним забезпеченням, оскільки програмне забезпечення як таке не потрапляє до кінцевих замовникам. Крім того, концепція SaaS часто дозволяє зменшити витрати на розгортання і впровадження систем технічної і консультаційної підтримки продукту, хоча і не виключає їх повністю.

Із зазначеного вище, слід вивести основні вимоги котрими має володіти продукт, ознаки:

- доступ до програмного забезпечення, розробленого відповідно до моделі ПО як послуга, надається віддалено по мережевих каналах і, як правило, через веб-

інтерфейс, крім того, можуть використовуватися тонкі клієнти і термінальний доступ;

- програмне забезпечення розгортається в центрі обробки даних у вигляді єдиного програмного ядра, з яким працюють всі замовники;
- програмне забезпечення надається на умовах сплати періодичних орендних платежів;
- обслуговування та оновлення програмного забезпечення виконується централізовано на стороні постачальника програми, що надається як послуга (SaaS);
- вартість технічної підтримки зазвичай включається в орендну плату.

Окрім цього, нам ще потрібно мати можливість створювати потрфель із позиціями, тобто програмне забезпечення повинне мати:

- можливість створювати користувачів для системи за імейлом
- можливість додавати позиції.
- можливість видаляти та редагувати позиції
- робити базовий аналіз портфелю користувача.

Повний перелік функціональних вимог наведено в таблиці 2.2

Корисний додаток для формування потрфелю буде містити в собі необхідні позиції різних типі, та підтягувати необхідні дані. Основний функціонал - це можливість створити нових користувачів та авторизувати існуючих. Якщо хтось забув пароль – йому потрібно допомогти його нагадати – направив листа на імейл.

Програма допоможе вам сформувати свій портфель, котрий буде мати тільки важливу для вас інформацію та допоможе із підрахунками, наприклад суми яку слід уплатити на податки, або основного стану портфелю: загальної суми в різних валютах, початкову та текущю суму позиції, тощо.

2.2 Вибір методів

Під час аналізу програмних продуктів конкурентів, було помічено що більшість с них використовує Javascript, котрий використовується як вбудований мова для програмного доступу до об'єктів додатків. Найбільш широке застосування знаходить в браузерах як мова сценаріїв для додання інтерактивності веб-сторінок.

Із урахуванням того що наш додаток буду використовувати як на мобільних пристроях та і на лаптомах у різних браузерах, цю мову було обрано через зручність та невиликі вимоги для железа користувача, аби він не мав проблем із користуванням.

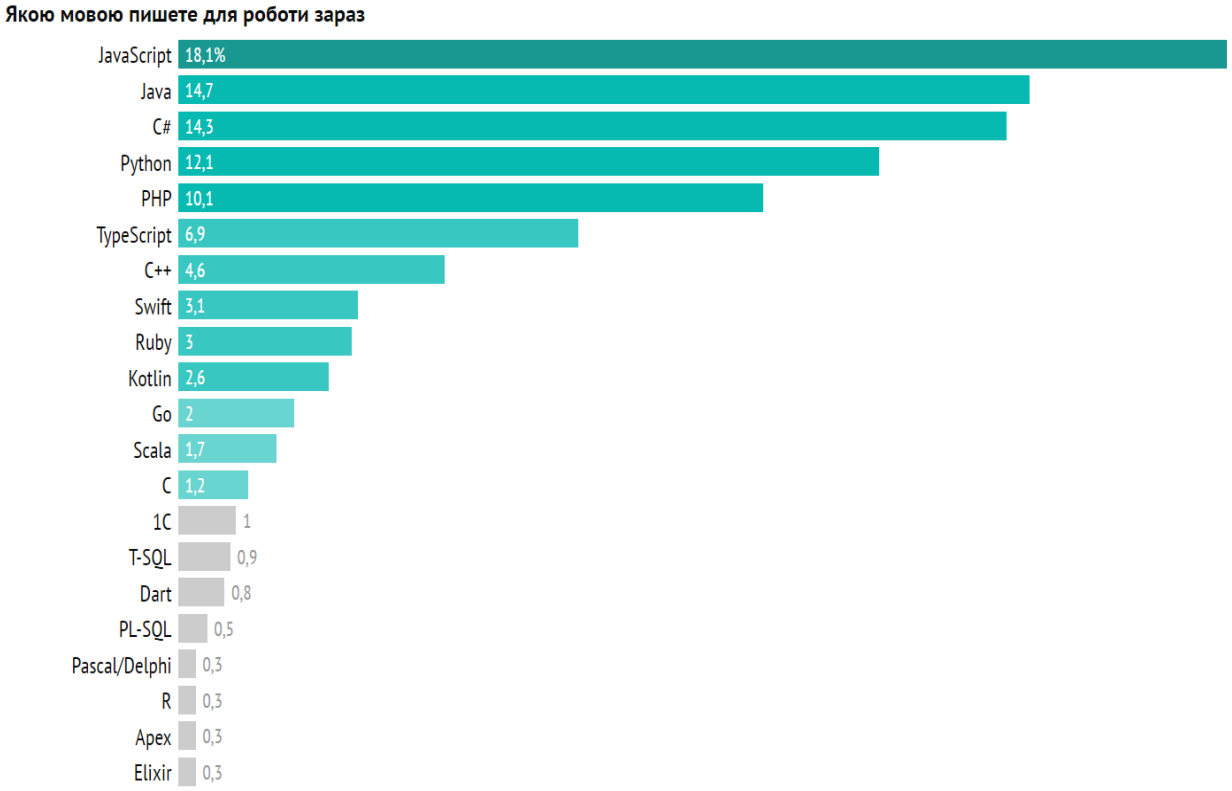


Рисунок 1.8 – рейтинг мов програмування 2021 за DOU.UA

Для того щоб розгорнути усю цю красу в облатці, було обрано Heroku - хмарна PaaS-платформа, що підтримує ряд мов програмування, та дозволяє компаніям створювати, доставляти, контролювати та масштабувати програми - ми найшвидший спосіб перейти від ідеї до URL-адреси, минаючи всі ці головні болі в інфраструктурі. Безкоштовна платформа котра дозволяю

зекономити кошти на початку розробки та надати можливість користувачам вже використовувати продукт, чи є крутіший варіант для початку?

Процес розробки починався з підбору стека технологій. Вибір бекенду пав на - node.js - з причин швидкої розробки, гнучкості та хороших допоміжних інструментів та інтеграцій. Фронтенд вдалося написати без використання JavaScript фреймворків, за формування дизайну відповідав CSS-фреймворк UIKit. База даних обрана NoSQL - MongoDB з причин простого горизонтального масштабування, роботи з колекціями та інтеграції з платформаю Node.js, база даних висить на бехкоштовній хмарі mongoDB Atlas. Хостингом є безкоштовна та проста у використанні платформа heroku. Сервер підтримує server side rendering для авторизації та зчитування даних та rest специфікацію - для повного керування даними.

2.3 Планування робіт

Виокремлення мети проекту методом SMART

SMART-цілі - метод постановки конкретних цілей, реалістичних завдань, які чітко обмежені часі та мають свій кінцевий термін та аналіз роботи виконавця якийзнає скільки роботи вже виконано.

Методологія SMART являє собою концепцію правил:

- Конкретність (Specific) - прозора постановка мети, де ціль повинна бути зрозумілою.
- Вимірюваність (Measurable) – виконавець має уявлення на якому етапі йде розробкапроекту
- Досяжність (Achievable) – мета конкретної задачі закріплена за одним виконавцем якийповинен розуміти всі критерії цілі та нести відповідальність.
- Реалістичність (Relevant) – мета повинна мати свої певні реальні потреби, які залежатьвід виконавця.

- Обмеженість в часі (Time-bound) – проект має бути виконаний в певний, заданий час.
- Кожен етап проекту потрібен мати свій

ліміт часу.

Результати розміщені у таблиці 2.1

Specific (Конкретність)	Платформа для формування власного портфелю
Measurable (Вимирюваність)	Результатом роботи є задоволеність користувачів
Achievable (Досяжність)	Реалізація виконана на мовах JavaScript, та node.js
Realistic (Актуальність)	В наявності є всі необхідні технічні та програмні засоби.
Timed (Обмеженість в часі)	Ціль має свій кінцевий термін. Кожен етап повинен бути виконаний вчасно. Терміни повинні бути обговорені виконавцем та керівним проекту або замовником.

Таблиця 2.1 – Висновок мети методом SMART

ABC-аналіз - метод, що дозволяє класифікувати ресурси за ступенем їх важливості. Цей аналіз є одним з методів раціоналізації і може застосовуватися в сфері діяльності будь-якого підприємства. В його основі лежить принцип Парето - 20% всіх товарів дають 80% обороту. По відношенню до ABC-аналізу правило Парето може прозвучати так: надійний контроль 20% позицій дозволяє на 80% контролювати систему, будь то запаси сировини і комплектуючих, або продуктовий ряд підприємства і т. П. Часто ABC-аналіз плутають з ABC-методом, розшифровуючи ABC як Activity Based Costing, що в корені невірно.

ABC-аналіз - аналіз шляхом ділення на три категорії:

- A - найбільш цінні, 20% - функціоналу; 80% - користуються
- B - проміжні, 30% - функціоналу; 15% - користуються
- C - найменш цінні, 50% - функціоналу; 5% - користуються

Залежно від цілей аналізу може бути виділено довільну кількість груп. Найчастіше виділяють 3, рідше 4-5 груп.

По суті, ABC-аналіз - це ранжування вимог за різними параметрами. Ранжувати таким чином можна все, що має достатню кількість статистичних даних. Результатом ABC аналізу є групування об'єктів за ступенем впливу на загальний результат.

ABC-аналіз ґрунтується на принципі дисбалансу, при проведенні якого будується графік залежності сукупного ефекту від кількості елементів. Такий графік називається кривою Парето, кривою Лоренца або ABC-кривої. За результатами аналізу асортиментні позиції ранжуються і групуються в залежності від розміру їх вкладу в сукупний ефект. У логістиці ABC-аналіз зазвичай застосовують з метою відстеження обсягів відвантаження певних артикулів і частоти звернень до тієї чи іншої позиції асортименту, а також для ранжирування клієнтів за кількістю або обсягом зроблених ними замовлень.

Управління вимогами

Функціональні вимоги - це опис дій програмного продукту. Він описує програмну систему або її компонент. Наприклад, ви можете взаємодіяти з користувачем або запитувати дані Користувача. Функціональні вимоги також називаються функціональними специфікаціями.

№	Вимога
1	Користувач потрібен мати можливість створити аккаунт, якщо потрібно
2	Користувач повинен мати можливість зайти в існуючий аккаунт, якщо такий було створено
3	Якщо користувач забув свій пароль, він повинен мати можливість відновити його
4	Користувач повинен отримати листа для відновлення пароля на імейл
5	Додаток повинен відображати головний екран після проходження аутентифікації
6	Додаток повинен відображати активні позиції портфелю
7	Додаток повинен відображати діаграму типів позицій портфелю, якщо такі є
8	Додаток повинен відображати діаграму співвідношення долі конкретної позиції до всієї частки портфелю
9	Додаток повинен мати можливість створення нових позицій
10	Додаток повинен мати можливість видалення або редагування позицій
11	Додаток повинен сумувати всі позиції портфелю
12	Додаток повинен відображати зміни ціни позиції
13	Додаток повинен показувати частки позицій у різних валютах

Таблиця 2.3 – Список функціональних вимог

Нефункціональні вимоги (NFR) визначають атрибут якості програмної системи. Вони оцінюють програмну систему на основі чутливості, зручності

використання, безпеки, переносність та інших нефункціональних стандартів, які мають вирішальне значення для успіху програмної системи. Недотримання нефункціональних вимог може призвести до того, що системи не відповідатимуть потребам Користувача. (таблиця. 2.4)

№	Вимога
1	Програма має бути розгорнена в облатці
2	Програма має бути доступна з моб. дивайсу
3	Програма має бути на англ. мові для більшого охвату аудиторії.

Таблиця 2.4 – Список нефункціональних вимог

Із допомогую методу ABC, слід кваліфікувати вимоги наступним чином:

- A – створення нових аккаунтів та користування існуючими; створення, редагування та видалення позицій;
- B – відображення даних та аналіз; відновлення паролів; використання додатку на різних девайсах;
- C – все інше.

3. МОДЕЛЮВАННЯ ДОДАТКУ

3.1. Збір вимог

Цільова аудиторія

Результатом цього проекту є додаток, який служить помічником для користувачів, які хочуть сформувати власний портфель інструментів. Користувач може обрати тип інструмента, задати такі деталі як: Quantity, Buy Price, Buy Date, USD/UAH для підрахування точної суми позиції та для зручнішого урахування податків. Цільова аудиторія – це інвестори, люди, які планують вкладавати кошти у різні компанії, об’єкти, тощо для із цілю збереження коштів або інвестицій.

Сценарії взаємодії

Після логіну в систему користувач повинен бачити свій портфель та діаграми початкового аналізу, також користувач має можливість персоналізувати назву портфелю.

Критичним функціоналом є створення та корегування існуючих позицій: видалення чи внесення змін у існуючи, адже без цього неможливе формування портфелю. Також нижче будуть наведені зазначені вище діаграми котрі будуть показувати важливу інформацію

3.2 Проектування прототипу

На цьому етапі моделювання програмного продукту вказується прототип екранів взаємодії з користувачем - це необхідно для установки основних елементів інтерфейсу для вже більш простого проектування дизайну користувальницького інтерфейсу. Ця модель використання прототипу додатка використовується при отриманні інформації від замовника, де він вказує основні потреби інтерфейсу і всі варіанти взаємодії між додатком і користувачем, які йому потрібні, і після узгодження прототипу проект переходить до етапу створення користувальницького інтерфейсу.

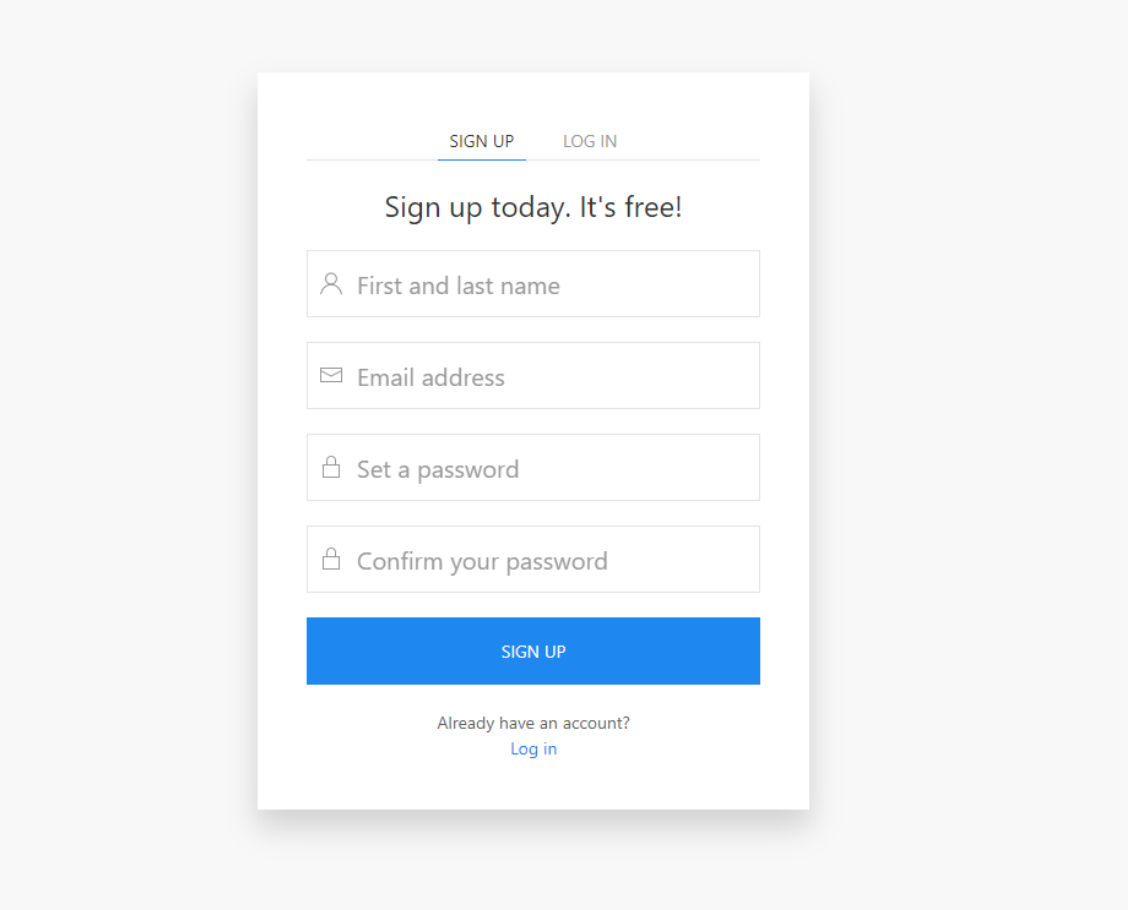


Рисунок 3.1–дизайну вікна створення нового аккаунту

Після успішної авторизації або реєстрації користувач потрапляє на головний екран вибору тестового розділу (рис. 3.2).

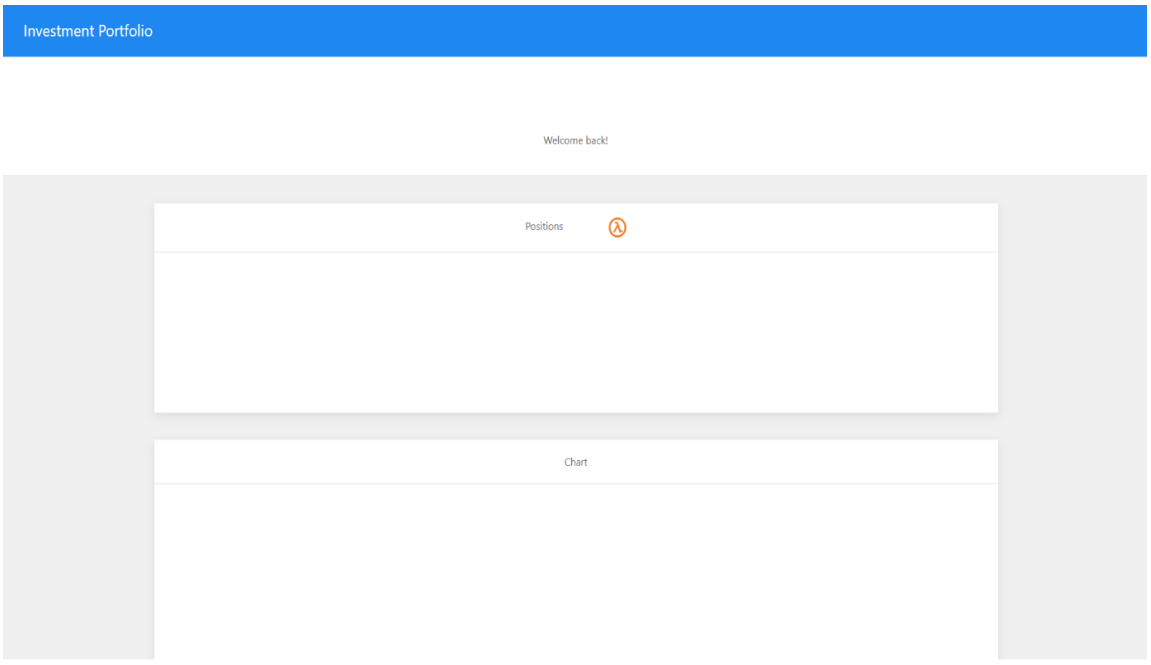


Рисунок 3.2– Схематичне зображення домашнього екрану

Після цього у користувача повинен бути change mode в котрому він буде вносити зміни чи створювати нові позиції.

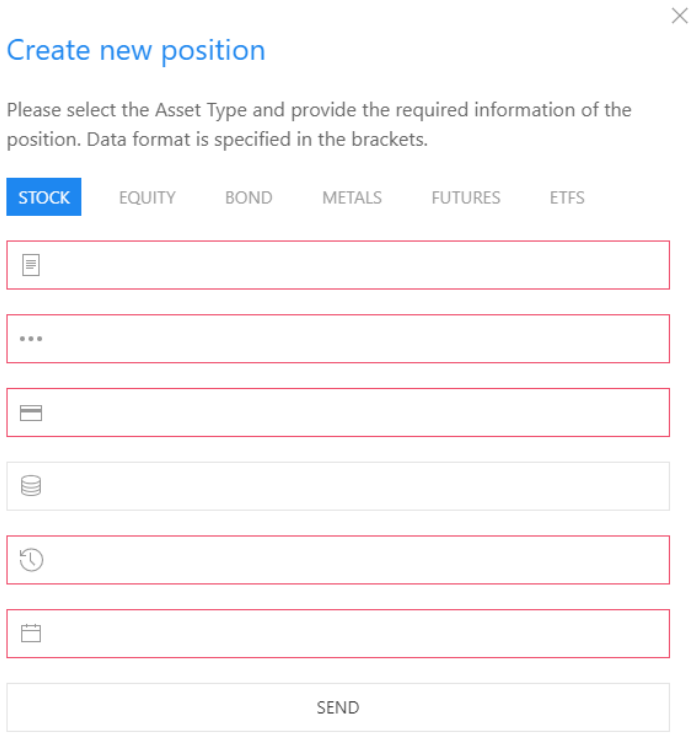


Рисунок 3.3 - вікно створення позицій

Архітектура програмного продукту

Архітектура веб застосунку – Model-View-Controller. Модель представляє собою дані і реагує на дії контролеру, змінюючи свій стан. Представлення(View) відповідає за відображення даних у коректній формі для користувача. Контроллер реагує на дії користувача та оповіщає модель про необхідність зміни даних у базі. Хоча індустрія почала відходити від стандартного підходу MVC, він залишається хорошим архітектурним рішенням для застосунків.

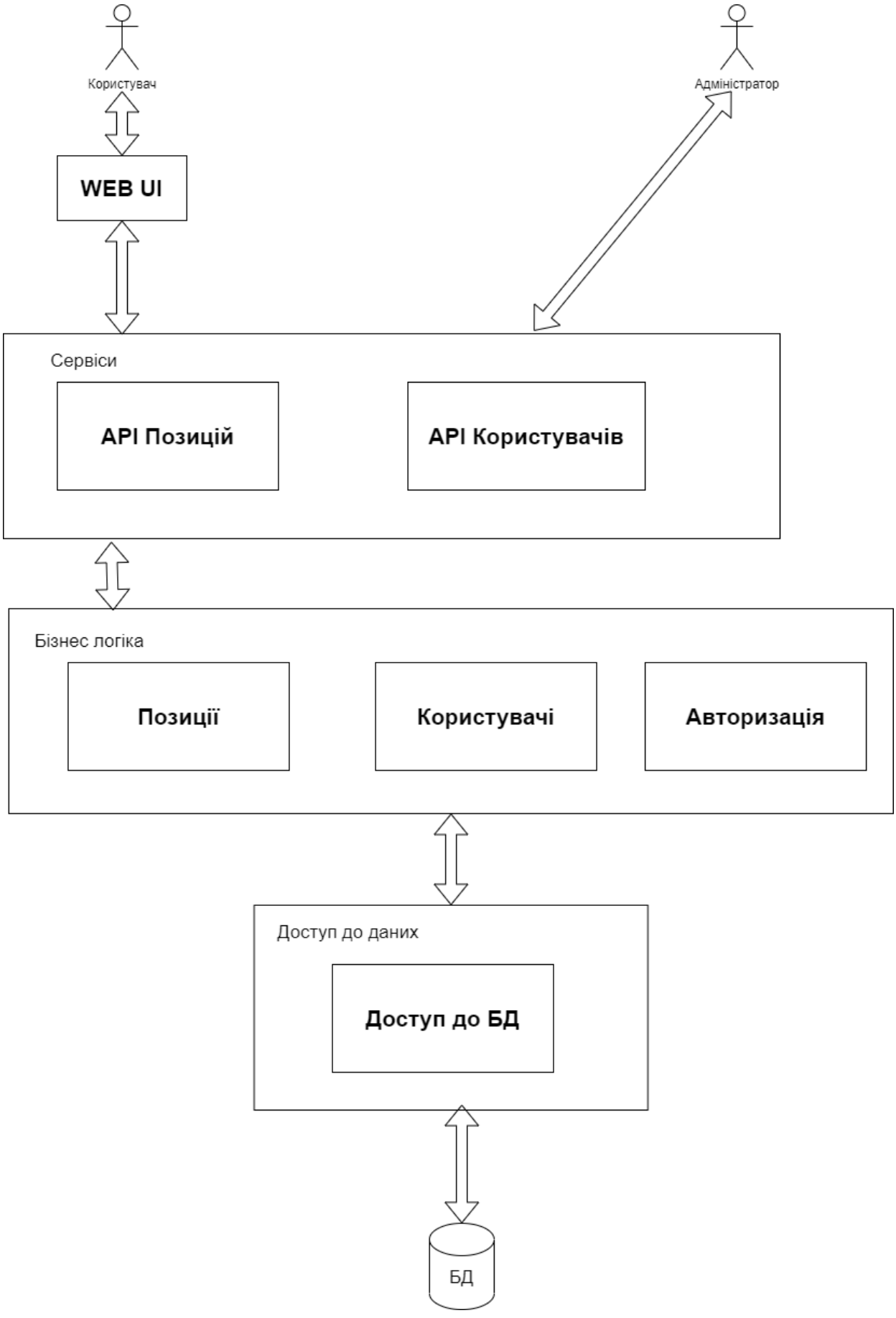


Рисунок 3.4 – Архітектура програмного продукту

Діаграма взаємодії

Діаграма варіантів використання - це список дій, сценарій, в якому користувач взаємодіє з додатком або програмою для виконання дії для досягнення певної мети.

Варіанти використання були вказані в тексті та представлені діаграмою варіантів використання. Компонент сутності моделювання діаграми взаємодії допомагає розробити систему з точки зору майбутнього користувача. Використовуючи прецедент, ви можете описати вимоги користувача, вимоги до взаємодії з системою та опис взаємодії людей і компаній в реальному житті.

Була створена діаграма варіантів використання, яка показувати відносини між суб'єктами та варіантами використання і є невід'ємною частиною моделі варіантів використання. Діаграма була побудована з використанням UML і сайту draw.io.

Параметри використання для авторизованих користувачів:

- Створити новий інструмент:
- Обрати тип нового інструменту.
- Ввести необхідні мандаторні поля:
  - Header
  - Quantity
  - Buy Price $
  - USD/UAH
  - Purchase date
- Ввести значення у немандаторне поле – Current amt. in UAH.
- Створити позицію.
- Редагутавити позиція за будь-яких полем чи типом активу.
- Видалити будь-яку позицію.
- Переглянути Chart:
  - by Position – процентне співвідношення по кожному типу до загального
  - by Asset Type – процентне співвідношення будь-якого активу до загальної вартості портфелю

Варіанти використання для неавторизованих користувачів:

- Реєстрація користувача.
- Авторизація користувача.
- Відновлення пароля.

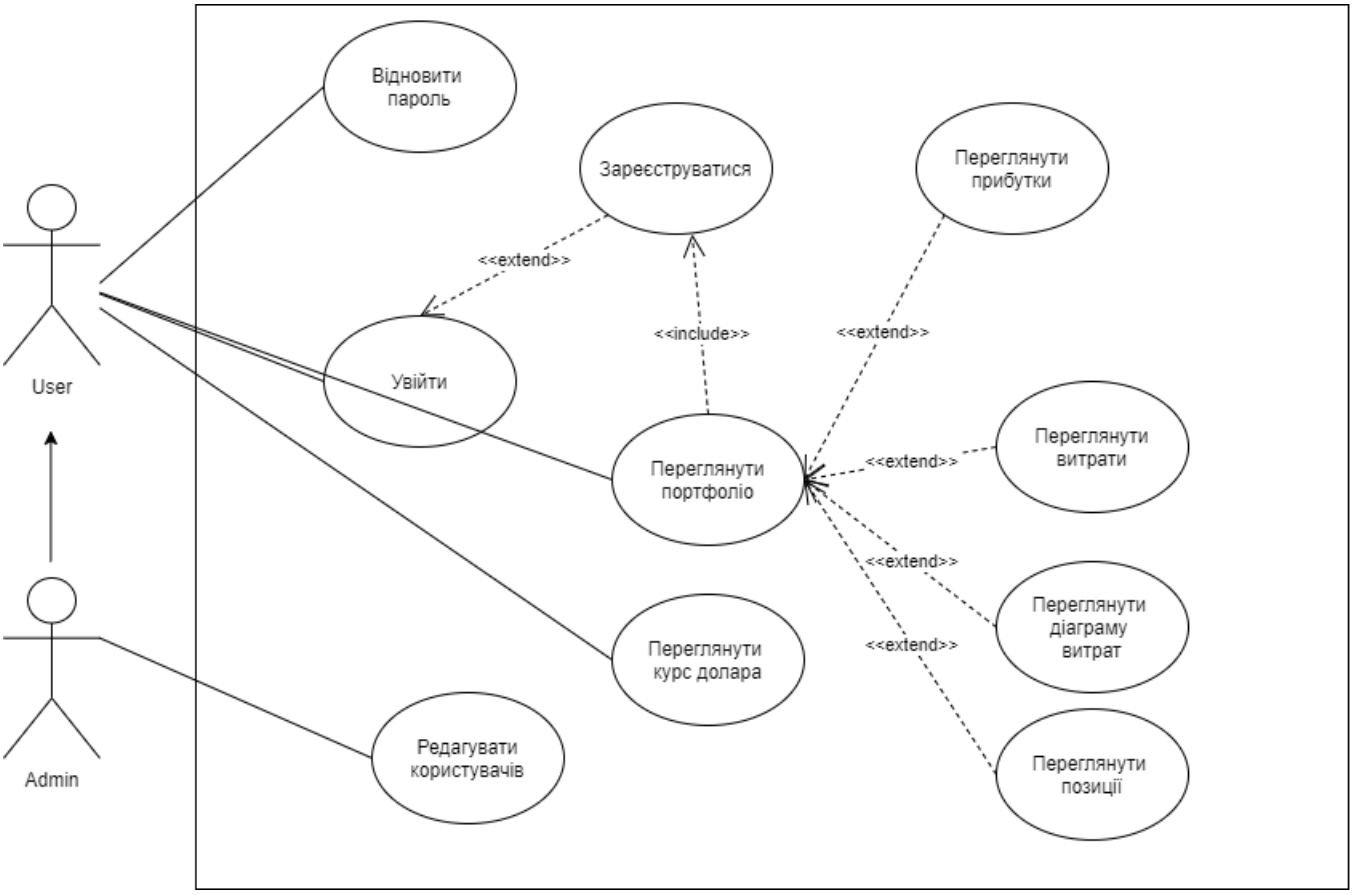


Рисунок 3.5 – Діарама прецедентів

Діаграма послідовності — різновид діаграми в UML. Діаграма послідовності відображає взаємодії об'єктів впорядкованих за часом. Зокрема, такі діаграми відображають задіяні об'єкти та послідовність відправлених повідомлень.

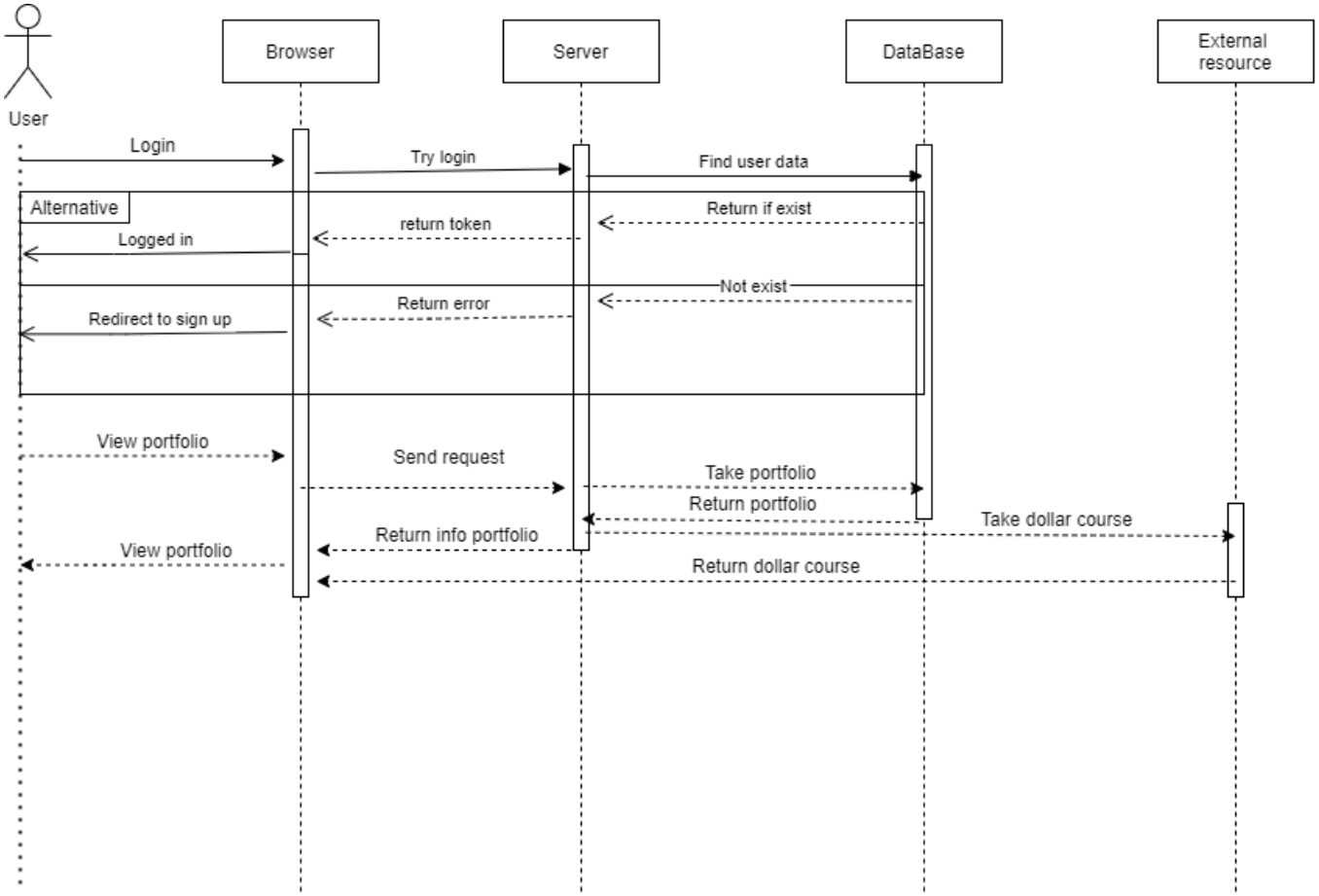


Рисунок 3.6 – Діарама послідовності

4. РОЗРОБКА ДОДАТКУ

4.1 Огляд основних можливостей програмного продукту

Розроблюваний додаток повинен забезпечувати наступний функціонал:

1. **Інтуїтивно зрозумілий інтерфейс**. Було знайдено приклад дизайну та по ньому створенно аналог дизайну сайту.
2. **Швидка робота та коректне відображення даних**. Було знайдено та використано методи для забезпечення відповідної якості ПЗ.
3. **Аутефікація.** Всі дані користувача знаходяться у базі в облаці.
4. **Вибір користувачем даних для внеску.** Користувач сам повинен задавати дані для внесення у систему.
5. **Тестування продукту.** Система працює коректно у відповідності до поставлених умов.
6. **Створення діаграм.** Діаграми повинні бути створенні для відповідних позицій та типів активів.

При розробці дизайну для застосунку виникла дилема: розробляти дизайн з нуля або використати готові, перевірені часом рішення. Обрано було використати готове рішення, так як його розробляли професіональні дизайнери та frontend розробники. Це одразу позбавило від проблем верстки та написання важких CSS файлів.

Обраний CSS фреймворк UIKit - легкий і модульний інтерфейсний фреймворк для розробки швидких і потужних веб-інтерфейсів. Процес розробки дизайну проходив у вигляді пошуку готових рішень на фреймворку та перелаштування під свої запити.

Процес розробки серверної частини починався з підбору стека технологій. Вибір бекенду пав на - node.js - з причин швидкої розробки, гнучкості та хороших допоміжних інструментів та інтеграцій.

Сервер підтримує server side rendering для авторизації, зчитування та зміни даних. Також реалізована підтримка rest специфікації - для повного

керування даними адміністратору.

Фрейморк для бекенду Express.js - дефакто стандарт платформи node.js. Стояв вибір перед koa.js, fastify, nest.js. Обрано express з причини великої підтримки спільноти та готових рішень для будь-яких проблем. В ньому є модулі безпеки, MVC, підтримка передачі статичних файлів та server side rendering за допомогою шаблонізатора PUG. В процесі розробки виникли труднощі зі зберіганням секретних даних, таких як: пароль підключення до бази даних, сіль для хешування JWT токенів та ін. Вирішено додаванням модулю dotenv та винесенням секретних даних у окремий файл ".env" у корінній директорії проекту. Реалізоване обмеження доступу користувачам до роутів, якими має користуватися лише адіністратор. Логіку було винесено в окремий файл авторизації.

З причин слабкої типізації Java Script для підтримки коду в чистоті та DRY принципів в проекті використовується лінтер ESLint, конфігураційний файл «.eslintrc.json» знаходиться в кореневій директорії проекту. Для автоматичного форматування коду використовується розгирення «prettier».

Усі файли бізнес логіки та серверної логіки поміщені у директорію «src» та розбиті у підпапки зі зрозумілими з контексту та описуючі MVC модель назвами «controllers», «model», «routes» і т.д. Вхідна точка проекту – це файл «server.js», у ньому запускається застосунок, конфігурується підключення до бази даних.

База даних обрана NoSQL - MongoDB з причин простого горизонтального масштабування, роботи з колекціями та інтеграції з платформаю Node.js. Для розгортування бази даних у хмарі використано сервіс mongoDB Atlas, який пропонує безкоштовний тариф з певними обмеженнями у використанні пам’яті. MongoDB Atlas - це найбільш інноваційна служба хмарних баз даних на ринку з неперевершеним розподілом даних і мобільністю в AWS, Azure і Google Cloud, вбудованої автоматизацією для оптимізації ресурсів і робочих навантажень і багатьом

іншим. З’єднання серверу з базою даних виконується шляхом передачі посилання підключення по сокету до хостингу до ORM для роботи з mongoDB - «mongoose». У процессі проектування системи було вирішено використати дві колекції: користувач та позиції з відношенням один до багатьох, відповідно. Mongoose – пропонує свою власну валідацію даних з простиим налаштуванням конфігурації, тому обрано було працювати не з чистим драйвером бази даних, а з ORM бібліотекою.

На клієнтській частині використані передові технології з ілюстрації графіків - бібліотека chart.js. Також реалізована повноцінна авторизація, JWT автентифікації на стороні клієнта зберігається у localstorage та у cookie-сховищі. Коли користувач заходить на свою домашню сторінку/портфоліо, він має можливість змінити назву портфоліо та переглянути всю інформацію по своїм інвестиціям.

В процесі розробки потрібно було вирішувати, чи варто використовувати фреймворк для розробки frontend частини застосунку. Вирішено було писати без фреймворку, на чистому JavaScript, за необхідності з використанням Ajax – запитів до серверу, використовуючи бібліотеку «Axios».

До кожної кнопки створення, редагування та видалення позиції існує своя функція для валідації даних, повідомлення користувача про те, чи пройшли дані валідацію. Після успішної валідації дані надсилаються на сервер та за допомогою асинхронної моделі, після надходження відповіді від сервісу клієнту повідомляється, успішний був запит чи ні. Процес авторизації реалізований таким же чином.

Heroku - хмарна PaaS-платформа, що підтримує ряд мов програмування, до яких входить платформа Node.js. Процес поставки застосунку на сервер проходить за допомогою системи git. На комп’ютер розробника встановлюється командний інтерфейс heroku. В гіт репозиторій проекту додається покажчик, який посилається на іншу копію сховища, звичайно

розташовану на віддаленому сервері. Коли приходить час релізу, за допомогою «push» – команди код усього застосунку передається на сервер, де відбувається завантаження необхідних модулів та запуск застосунку. На відміну від інших хостингів heroku – це платформа, тому в ній процес перенесення та запуску застосунку набагато зручніший, ніж у конкурентів. Обрано цю платформу також з причини безкоштовних послуг для утримання невеликого застосунку.

5. ТЕСТУВАННЯ

Стратегія тестування розробленого продукту була наступна – після кожної ітерації розробки, було пройдено смок тестування нового функціоналу з метою: впевнитися що він працює.

Після розробки основного функціоналу та “code freeze`y ”, було зроблено перевірки зазначені у чек листі (табл. 5.1)

Чек-ліст було обрано тому що користі від витрати часу на написання тестів та їх проходження немає, в моему проекті. Тест план не було створенно по тій єї ж причине. Так як на проекті я тільки 1 людина, яка все і робить, вона добре знакома із вимогами, котрі сам створював, та із системою котру сам розробляв, тому перевірки можна зробити за наявності чек-ліста котрий має усі перевірки які необхідні.

Таблиця 5.1 – Тестування мобільного додатку – чек-ліст

#	**Опис перевірки**
1	Створення нового користувача
2	Авторизація існуючого аккаунту
3	Відновлення паролю
4	Зміна назви портфелю
5	Переход до change mode та назад
6	Додавання нової позиції за будь-яким типом ассету
7	Редагування будь-якої інформації у існуючій позиції
8	Видалення будь-якої позиції
9	Підрахунок total, якщо всі стовпчики мають дані
10	Створення діаграми для Asset Type
11	Створення діаграми для Position(Desctiption)
12	Вихід із аккаунту до сторінки логіну/реєстрації

ВИСНОВОК

Результатом кваліфікаційної роботи є система що розгорнута в облатці та працює з доступом через будь-який браузер або навіть мобільний додаток.

Система допомагає інвесторам займатися трекінгом існуючих інвестицій в своєму портфелі. Інструменти можуть бути створені в залежності від побажань користувача: обрати тип, ввести необхідні дані, котрі система після цього автоматично підрахує.

Система виконує вимоги SaaS та може бути одночасно використаною декількома користувачами котрі будуть мати свої власні портфелі та позиції в них, для яких будуть створені відповідні діаграми для базового аналізу портфелю.

Найбільшу користь система буде приносити пасивним інвесторам, атже це ті люди котрі купують позиції «в длинную» та держать їх довгий час. Їм непотрібно часто рахувати податки, або заходити/виходити в нові старі позиці. В основному вони докупають вже існуючи, тому розроблена система стане їм в нагоді.
Під час роботи було вдосконалені навики роботи із Javascript & node.js; опанована робота із mongoDB. Робота над проектом допомогла поглибити та вдосконали знання в фінансах та інвестиціях та відкрити свій потрфель, котрий за моменту створення(4 місяці) вже виріс на 6.07%:

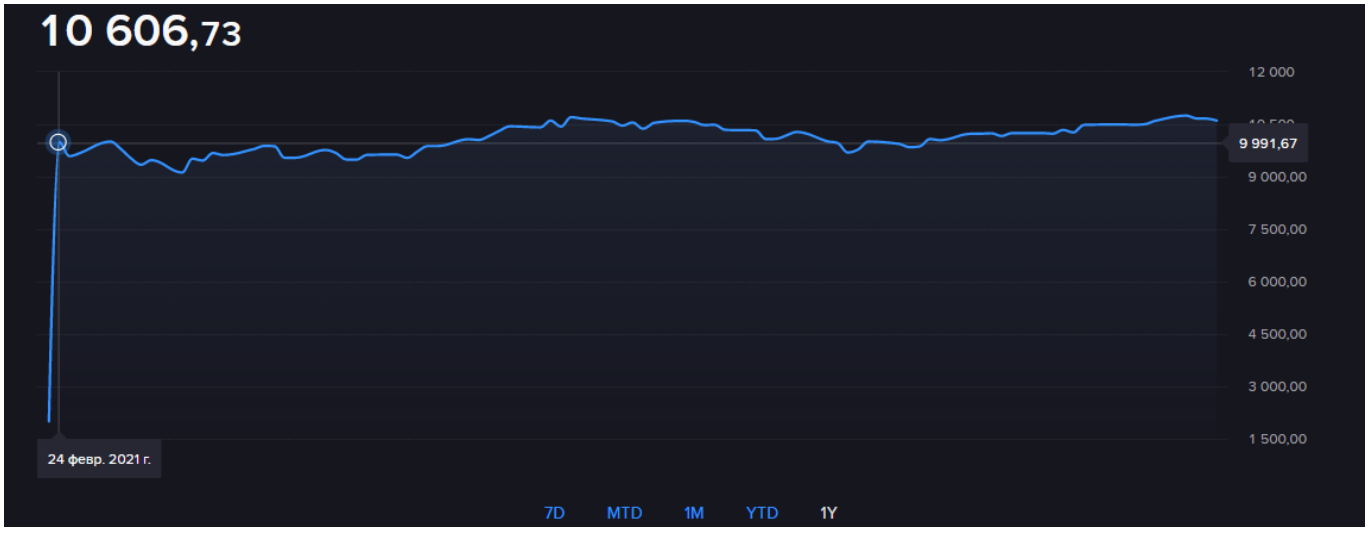


Рисунок 6.1 – стан портфелю на порталі Interactive Brokers

СПИСОК ЛІТЕРАТУРИ

1. Software Architecture Document https://projects.cecs.pdx.edu/attachments/download/1130/SAD_DTCPII_ver1.3_andy.doc (дата звернення 04.05.2020)
2. https://stackru.com/questions/54871399/metodyi-ekzemplyara-mongoose-privodit-k-oshibke-tipa-metod-ne-najden (дата звернення 20.04.2021)
3. https://pastebin.com/VbgTBDDQ - (дата звернення 06.05.2021)
4. http://ni.biz.ua/11/11_6/11_69393_metodika-analiza-konkurentosposobnosti.html - АБС аналіз задач (дата звернення 12.05.2021)
5. https://stackoverflow.com/questions/64527340/model-find-returns-empty-even-though-there-are-documents-in-the-collection (дата звернення 15.01.2021)
6. https://github.com/graphql-compose/graphql-compose-mongoose/issues/158 (дата звернення 15.01.2021)

ДОДАТОК А

ТЕХНІЧНЕ ЗАВДАННЯ

на Розробка SaaS для менеджменту інвестицій

Продовження додатку А

1. Призначення й мета створення додатку.

1.1. Призначення додатку

Додаток призначений для менеджменту інвестицій користувача: інструмент трекінгу.

1.2. Мета створення додатку

Допомогти користувачам слідкувати за станом своїх позицій.

1.3. Цільова аудиторія

Пасивні інвестори. Люди котрі планують інвестувати кошти

2. Вимоги програмного продукту

2.1. Вимоги до програмного продукту загалом

2.1.1. Вимоги до структури й функціонування

Створений продукт має бути зроблений в мінімалістичному стилі та інтуїтивно зрозумілим для користувачів; повинна бути можливість створювати, редагувати та видаляти позиції із портфелю. Відображення відповідних діаграм.

2.1.2. Вимоги до користувачів та персоналу

Для користуванням додатком потрібно створити аккаунт у системі – користувач повинен пати імейл для цього та ввести необхідні данні.

2.1.3. Вимоги до подання інформації

Користувач повинен мати доступ до мережі інтернет.

2.1.4. Основні вимоги

2.1.4.1. Структура додатку

Додаток повинен складатися з наступних екранів користувача:

- Екран аутентифікація користувача (авторизація, реєстрація, відновлення пароля за допомогою імейлу).
- Головний екран – користувач бачить свій портфель та діаграми.
- Спливаюче вікно створення – користувачеві буде надано вікно де він зможе ввести необхідні дані для того щоб додати нову позицію.
- Спливаюче вікно редагування – користувачеві буде надано віконце для редагування атрибутів конкретної позиції.
- Спливаюче вікно видалення – за бажанням, користувач може видалити будь-яку позицію в відповідному віконці при натиску на кнопку.
- Секція діаграм – місце де показуються діаграми стосовно портфелю користувача.

2.1.4.2. Навігація

На сторінці home є вся необхідна інформація, інших немає. Користувач може вийти із системи за бажанням.

2.1.4.3. Система навігації (дизайн головного екрану додатку)

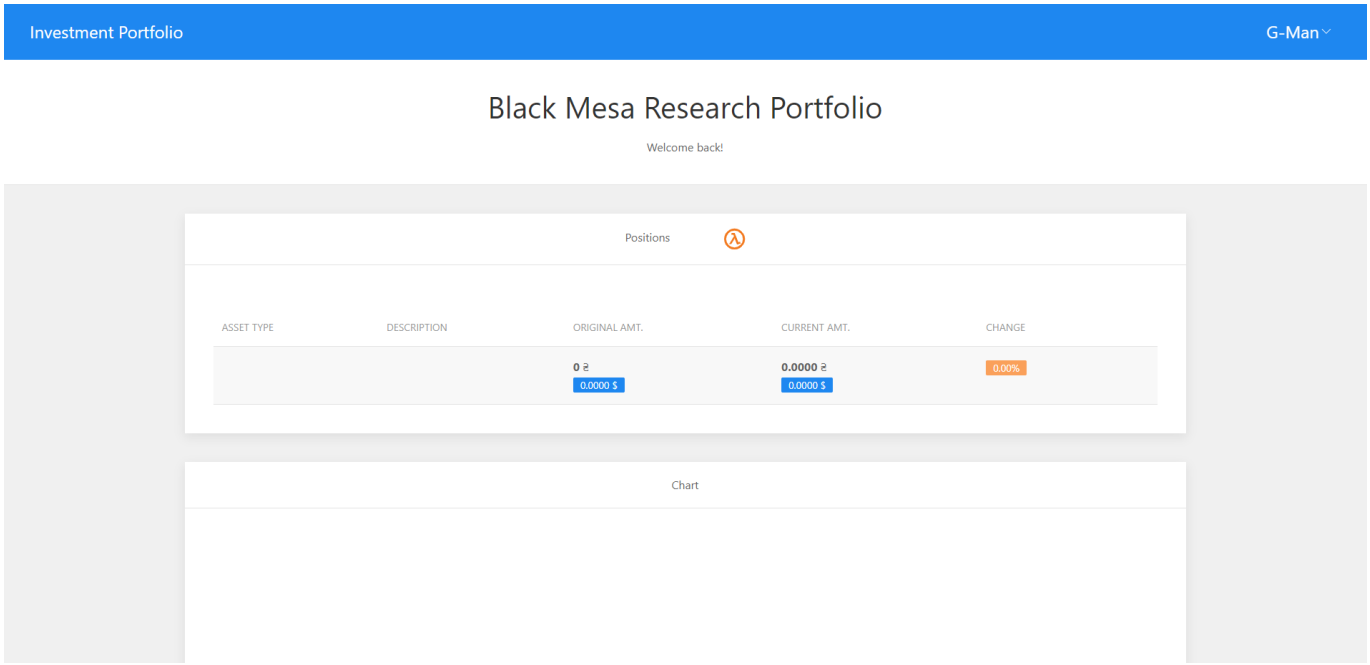


Рисунок А.1 – Дизайн головного екрану додатку (**G-Man** ім’я користувача, **Black Mesa Research Portfolio** – персоналізована назва)

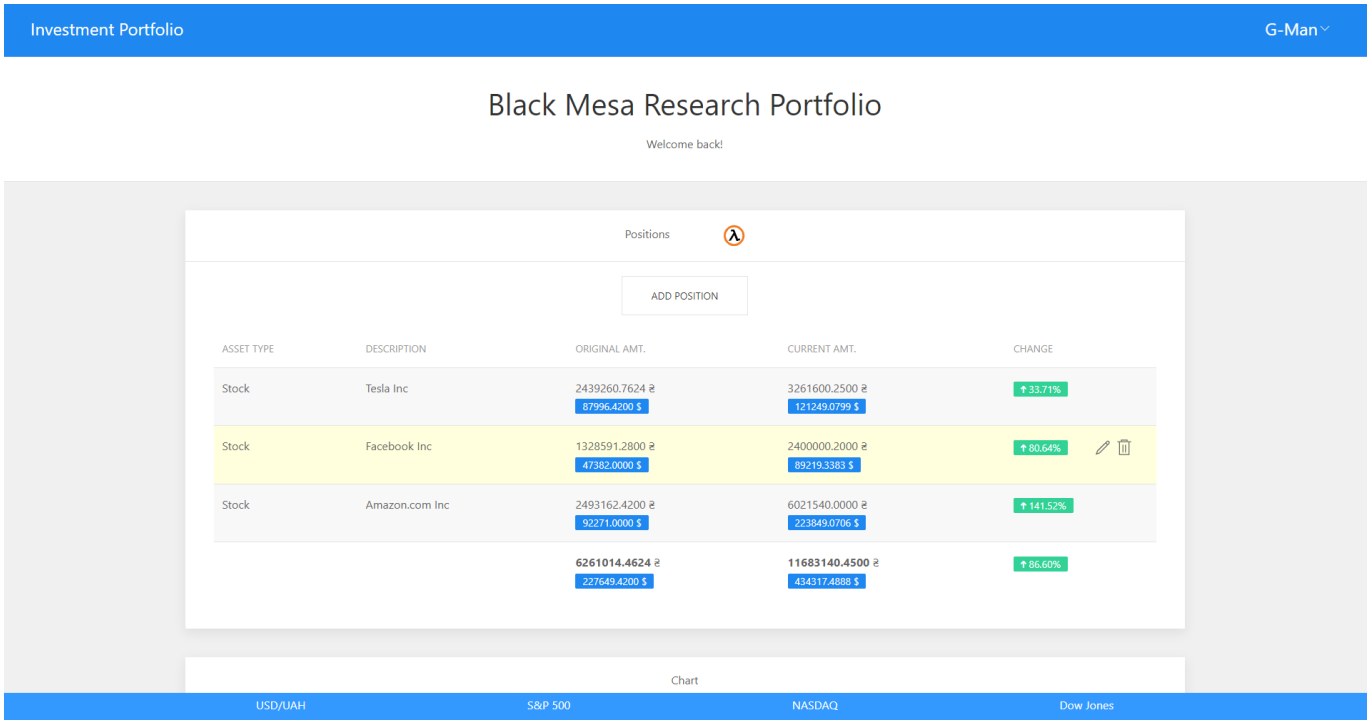


Рисунок А.2 – Дизайн головного екрану після додання позицій

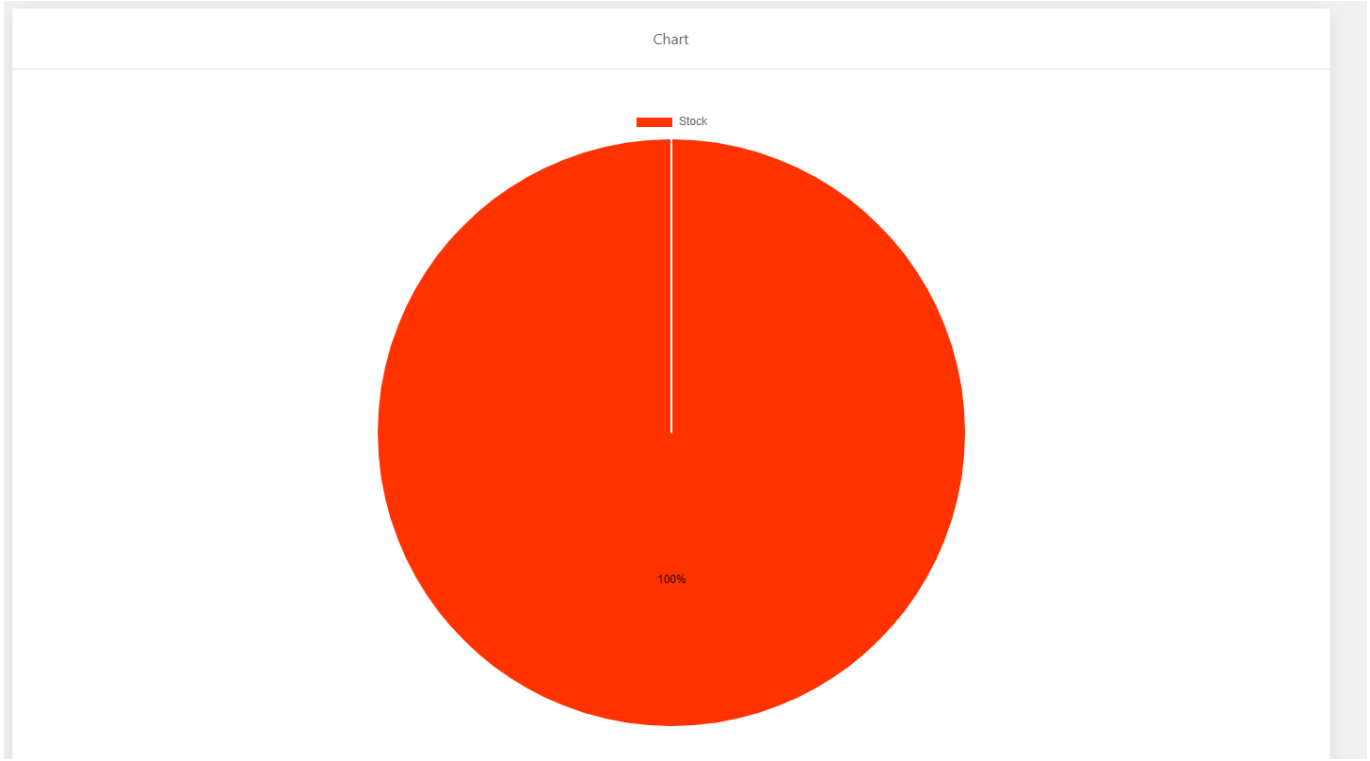


Рисунок А.3 – Дизайн секції Chart – *by Asset Type*

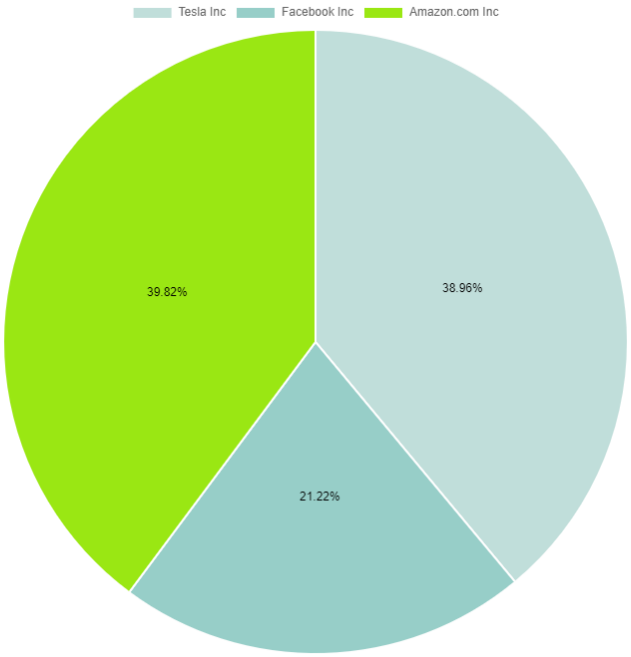


Рисунок А.4 – Дизайн секції Chart – *by Position*
(колір обираються випадково)

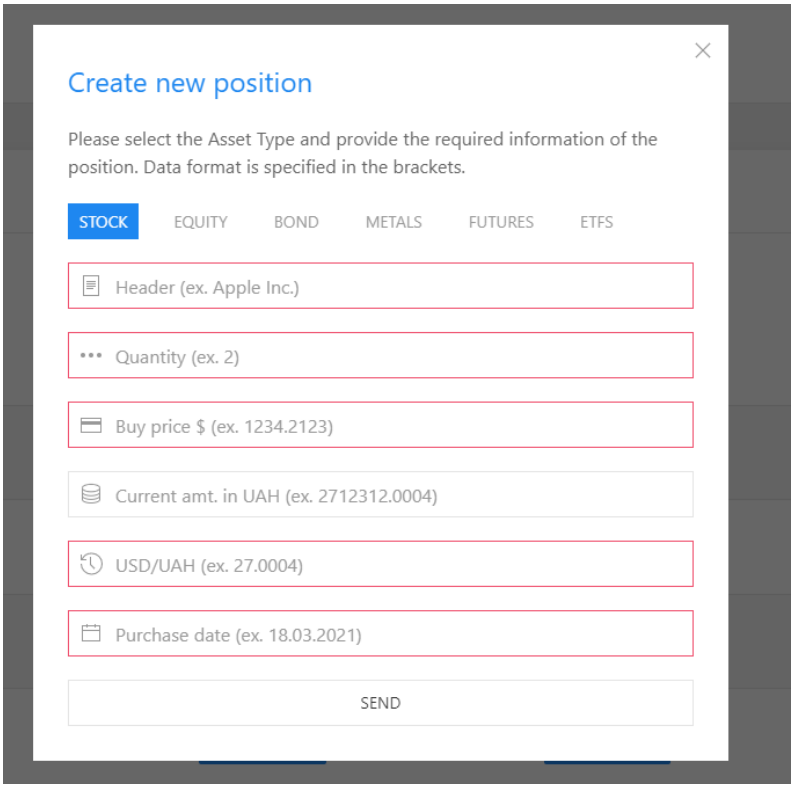


Рисунок А.5 – Дизайн вікна додавання нової позиції

2.1.4.4. Типові навігаційні й інформаційні елементи

- Відсутні. У користувача всього 1 сторінка на котрій він може все дивитися та створювати/редагувати

2.2. Вимоги до видів забезпечення

2.2.1. Вимоги до інформаційного забезпечення

Back-end - node.js - з причин швидкої розробки, гнучкості та хороших допоміжних інструментів та інтеграцій. Front-end вдалося написати без використання JavaScript фреймворків, за формування дизайну відповідав CSS-фреймворк UIKit. База даних обрана NoSQL - MongoDB з причин простого горизонтального масштабування, роботи з колекціями та інтеграції з платформаю Node.js, база даних висить на безкоштовній хмарі mongoDB Atlas. Хостингом є безкоштовна та проста у використанні платформа heroku.

2.2.2. Вимоги до лінгвістичного забезпечення

Додаток повинен бути виконаний англійскою мовою.

2.2.3. Вимоги до програмного забезпечення

Програмне забезпечення клієнтської частини повинне задовольняти наступним вимогам:

- Наявність телефону або персонального комп’ютера / ноутбуку котрі підтримують браузер та мережу інтернет

1. Склад і зміст робіт

Докладний опис етапів роботи зі створення додатку наведено в табл. А1.

Таблиця А1 – Етапи створення функціонального додатку

№	Склад і зміст робіт	Строк розробки (у робочих днях)
1	Дослідження предметної області	5 днів
2	Створення дизайн макетів екранів додатка	10 днів
3	Верстка екранів	10 днів
4	Створення функціоналу додатку	25 днів
5	Тестування та знаходження багів.	3 днів
6	Виправлення багів	2 дні

ДОДАТОК Б

Файл positionController

```
const { promisify } = require('util');
const jwt = require('jsonwebtoken');
const fetch = require('node-fetch');

const Position = require('../models/positionModel');
const User = require('../models/userModel');
const factory = require('./handlerFactory');
const AppError = require('../utils/appError');
const catchAsync = require('../utils/catchAsync');

exports.createPosition = catchAsync(async (req, res, next) => {
  const token = req.body.token || req.cookies.jwt;
  if (!token) {
    next(new AppError('You are not logged in!', 403));
  }

  const decoded = await promisify(jwt.verify)(
    token,
    process.env.JWT_SECRET
  );
  req.body.owner = decoded.id;

  if (!req.body.usdCourse) {
    await fetch(
```

```
      'https://api.privatbank.ua/p24api/pubinfo?json&exchange&coursid=5'
    )
      .then((r) => r.json())
      .then((r) => {
        req.body.usdCourse = r[0].buy.slice(0, 5);
      });
  }
  const doc = await Position.create(req.body);
  await User.findByIdAndUpdate(req.body.owner, {
    $push: { positions: doc._id },
  });
  res.status(201).json({
    status: 'success',
    data: {
      data: doc,
    },
  });
});

exports.getPosition = factory.getOne(Position);
exports.getAllPositions = factory.getAll(Position);
exports.updatePosition = factory.updateOne(Position);
exports.deletePosition = catchAsync(async (req, res, next) => {
  const doc = await Position.findByIdAndDelete(req.params.id);
  if (!doc) {
    return next(new AppError('No document found with that ID', 404));
```

```
  }

  if (doc.owner) {
    await User.findByIdAndUpdate(doc.owner, {
      $pull: { positions: doc._id },
    });
  }

  res.status(204).json({
    status: 'success',
    data: null,
  });
});
```

файл контролеру позицій користувача. має в собі всі CRUD операції з позиціями.

- createPosition - має кастомну логіку, доступ тільки авторизованим користувачам, власника присвоює автоматично, беручи його з токену. підтягує курс долара до гривні з національного сайту приват банку та створює позицію в базі даних.
- getPosition - функція для взяття позиції з бази даних за id
- getAllPositions - функція для взяття всіх позиціх з бази даних
- updatePosition - функція для оновлення позиції в бази даних за id
- deletePosition - функція для видалення позиції в бази даних за id. також видаляє її з поля positions колекції users

Файл UserModel

```
const crypto = require('crypto');
```

```
const mongoose = require('mongoose');
const validator = require('validator');
const bcrypt = require('bcryptjs');

const userSchema = new mongoose.Schema({
  name: {
    type: String,
    required: [true, 'Please tell us your name!'],
    minLength: 2,
    maxLength: 50,
  },
  portfolioName: {
    type: String,
    required: [true, 'Please tell us portfolioName!'],
    minLength: 2,
    maxLength: 50,
    default: 'My portfolio',
  },
  email: {
    type: String,
    required: [true, 'Please provide your email'],
    unique: true,
    lowercase: true,
    validate: [validator.isEmail, 'Please provide a valid email'],
  },
  role: {
```

```
  type: String,
  enum: ['user', 'admin'],
  default: 'user',
},
positions: [
  {
    type: mongoose.Schema.ObjectId,
    ref: 'Position',
  },
],
password: {
  type: String,
  required: [true, 'Please provide a password'],
  minlength: 2,
  maxlength: 50,
  select: false,
},
passwordConfirm: {
  type: String,
  required: [true, 'Please confirm your password'],
  validate: {
    // This only works on CREATE and SAVE!!!
    validator: function (el) {
      return el === this.password;
    },
    message: 'Passwords are not the same!',
```

```
    },
    minlength: 2,
    maxlength: 50,
  },
  passwordChangedAt: Date,
  passwordResetToken: String,
  passwordResetExpires: Date,
  active: {
    type: Boolean,
    default: true,
    select: false,
  },
});

userSchema.pre('save', async function (next) {
  // Only run this function if password was actually modified
  if (!this.isModified('password')) return next();

  // Hash the password with cost of 12
  this.password = await bcrypt.hash(this.password, 12);

  // Delete passwordConfirm field
  this.passwordConfirm = undefined;
  next();
});
```

```
userSchema.pre('save', function (next) {
  if (!this.isModified('password') || this.isNew) return next();

  this.passwordChangedAt = Date.now() - 1000;
  next();
});

userSchema.pre(/^find/, function (next) {
  // this points to the current query
  this.find({ active: { $ne: false } });
  next();
});

userSchema.methods.correctPassword = async function (
  candidatePassword,
  userPassword
) {
  return await bcrypt.compare(candidatePassword, userPassword);
};

userSchema.methods.changedPasswordAfter = function (JWTTimestamp) {
  if (this.passwordChangedAt) {
    const changedTimestamp = parseInt(
      this.passwordChangedAt.getTime() / 1000,
      10
    );
```

```
    return JWTTimestamp < changedTimestamp;
  }

  // False means NOT changed
  return false;
};

userSchema.methods.createPasswordResetToken = function () {
  const resetToken = crypto.randomBytes(32).toString('hex');

  this.passwordResetToken = crypto
    .createHash('sha256')
    .update(resetToken)
    .digest('hex');

  // console.log({ resetToken }, this.passwordResetToken);

  this.passwordResetExpires = Date.now() + 10 * 60 * 1000;

  return resetToken;
};

userSchema.pre(/^find/, function (next) {
  this.populate({
    path: 'positions',
```

```
    select: '-__v -owner',
  });
  next();
});
const User = mongoose.model('User', userSchema);
module.exports = User;
```

Файл схеми для ORM mongoose. Так як у проекті використовується ORM, то кожна модель має мати своє представлення. User має поля name, portfolioName, email, role, positions, password, passwordConfirm, passwordChangedAt, passwordResetToken, passwordResetExpires, active. На кожному полі,якщо потрібно, є своя валідація яку пропонує ORM.

Також в класі схеми прописані методи хешування паролю, перевірки на співпадіння при вході. Оскільки база даних в проекті нереляційна, то зв'язок між user та position виглядає у формі link Referencing, та при зчитуванні користувача автоматично на місце рядка ідентифікатора позиції підставляється об'єкт, взятий з колекції позицій за даним ідентифікатором.

Додаток В(L)

Investment Portfolio Web Application (IPWA)

Volodymyr Tkach

Version 1.0

June 2021

Revision History

NOTE: *The revision history cycle begins once changes or enhancements are requested after the initial version of the Software Architecture Document has been completed.*

Date	Version	Description	Author
01/06/2021	1.0	Initial version of SAD for comments by team	Volodymyr Tkach

Table of Contents

Software Architecture Document

Introduction

This document provides an overview and explains the implementation of the Investment Portfolio Web Application (IPWA). It explains how an online user will be able to create and use web application definitions and includes the architecture of the application.

PURPOSE

The Software Architecture Document (SAD) provides a comprehensive architectural overview of the Investment Portfolio Web Application (IPWA). It presents several different architectural views to depict different aspects of the system.

It is intended to capture and convey the significant architectural decisions which have been made on the system.

SCOPE

The scope of this SAD is to depict the architecture of the Investment Portfolio Web Application (IPWA) online application created by the student of KNU – 2021.

This document describes the aspects of Investment Portfolio Web Application (IPWA) design that are considered to be architecturally important Readers who require a technical understanding of the Investment Portfolio Web Application are encouraged to start by reading this document, then reviewing the UML model of the Investment Portfolio Web Application.

DEFINITIONS, ACRONYMS, AND ABBREVIATIONS

Node.js – JavaScript runtime platform

Express.js - Node.js web server framework

HTTP – Hypertext Transfer Protocol

JavaScript – scripting language

UIKit – CSS framework

JWT – JSON web token

Pug – template engine

MongoDB – norelational database management system

WWW – World Wide Web

SAD - Software Architecture Document

MongoDB Atlas – database hoisting platform

UML – Unified Modeling Language

User - This is any user who is registered on the website

Administrator – this user can read, modify and delete any of IPWA models. Administrator can delegate or share administrative rights to other users in the system.

Architectural Representation

Thanks for simplicity of creating MVC on express.js it's possible to create server-side rendering applications with all business logic into server. Also pure js and UIKit used to display user interface and send requests to server what is conveniently to update, read, delete and create data in server.

Architectural Goals and Constraints

Server side

IPWA will be hosted on one heroku platform, and connecting to one of the instance of database in MongoDB Atlas. All communication with client has to comply with public HTTPS, TCP/IP communication protocol standards.

Client Side

Users will be able to access IPWA only online. Clients/users are requiring using a modern web browser such as Mozilla Firefox 10, Internet Explorer 9, latest versions of Google Chrome or Safari. They should login/signup for use service. If someone forgot password, he/she can reset it, only provide email and would have reset link.

PERSISTENCE

Data persistence will be addressed using a no-relational database.

RELIABILITY/AVAILABILITY

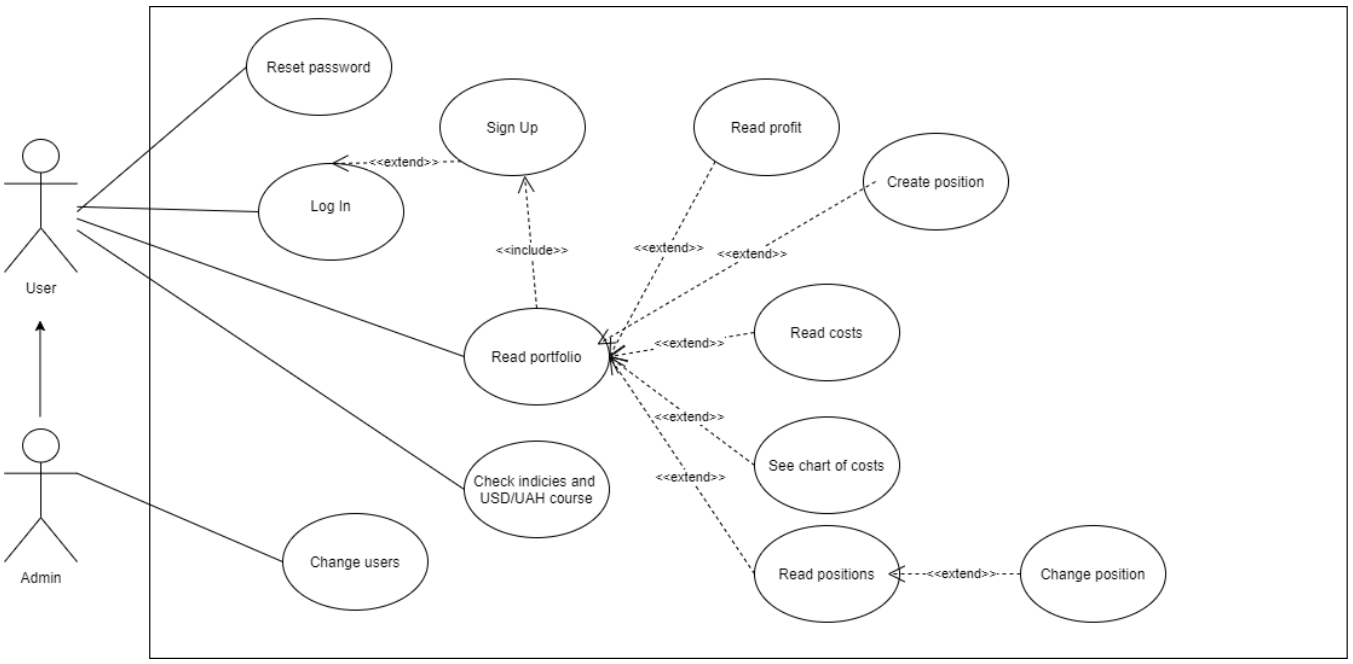
Reliability/Availability will be addressed through the MongoDB Atlas platform, which save backups and has snapshot options.

PERFORMANCE

There is no particular constrains related to system performance. It is anticipated that the system should respond to any request well under standard database and web server script timeouts (4 seconds), also system performance can depend on available hardware, PSU network and internet connection capabilities. In addition, upload / download times can depend on data size which in turn depends on user input. Therefore, actual performance can be determined only after system deployment and testing.

Use-Case View

This Use Case below illustrates the User registering for the first time within the application and the response given. It also illustrates the capabilities, if authentification passed. This Use case also illustrates the capabilities of administrator role.



ACTORS

As described in the actors' correspondence diagram below, web user could be one of two types:

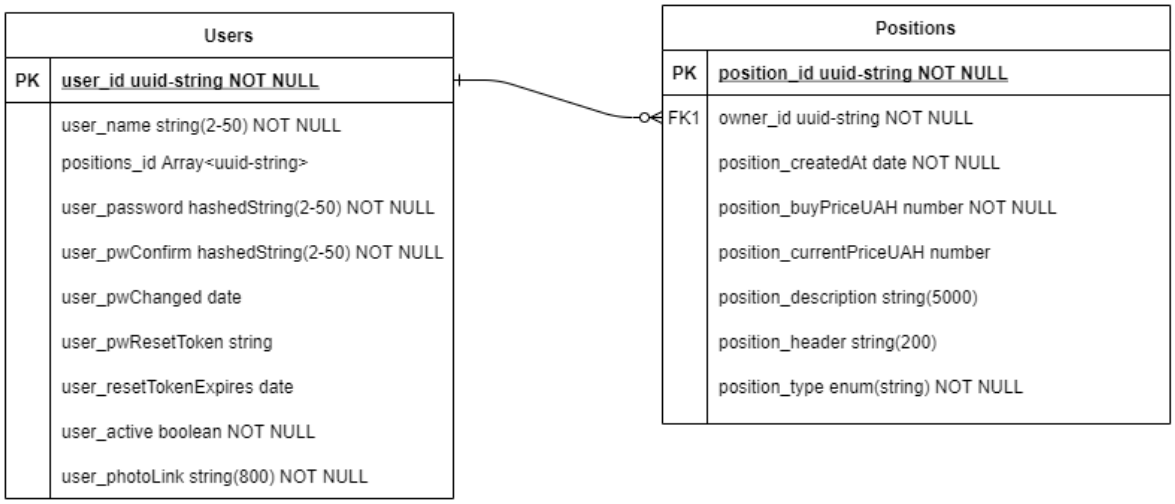
1. **Admin** has enhanced privileges to view, delete or download users.
2. **User** – could create, update, delete and read own positions.

Data View

The data view represents important part of the IPWA. Referencing (normalization) is selected as design approach of physical data model. Data consistency and quality are enforced through the series of ORM "Mongoose".

Data access will be provided only through the user web interface.

Nevertheless, the Data View structure will allow easy maintainable because all process complexity is hidden in the schema models, therefore creating or modifying process template will require minimum efforts.



Deployment View

IPWA tool deployment is provided by "Heroku" platform. It's free for deploying simple applications.

Size and Performance

Volumes

- Simultaneous users 10 000+(because of async aproach of node.js)
- Data storage under 1MB per user (including uploaded charts)

Performance

- With maximum load all CRUD-operations well under standard server script database connection timeout – 3 seconds.
- The system provides access to the information of the dataset in a few seconds.

Issues and concerns

- User authentication with JWT
- Charts (image) generating – is it feasible to generate chart on client side
- The data structure will allow creating many users with many positions in one portfolio