Міністерство освіти і науки України
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»
Кафедра мультимедійних систем факультету інформатики

Розробка веб-застосунку за допомогою фреймворку React з використанням можливостей cloud-платформи Google Firebase

Текстова частина до курсової роботи
за спеціальністю «Інженерія програмного забезпечення» - 121

Керівник курсової роботи
Ст. викладач
Борозенний С. О.
_____
(підпис)
“____” _____ 2021 р.
Виконав студент ІПЗ-4:
Марчук Р.Е.
“____” _____ 2021 р.

Міністерство освіти і науки України

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»

Кафедра мультимедійних систем факультету інформатики

ЗАТВЕРДЖУЮ Зав. кафедри мультимедійних систем,
к. ф.-м. н., доц.
_____ Жежерун О.П.
(підпис)
„____”_____2021 р.

ІНДИВІДУАЛЬНЕ ЗАВДАННЯ

на курсову роботу

студенту Марчуку Ростиславу Едуардовичу факультету інформатики 4 курсу

ТЕМА Розробка веб-застосунку за допомогою фреймворку React з використанням можливостей cloud-платформи Google Firebase

Вихідні дані:
Зміст ТЧ до курсової роботи:
Індивідуальне завдання
РОЗДІЛ 1: Аналіз завдання курсової роботи
РОЗДІЛ 2: Теоретичні відомості
РОЗДІЛ 3: Опис реалізації застосунку
Висновки
Список використаної літератури

Дата видачі „___” _____ 2021 р. Керівник _____
(підпис)
Завдання отримав _____
(підпис)

Календарний план виконання роботи:

Тема: Розробка веб-застосунку за допомогою фреймворку React з використанням можливостей cloud-платформи Google Firebase

№	Назва етапу	Термін виконання	Примітка
1.	Отримання теми курсової	17.10.2020	
2.	Пошук тематичної наукової літератури	26.11.2020	
3.	Ознайомлення з науковою літературою	29.12.2020	
4.	Розбір предметної області	10.01.2021	
5.	Перші спроби реалізації	21.01.2021	
7.	Визначення структури програми	23.02.2021	
8.	Написання 1 частини курсової роботи	12.03.2021	
9.	Написання 2 частини курсової роботи	20.03.2021	
10.	Написання 3 частини курсової роботи	28.03.2021	
11.	Написання висновків курсової роботи	01.04.2021	
12.	Перегляд змісту роботи з керівником	06.04.2021	
13.	Внесення змін до роботи	08.04.2021	
14.	Завантаження курсової роботи	12.04.2021	

Студент Марчук Р. Е.
Керівник Борозенний С. О.
“_____” _____

Зміст

Анотація

В цій курсовій роботі детально досліджено предметну область застосунків, що слідкують за витратами. Також зроблено огляд вже існуючих аналогів, проведено аналіз їх переваг і недоліків. В результаті створено новий застосунок з врахуванням недоліків вже існуючих аналогів, описана структура проекту, алгоритми і бібліотеки, що були застосовані при розробці.

Робота ілюструє процес розробки застосунку протягом усіх ключових етапів розробки: дослідження предметної області, планування функціоналу, визначення основних сутностей, що використовуватимуться, визначення компонентів для сутностей, які знадобляться для відтворення запланованого функціоналу, схематичне зображення макетів сторінок, що відповідатимуть запланованому функціоналу та визначеним сутностям, розробка компонентів, об’єднання компонентів у повнофункціональні сторінки, підключення бази даних до проекту, створення контекстів для тих частин застосунку, де вони потрібні, проведення тестування функціоналу.

В даній роботі використовувались такі технології: React 17, Material-UI 4, Firebase 8, VS Code IDE.

Вступ

Актуальність теми. В сучасному світі гроші – основа при ведені будь-яких справ. У будь-якої людини гроші знаходяться в постійному русі: купити в магазині їжу, заплатити комунальні послуги, заплатити за ремонт автомобіля і т.д. Це, звичайно, буденні витрати, які є у будь-якої людини, але деякі особи задаються ціллю документувати всі свої витрати, щоб в результаті проаналізувавши їх, звести кількість непотрібних витрат до мінімуму і, таким чином, економити гроші. Звичайно це можна робити в паперовому вигляді, записувати все в блокнот-зошит, таким чином, постійно вести облік витрачених коштів. Але змоделюємо ситуацію: Що, якщо такі витрати не у однієї людини, а у цілої організації, що має щомісячний бюджет, який надається вищим керівництвом компанії? Наскільки зручно тоді буде вести облік будь-яких витрат, якщо до бюджету має доступ кожен член компанії? Якщо в компанії 10-20 чоловік, то це вже незручно, але, загалом, можливо. А що, якщо в компанії декілька сотень працівників, як тоді це все документувати? У одного зламався принтер і потрібно викликати людину, щоб починити його і заплатити їй, інший скаржиться на відсутність кави/чаю на кухні, їх також потрібно купити, але що важливіше, потрібно ще пам’ятати про кожну з цих проблем. Так з’являється проблема: як спростити процес створення таких прохань (відремонтувати принтер, купити каву/чай на кухню), зробити його максимально швидким, зручним і при цьому позбутись паперової роботи? Вирішення даної проблеми знаходиться під рукою – смартфон. Використовуючи мобільний застосунок для відслідковування витрат, створення запитів на покупку необхідних речей вирішить всі проблеми.

Мета дослідження – провести дослідження ринку застосунків для слідкування за фінансами, визначити їх недоліки та переваги, визначити технології, що використовуватимуться при розробці та розробити додаток для ведення статистики витрат.

Для досягнення поставленої мети, необхідно реалізувати низку наступних завдань:

- дослідити існуючі фінансові застосунки
- їх аналіз, визначення функціоналу та функцій, що не вистачає для вирішення поставленої проблеми:
    1. створення запитів на покупку необхідних речей
    2. створення продуктів, що будуть купуватись регулярно – кожен місяць
    3. ведення журналу дій, що відбулись в застосунку

Об’єкт дослідження – застосунки для слідкування за фінансами, їх функціонал, переваги і недоліки.

Предметом дослідження популярні стеки технологій, що використовуються при розробці веб-застосунків, їх переваги та недоліки, аналіз сильних і слабких сторін вибраного варіанту стеку технологій.

Структура курсової роботи – вступ, 3 розділи, висновок, список використаної літератури.

Перший розділ описує загальні відомості про поставлену задачу, описується які болі вирішить цей застосунок. Також проведено детальний аналіз і опис предметної області. Визначено, типи застосунків, що використовуються для вирішення поставленої задачі. Окрім цього проаналізовано найбільш схожі вже існуючі

аналоги застосунків, що вирішують поставлену задачу і найкраще задовольняють визначені вимоги до застосунку.

У другому розділі описано в загальних рисах застосунок, який буде розроблено, як результат даної роботи, також розглянуто технології, що використовуватимуться при розробці застосунку, проаналізовано їх сильні сторони.

У третьому розділі описано детально всі етапи розробки застосунку: визначення основного функціоналу, планування дизайну майбутнього застосунку, опис ключових сутностей застосунку, детальний опис структури проекту – пакетів на, які розділений застосунок, їх призначення.

1. Аналіз завдання курсової роботи

1.1 Загальні відомості

Метою будь-якої організації і ФОП є отримання доходу. Цього можна досягти шляхом ведення фінансового контролю і під час цього процесу одразу з’явиться поняття витрати. У фінансовій системі завжди існують такі поняття як витрати, але це поняття може мати різні значення.

Витрати – це грошова оцінка вартості матеріальних, трудових, фінансових, природних, інформаційних та інших видів ресурсів на виробництво і реалізацію продукції за певний період часу [1].

Витрачена сума – це витрати певного періоду часу, документально підтверджені, економічно виправдані (обгрунтовані), повністю перенесли свою вартість на реалізовану за цей період продукцію[2].

Якщо порівнювати ці два визначення, то здається перше з двох трохи змістовніше. Крім того, будь які витрати включають: єдиний податок, гарантійний ремонт та інше. Для аналізу, обліку, планування витрати об’єднуються в обмежене число груп для узагальнення результатів обліку.

Керування витратами – це процес цілеспрямованого формування витрат за їх видами, місцями і носіями при постійному контролі і стимулюванні їх зменшення. Воно є важливою частиною економічного механізму будь-якого підприємства. Управління витратами підприємства є частиною системи управління підприємства в цілому.

Процес керування складається з декількох функцій - планування, організації, мотивації, контролю та координації. Поняття функцій менеджменту можна застосувати і до керування витратами на виробництво і реалізацію продукції.

Короткострокове планування передбачає складання кошторису витрат, який повинен бути заснований на поточних фінансових ресурсах, що є в розпорядженні організації. Метою стратегічного планування є досягнення бажаної мети в довгостроковому періоді за рахунок керування витратами.

1.2 Опис предметної області “Контроль фінансів”

На сьогоднішній день , ніхто не хоче відстежувати свої витрати, переглядаючи купи документів. Проте ведення обліку особистих фінансів є життєво-важливим, оскільки заощадження стало питанням виживання.

На даний момент проблему для бізнесу складає розробка програмного продукту (мобільний застосунок), який міг би допомогти користувачам і підприємцям вести облік фінансів.

Сьогодні програми для управління особистими фінансами є справжнім засобом для ефективного контролю витрат. Оскільки вони приносять значну користь користувачам і користуються великим попитом, розробка програми для управління грошима є гарною інвестицією. Багато з цих додатків містять різні пункти і варіанти вирішення проблем[3]:

1) Чат-боти та віртуальні помічники

Окрім банківських додатків, конкурентами вашого продукту - будуть чат-боти FinTech. Інтегровані з платформами обміну повідомленнями та банківськими програмами, або створені як самостійні програми, чат-боти та віртуальні помічники допомагають користувачам управляти своїми особистими фінансами. Вони відстежують фінанси, пропонують варіанти бюджетування.

Cleo, Digit - це лише кілька чат-ботів FinTech, про які слід знати.

2) Молода цільова аудиторія

Близько 26% населення світу до 15 років. І з кожним роком ця кількість збільшується. З цього приводу багато програм для управління особистими фінансами вирішують обслуговувати молодшу аудиторію. На ринку популярними стають програми, які навчають дітей та підлітків керувати власними фінансами. Наприклад: Greenlight, Plan’it Prom та BusyKid

3) Криптовалюта

На сьогодні можливість додатка підключати крипто-гаманці так само, як і банківські рахунки, є життєво-важливою для програми. Дозвіл користувачам відстежувати та управляти своїми криптовитратами стало конкурентною перевагою. Уже багато додатків дозволяють управляти балансами в Bitcoins.

4) Візуалізація даних

Красива і правильна візуалізація даних необхідна для бюджетування програми. Тому що саме це робить його привабливим для користувачів. Інфографіка, діаграми та інформаційні панелі привертають увагу та вражають. Крім того, відображення даних на

діаграмах та інфографіці робить інформацію зрозумілою для користувача

Отже, якщо ви потрібно створити успішну програму для керування фінансами, потрібно створити її з приголомшливою візуалізацією даних. Ознайомитись як приклад можна з програмою Airthings[4].

5) Першокласна безпека та відповідність стандартам

У будь-якому додатку слід захищати конфіденційні дані. Але безпека особливо важлива у програмі бюджетування, оскільки вона має прямий доступ до фінансових рахунків користувача. Високий рівень безпеки є обов'язковим для програми бюджетування.

6) Інтеграція з різними банками та платіжними системами

Очевидною перевагою програми для бюджетування є можливість пов'язати всі свої рахунки разом. Завдяки цій зручності додаток може в одному місці охоплювати та контролювати фінанси всіх користувачів. Однак, щоб це сталося, програма бюджетування повинна підтримувати інтеграцію з різними банками та платіжними системами.

7) Персоналізація

Персоналізація вже в тренді, і буде далі продовжуватись, пристосовуючи досвід відповідно до потреб користувача, програми для особистих фінансів стають привабливими для клієнта. Персоналізація програми не повинна закінчуватися простим відображенням історії транзакцій у прекрасній інфографіці. Потрібно зробити додаток з можливістю налаштувати різними способами, наприклад, створити корисний чат-бот.

Отже , програмний продукт у вигляді веб додатку повинен автоматизувати процес аналізу та обліку особистих фінансів користувачів і підприємств.

В основі застосунку може лежати економічна модель розподілення бюджету «Метод глечиків». Але трішки змінена, тепер її сенс буде полягати у тому , щоб кожного місяця бюджет користувача розподілявся на категорії за сферами життя і користувач може сам їх обрати. Хоч і існує багато категорій які є перш необхідними такі як:

1) Щоденні (регулярні) витрати. До них входить наступне: сплата комунальних послуг, оренди, виплата кредиту, харчування. На ці потреби виділяється 60% усього бюджету.

2) Розваги. Сюди включають різне обладнання. На розваги виділяють частину бюджету, яку зазвичай користувачі можуть собі дозволити витратити на будь-що.

3) Освіта. На ці гроші ви можете придбати курси, книжки, абонемент у спортзалу, загалом, те, що допоможе вам покращити свої вміння та навички. Складає близько 5% бюджету.

4) Резерв. Його можна витрачати на великі покупки

5) Подарунки та благодійність, теж приблизно 5% бюджету.

Але коли є змога для кожного визначити які саме категорії потрібні, це зробить додаток більш універсальним у використанні, та буде можливість налаштувати персонально під себе.

Хоч і «Метод глечиків» вважається як економічна модель, яка забезпечує рівномірне забезпечення фінансами в різних сферах

життя і ефективність методу підтверджена шляхом тестування, її завжди можна спробувати покращити.

1.3 Огляд і аналіз існуючих аналогів, що реалізують функції

Новий додаток для особистих фінансів повинен бути готовим для користування в конкурентному середовищі. Тому перед запуском і навіть розробкою мобільного додатка, потрібно ретельно дослідити та вивчити своїх конкурентів (таблиця 1.1), щоб уникнути можливих помилок під час розробки та виявити функціональні потреби, які повинна реалізовувати майбутня програма.

Найближчими конкурентами даного продукту є інші програми для керування особистими фінансами. Хоча вони можуть відрізнятись за функціоналом та спрямованістю, все ж вони полюють на ту саму аудиторію та частину ринку.

Фірма-розробник	“CB Mobile Ltd”	“Aimbity AS”
Назва	Monefy	Moneon
Версія продукту	1.3.3	6.0.3
Функціональність	Контроль витрат ,створення звітів ,підтримка декількох гаманців, синхронізація з хмарним сховищем	Контроль доходів та витрат, підтримка декількох гаманців та бюджетів, можливість створити власну категорію витрат,оформлення звітів

Інтерфейс програми	Інформативний, але не зовсім очевидний для нового користувача	Інформативний, містить декілька вкладок, що робить його більш зрозумілим
Допомога користувачу	Є підказки, рекомендації щодо використання	Існує лише процес “Демо-інструкції”, тех-підтримка

Таблиця 1.1 – Порівняльна характеристика програмних продуктів-конкурентів

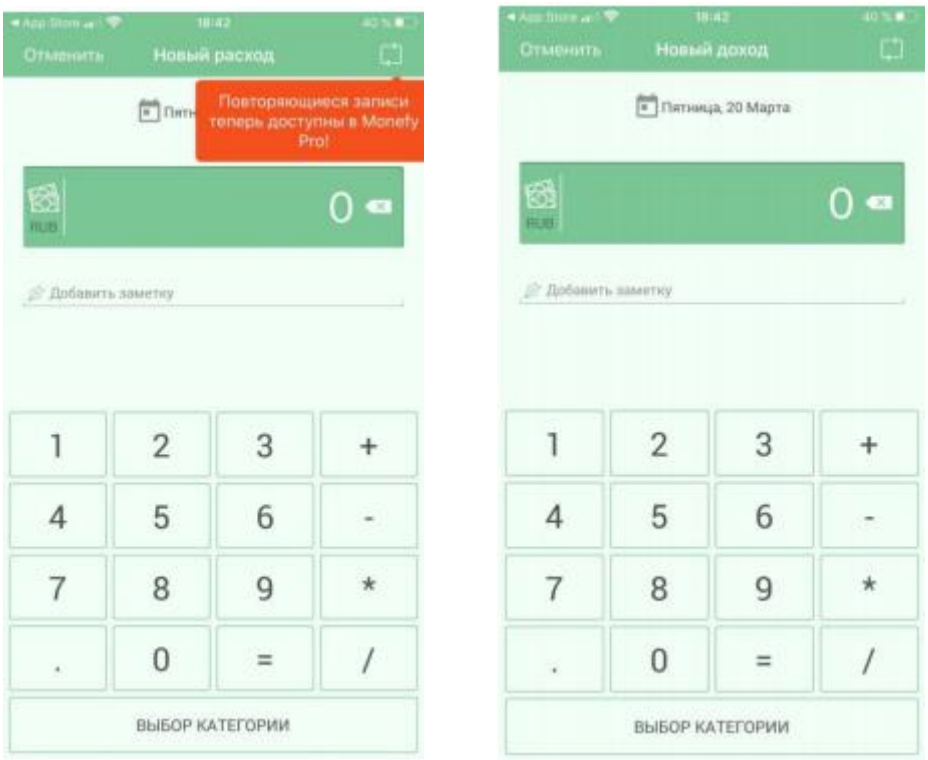
Об’єктами для дослідження конкурентних продуктів стали два мобільні додатки: «Monefy» та «Moneon».

Monefy – відносно новий додаток, котрий функціонально відповідає потребам описаним у проблематиці предметної області.

Головне меню програми (рис. 1.1) досить просте та інформативне, тут можна побачити загальний стан фінансів (порівняння доходів та витрат) та кнопки внизу для додавання доходів та витрат. Проте інтерфейс є не зовсім інтуїтивно зрозумілим, оскільки тут є багато кнопок навколо діаграми, і не зрозуміло для чого вони призначені та за що відповідає кожна кнопка.

Малюнок 1.1 – Головне вікно програми Monefy

Вікна додавання записів (рис. 1.2) мають мінімалістичний дизайн, як і вся програма, та є інтуїтивно зрозумілими для користування. Тут можна ввести новий запис, обравши суму для вводу, дату (за замовчуванням – сьогодні) та категорію. Також можна додати короткий опис запису.



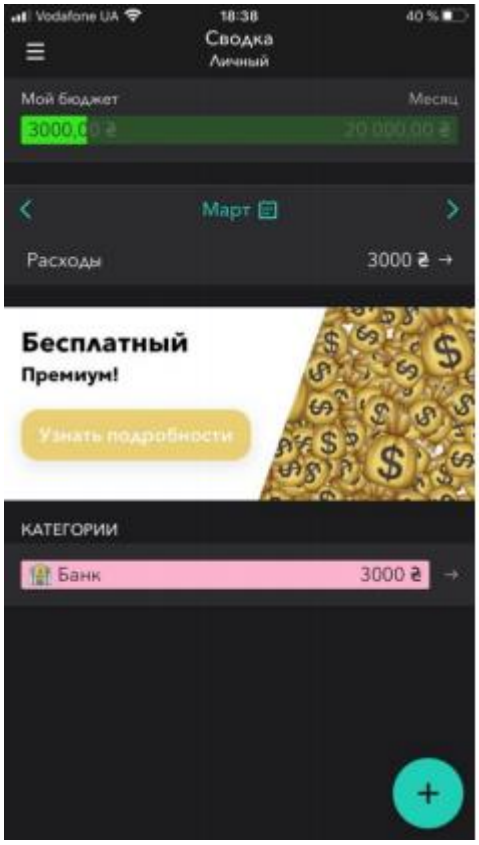
Малюнок 1.2 – Вікна додавання витрат та доходів

Також є декілька інформативних вікон, таких як перелік категорій, рахунків та статистика за обраним періодом (рис. 1.3)

Малюнок 1.3 – Інформативне вікно з категоріями

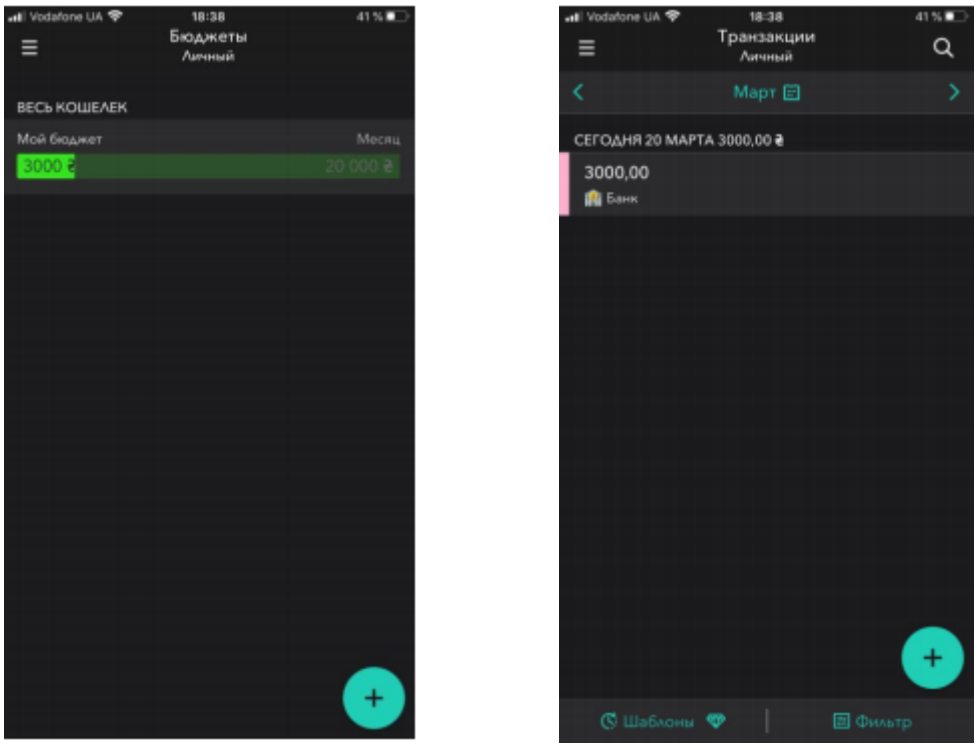
Даний додаток є досить вдалим, але зі своїми недоліками, такими як іноді не зрозумілий дизайн (підказки лише допомагають) та неможливість отримати звіт, що є основною потребою при обліку фінансів.

Moneon – мобільний додаток, який вже існує на ринку та є найбільш вдалим із подібний йому продуктів. Має майже весь потрібний функціонал для обліку. Ця програма не має визначеної головної сторінки, оскільки користувач може у налаштуваннях вибрати вікно, яке буде зустрічати його при вході у застосунок. За замовчуванням це вікно «Сводка», де відображений обраний користувачем гаманець для доходів та процес витрат по цьому гаманцю(рис. 1.4)



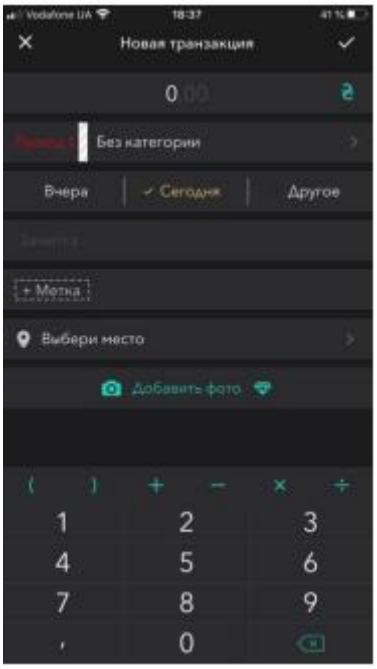
Малюнок 1.4 – Головне вікно «сводка»

Також серед вікон програми є «бюджети» та «транзакції», у котрих відображається відповідні дані та де можна додати новий запис до цих вікон (рис. 1.5)



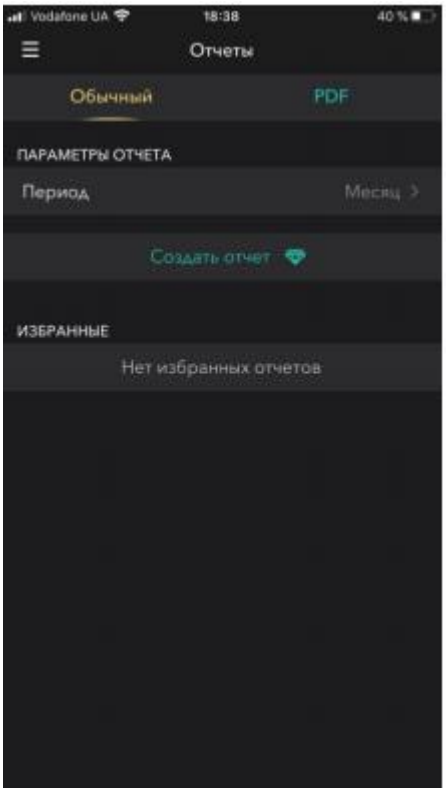
Малюнок 1.5 – Вікна «бюджети» та «транзакції»

Вікна із додаванням нових записів (рис. 1.6) є зручними та зрозумілими. Тут можна обрати необхідну суму для вводу, категорію, за якою буде записано витрати (категорій багато та можна додати власну), дату коли була зроблена витрата, додати короткий опис запису та фотографію чека, котрий буде зберігатись у даній програмі.



Малюнок 1.6 – Вікно додавання витрат

Також тут є дуже важлива частина функціоналу отримання звіту за обраним періодом та типом файлу (рис. 1.7).



Малюнок 1.7 – Вікно оформлення звіту

Даний програмний продукт є найбільш відповідним до вимог предметної області, проте, одним із його недоліків, і найбільш це те, що більшість функцій цієї програми можна використовувати тільки у платній версії додатку, до того ж і з досить високою платою в місяць.

Досліджені програмні продукти можуть допомогти у розробці мобільного додатку як функціональними ідеями, так і у створенні інтуїтивно зрозумілого та зручного інтерфейсу. Проаналізувавши програмний продукт «Moneon», було звернено увагу на структуру звіту та процес його формування та зручний інтерфейс додатку, що буде враховано при розробці власної програми.

2. Теоретичні відомості

2.1 Загальний опис програми, що потребується

Завдання даної роботи – створити мобільний веб-застосунок з використанням можливостей cloud платформ, для максимально зручного розгортання, великого вибору технологій для розробки та реалізації певної частини функціоналу застосунку на стороні cloud платформи.

Даний застосунок створюється для покриття потреб компаній у зручному відслідковуванні витрат, веденні статистики, надання можливості звичайному співробітнику запропонувати керівництву придбати певне покращення в офіс, необхідно розробити додаток, що дозволяє зручно і швидко робити все вище сказане. Програма розрахована на роботу з багатьма користувачами одразу, тому повинна бути централізована. Задля вирішення проблеми зручності вирішено робити мобільний застосунок.

Також потрібно реалізувати систему ведення журналу дій в застосунку: будь-яке додавання нового запиту в застосунку, видалення, абощо – буде записуватись до цього журналу і його зможе переглянути адміністратор застосунку в будь-який момент.

Ще однією особливістю застосунку будуть оповіщення, що будуть приходити:

- користувачу, коли його запит на покупку чогось нового буде схвалено;
- адміністратору, коли його назначено на покупку будь-якого продукту по запиту;

- адміністратору, коли новий користувач реєструється в застосунку;
- адміністратору, коли настає час купувати регулярний продукт.

Також слід вказати, що користувач зможе зареєструватись в застосунку, лише по запрошенню, яке висилає адміністратор на електронну пошту і реєструватись користувач зможе лише використовуючи пошту на яку прийшло запрошення.

2.2 Переваги React.js

React – дуже зручний JavaScript фреймворк для розробки Single-page application. Завдяки зручному інтерфейсу і величезній кількості можливостей, що надає даний фреймворк, нескладно розробляти багатофункціональні застосунки з можливістю масштабувати їх надалі та розширювати, додаючи новий функціонал.

Переваги React полягають у декількох факторах:

- React використовує Virtual DOM, а це підвищує продуктивність застосунку та зменшує шанс виникнення незручностей для користувачів під час користування застосунком
- React працює особливим способом, що підвищує швидкість відмальовування сторінок і завдяки цьому, користувачам набагато зручніше працювати з React застосунками.
- В React розповсюджена практика code reusability, що полягає у повторному використанні вже написаних компонент, при цьому лише додаючи новий потрібний

функціонал, як інтерфейс вже написаного раніше компонента.

2.3 Переваги Google Firebase

Firebase – це платформа для розробки мобільних застосунків від компанії Google, в якій є всі сучасні функції для розробки, перекомпонування та покращення застосунків.[5]

Основними перевагами Firebase є:

- Безкоштовне користування обмеженим функціоналом. Для користування сервісами Firebase не потрібно купувати підписку, більшість сервісів доступні безкоштовно, але з обмеженою кількістю добових запитів до цих сервісів. Для користування Firebase, потрібно лише увійти в Firebase консоль за допомогою Google акаунта.
- Різноманітність сервісів. Firebase надає всі, необхідні при розробці сервіси. Наприклад для вирішення проблеми баз даних, Firebase надає такі сервіси, як Firestore та Realtime Database; для вирішення проблеми зберігання мультимедіа-даних, є Firebase Storage; для авторизації використовується Firebase Authentication, що надає можливість авторизації за допомогою акаунтів в безлічі відомих сервісів (Google, GitHub, Facebook, Twitter, і т.д.).
- Розробка застосунків без серверної частини. Firebase надає середовище, в якому не потрібно будувати серверну архітектуру, займатись балансуванням навантаження і всілякими іншими проблемами, що виникають на стороні сервера при використанні клієнт-серверної архітектури.

Адже використання сервісів Firebase базується на основі запитів.

- Покращена індексація застосунків. Google розробили Firebase таким чином, що всі застосунки створені з використанням Firebase сервісів, індексуються за всіма найкращими методиками. З цього випливає, що пошукова система краще ранжує документи, сайту, що задеплоєний на Firebase Hosting і використовує Firebase сервіси. Добре індексовані документи відображаються вище в результатах пошуку, а це, як результат дає постійний збільшення клієнтської бази застосунку.
- Захист від втрати даних. Google попіклувались, щоб клієнти Firebase платформи мали резервні копії на випадок втрати даних, тому в них є функція резервного копіювання бази даних, яку можна ввімкнути в будь-який момент і це допоможе зберегти дані у випадку непередбаченого видалення даних з основної бази.

3. Опис реалізації застосунку

3.1 Визначення функціоналу майбутнього застосунку

Під час аналізу предметної області, визначення проблем, з якими зустрічається цільова аудиторія, було визначено наступні вимоги до майбутнього функціоналу застосунку:

- В застосунку обов’язково має бути 2 ролі:
    1. Адміністратор, що здійснюватиме покупки, буде визначати чи варто купляти певну забаганку користувача, буде мати доступ до статистики і журналу подій в застосунку, займатиметься менеджментом користувачів;
    2. Користувач, що зможе додавати свої побажання, що потрібно купити, слідкуватиме за підтвердженням від адміністратора.
- Авторизація відбуватиметься за допомогою Google Auth, з використанням Google аккаунта.
- Реєстрація відбуватиметься лише за допомогою запрошень, що присилатимуться на електронну пошту, адміністраторами.
- У адміністратора буде наступний функціонал:
    1. Створювати/редагувати/видаляти категорії товарів
    2. Створювати/редагувати/видаляти регулярні продукти, що купуються щомісяця. Н-д: оплата інтернету, оренди офісу і т.д.
    3. Створювати сповіщення-нагадування, що потрібно купити регулярний продукт, яке буде автоматично

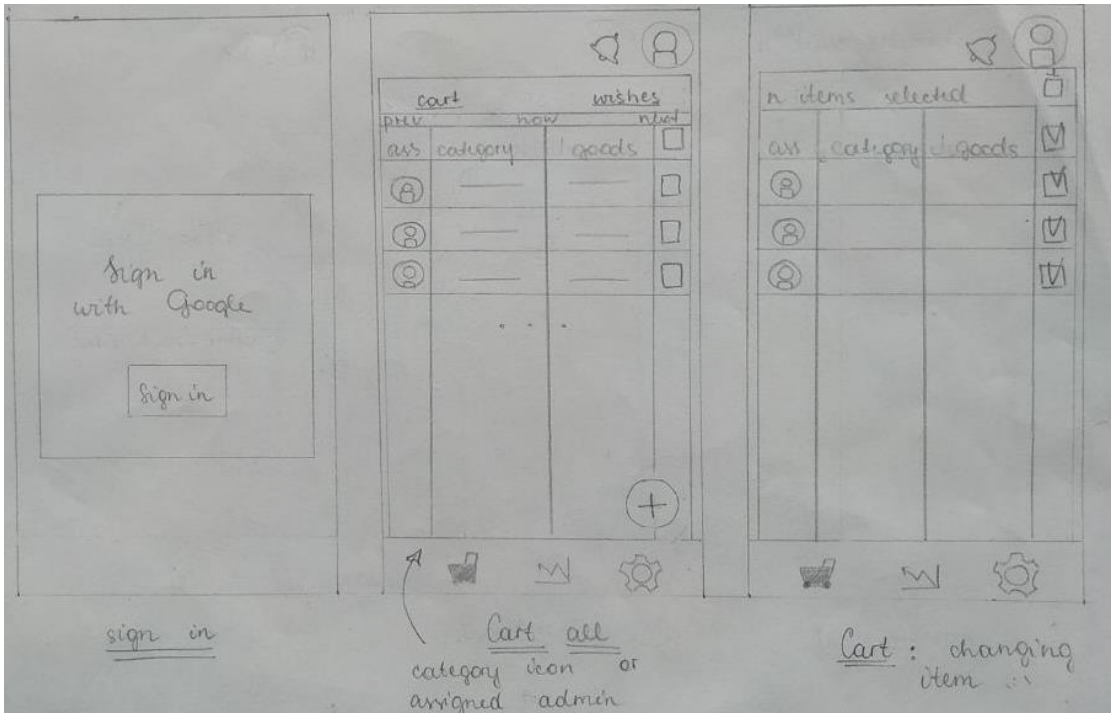
приходити в день, коли потрібно його купити щомісяця.

4. Проглянути статистику за: день, місяць, рік; по категоріям; по користувачам: хто, скільки витратив коштів за певний період.
5. Додати в список покупок затверджену забаганку звичайного користувача, при цьому вибрати адміністратора, котрий купуватиме його, та відредагувати додаткову інформацію про продукт.
6. Створювати гаманці, що відображатимуть виділений бюджет, на даний період.
7. Отримувати сповіщення всередині застосунку, на певні дії
8. Переглядати журнал дій.
9. Додавати власні побажання в список забаганок
10. Писати коментарі до продуктів.

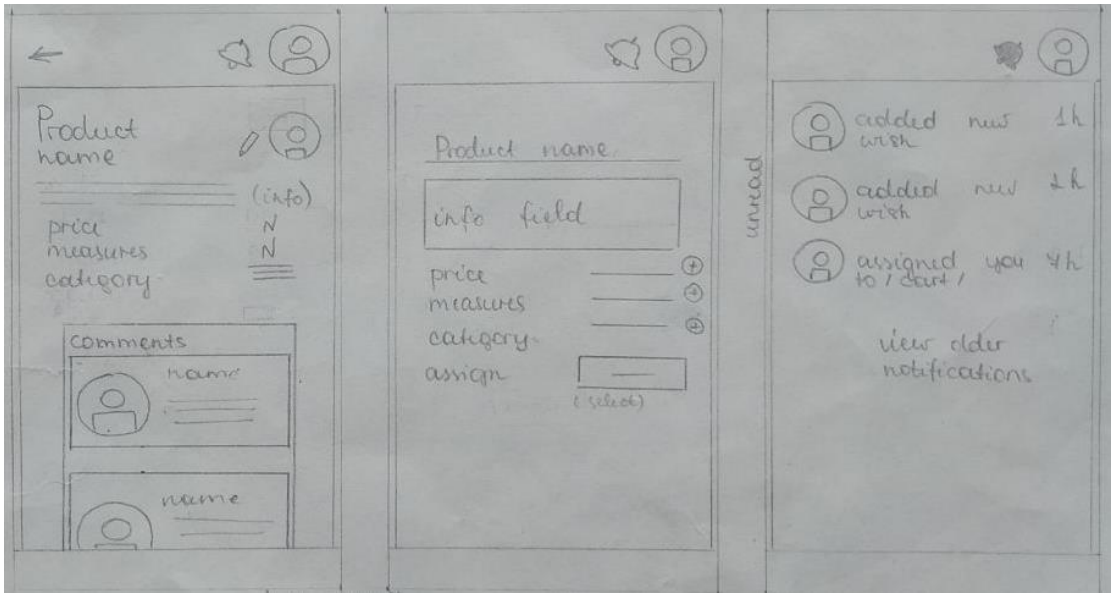
- Користувач матиме такі можливості:
    1. Авторизуватись в застосунку, у випадку, якщо користувача було запрошено до застосунку.
    2. Створювати запит на покупку якогось продукту, додаючи відповідний продукт до списку забаганок
    3. Переглядати список вже існуючих забаганок, та детальну інформацію про забаганки.
    4. Переглядати коментарі, що були написані до забаганок і додавати власні коментарі.

3.2 Проектування макетів застосунку

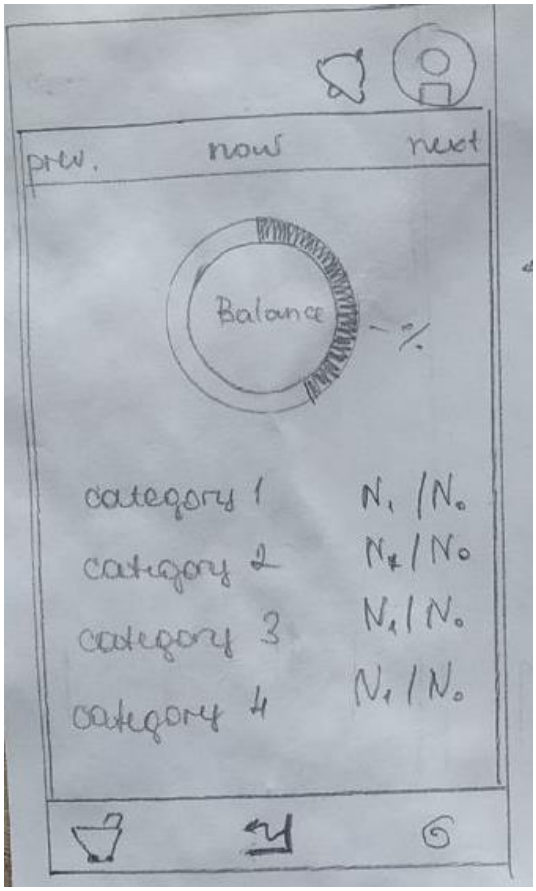
В результаті аналізу функціоналу, що має бути реалізований, було накидано наступні початкові макети сторінок, що задовольняють функціонал визначений в технічному завданні:



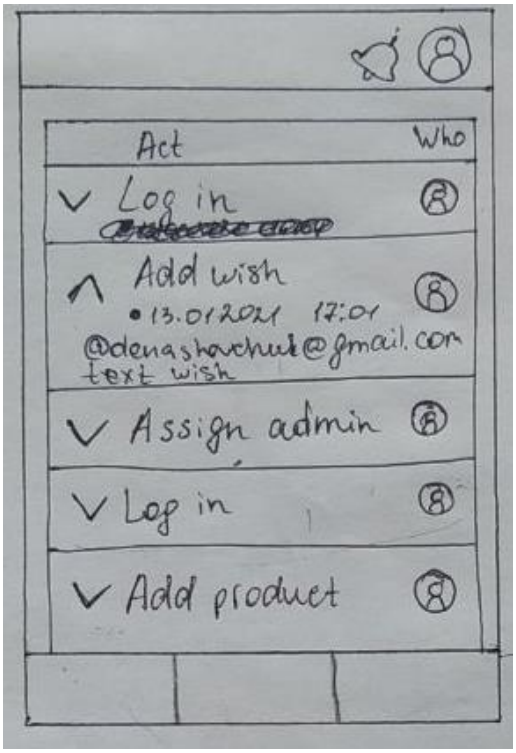
Малюнок 3.1 Макети сторінок Авторизація та Список Покупок



Малюнок 3.2 Макети сторінок Продукт(Детальніше), Створення/Редагування Продукту та Сповіщення



Малюнок 3.3 Макет сторінки Статистика



Малюнок 3.4 Макет сторінки Журнал дій

3.3 Визначення структури даних і сутностей

Firebase firestore – належить до класу нереляційних баз даних. Відповідно до цього, непотрібно буде створювати ER та R моделі бази даних, все що потрібно – визначити сутності, та поля, що будуть якісно відображати сутність. Будь-яка сутність буде прив’язана до компонент, а компоненти в свою чергу сформують сторінки. Оскільки сутності – це об’єкт навколо якого створюватиметься функціонал, тому було визначено наступні сутності, опираючись на визначений раніше функціонал:

1. Category – сутність, що відображатиме категорії товарів, поля:
   - budget – бюджет, що було виділено для даної категорії;
   - colorCategory – назва кольору категорії;
   - nameCategory – ім’я категорії;
   - spent – кількість коштів, що було витрачено на дану категорію, за даний період;

2. Comment – сутність, що відображає коментарі додані користувачами до товарів, поля:

   - date – дата, коли було написано коментар;
   - productId – id-посилання на продукт, до якого був написаний даний коментар;
   - text – текст коментаря;
   - userId – id-посилання на користувача, що написав даний коментар;

3. Log – сутність, що відображає запис у журналі дій в застосунку, поля:

- action – текст з типом дії, яка відбулась;
- date – дата, коли дія відбулась;
- description – детальніший опис дії, що відбулась;
- userId – id-посилання на користувача, що здійснив дану дію;

4. Notification – сутність, що відображає сповіщення, що приходитимуть на певні дії в застосунку, поля:

- date – дата, коли спрацювало сповіщення;
- text – текс сповіщення;
- userId – масив id-посилань на користувачів, яким дане сповіщення адресується;
- viewed – мапа, що відображає, який користувач з тих, кому адресоване дане сповіщення, переглянув його;

5. Purchases – сутність, що відображає куплені продукти і додаткову інформацію про них, поля:

- userId - id-посилання на користувача, що придбав даний продукт;
- category – категорія, до якої належав даний продукт;
- date – дата, коли продукт було придбано;
- name – назва продукту, що було придбано;
- price – ціна за одиницю продукту;
- quantity – кількість продукту, що було придбано;
- measure – одиниця виміру продукту;
- walletId – id-посилання на гаманець, яким розрахувались за продукт;

6. RegularProduct – сутність, що відображає регулярні продукти і додаткову інформацію про них, поля:

- assignId – id-посилання на користувача, якого призначили відповідальним за оплату даного продукту;
- category – категорія, до якої належить даний продукт;
- description – детальніший опис продукту;
- measure – одиниця виміру продукту;
- price – ціна за одиницю продукту;
- quantity – кількість продукту, що потрібно придбати (в одиницях виміру);
- name – назва продукту;
- remind – дата-нагадування, коли потрібно придбати продукт;

7. User – сутність, що відображає користувача застосунку, інформацію про нього, його роль в застосунку, поля:

- avatar – посилання на аватар користувача;
- birthday – дата дня народження користувача;
- email – адреса електронної пошти користувача;
- firstName – ім’я користувача;
- surname – прізвище користувача;
- phone – номер телефону користувача;
- role – роль користувача в застосунку;

8. Wallet – сутність, що відображає інформацію про гаманець, поля:

- balance – поточний баланс гаманця;
- currency – валюта, в еквіваленті якої знаходиться баланс гаманця;
- name – назва гаманця;

Отож, даний набір сутностей – це стартовий мінімум, що знадобиться для покриття визначеного раніше функціоналу, це не фінальна версія, в майбутньому вона може змінюватись, але при розробці даної версії застосунку, сутності будуть саме такими.

3.4 Опис структури застосунку

Застосунок складається з наступних пакетів:

- components
- config
- constants
- contexts
- domains
- hooks
- services

Кожен з цих пакетів відповідає за частину логіки застосунку. Далі наведено детальніший опис цих пакетів:

1. Components – пакет, в якому знаходяться базові компоненти застосунку. Базові компоненти – це ті, які не мають семантичної прив’язаності до певної сутності, можуть застосовуватись в застосунку багато раз , н-д: Header – «шапка» застосунку, компонента, що буде на кожній сторінці, але при цьому не прив’язана до певної сутності.
2. Config – пакет, в якому знаходяться конфігураційні дані проекту. Застосунок розроблено таким чином, що він опирається на загальні конфігурації, які записані в файлах даного пакету. Таким чином забезпечується гнучкість застосунку, адже лише трохи змінивши дані в файлі даного пакету, зміниться вигляд цілого застосунку. Н-д: файл theme.js

надає конфігурацію стилю застосунку, його теми. Змінивши значення однієї змінної primaryColor, ми змінимо основний колір теми по всьому застосунку.

3. Constants – пакет, в якому знаходяться константи проекту, такі як: конфігурація з'єднання з базою даних, routes і т.д.
4. Contexts – пакет, в якому знаходяться специфічні сутності – контексти. Контекст дозволяє передавати дані через дерево компонентів без необхідності передавати props на проміжних рівнях[6]. Така сутність допомагає передавати, наприклад, дані сесії користувача в будь-яку частину застосунку і оперувати ними всередині компоненти, при цьому не доводиться вдаватись до props-drilling.
5. Domains – пакет, в якому знаходяться компоненти прив'язані до сутностей бази даних. Ці компоненти згруповані навколо кожної сутності, а їх кількість і типи залежать від функціоналу, що потрібен сутності. Н-д: У сутності product є компонент AdvancedView – що відповідає за відображення детальної інформації про продукт, також є компонент AdvancedForm, що відповідає за форму створення/редагування продукту. Таким чином ці дві компоненти покривають функціонал, що вимагається від сутності Product.
6. Hooks – пакет, в якому знаходяться спеціальні функції – хуки. Хуки - нововведення в React 16.8, яке дозволяє використовувати state і інші можливості React без написання класів[7]. У React немає способу «приєднати» повторно використовується поведінку до компоненту (наприклад, підключення до бази даних, або стягування даних з бази). Хуки дозволяють вам повторно використовувати логіку state, не зачіпаючи дерево компонентів[8].

7. Services – пакет, в якому знаходяться функції-обгортки для запитів до бази даних. Ці функції скорочують написання всіх запитів, які використовуються, а також замінюють результат запиту на масив об’єктів а не Promise (за замовчуванням, результат будь-якого запиту до firestore – Promise).

Висновки

Результатом даної роботи, став застосунок для зручного відслідковування, менеджменту, та пропозицій від звичайних співробітників щодо витрат організації.

Застосунок було розроблено з урахуванням найкращих методів розробки комплексного mobile web application. Було застосовано всі переваги фреймворку React, чітко розпланована вся структура згідно з особливостями даної технології. Також при розробці дотримувались всі принципи SOLID.

Зі сторони back-end частини застосунку, було використано всю потужність платформи Google Firebase, що і допомогло реалізувати частину комплексного функціоналу, такого як: реєстрація користувачів за допомогою запрошення на електронну пошту; Система щомісячних сповіщень на покупку регулярного продукту; Видалення користувача з застосунку, при цьому і видалення його авторизаційних даних з Firebase Google Authenticator.

Список використаної літератури:

1. М.Н. Кондратьева. Экономика предприятия : учебное пособие / М.Н. Кондратьева, Е.В. Баландина. – Ульяновск: УлГТУ, 2011. – 174 с.

2. Игуменников А.С. Различие понятий «затраты» и «расходы». // Молодой ученый, Март 2015. – № 5 (85). – с. 275-277.

3. [Електронний ресурс] - Режим доступу: https://relevant.software/blog/personal-finance-app-like-mint/

4. [Електронний ресурс] - Режим доступу: https://budgetbakers.com/

5. [Електронний ресурс] - Режим доступу: https://firebase.google.com

6. [Електронний ресурс] - Режим доступу: https://ru.reactjs.org/docs/context.html

7. [Електронний ресурс] - Режим доступу: https://ru.reactjs.org/docs/hooks-faq.html

8. [Електронний ресурс] - Режим доступу: https://ru.reactjs.org/docs/hooks-intro.html#motivation