# SWE 417

# Team 15

# MyRewards

# Software Requirements Specification

# Document

# Document Revision History

| Date | Version | Description | Author |
|---|---|---|---|
| <10/06/22> | <1.0> | Specify Project functional requirements | Ali Alaql |
| <10/21/22> | <2.0> | Specify Project use cases description | Ali Albannay |
| <1/2/23> | <3.0> | Finalizing the Document | Mohammed M.Saleh |
| | | | |

# Table of Contents

# Introduction

MyRewards is a smartphone app that will allow consumers to earn rewards and discounts for their regular purchases. Customers will benefit from this since they will be able to obtain such discounts from their regular purchases. In this SRS document, we will present the system requirements, system features, system diagrams, how the system interacts with each component, and system non-functional requirements.

## 1.1 Purpose

Our project aims to establish a relationship between the user and the business owner in which the user can earn points by interacting with the system and converting them into coupons, receive discounts and rewards for their daily spending, and receive a reward for analyzing their data. Additionally, it will make it possible for business owners to quickly construct a loyalty reward system in order to employ a pull and push marketing campaign. In addition, it will enable business owners to establish a richness of reaching customers in order to provide them with products and services based on their requirements.

The SRS is for analyzing and showing the normal flow of features, using case diagrams, and diagrams for features to show how these features react with each other. The SRS will help all stakeholders to be able to know the flow of the system and how it will be working.

## 1.2 Scope

The project will provide the user set of chances to spend less on purchasing and managing his/her spendings. While the business owners will get the chance to study the charts that will be provided to come up with better plans that can improve the business performance.

## 1.3 Definitions, Acronyms and Abbreviations

| **Term /Abbreviation** | **Explanation** |
|---|---|
| **SRS** | Software Requirement Specifications |
| **SDD** | Software Design Document |
| **API** | API Application Programming Interface |
| **UI** | User Interface |
| **DBMS** | Database Management System |
| **DSS** | Data Storage System |
| **NLP** | Natural language processing |
| **AI** | Artificial Intelligence |
| **ML** | Machine Learning |
| **JSON** | JavaScript Object Notation |
| **SMS** | Short message service |
| **GPS** | Global Positioning System |
| **QR** | Quick Response code |
| **BI** | Business Intelligence |
| **BO** | Business owner |

## 1.4 References

## 1.5 Overview

The SRS document, which consists of 3 parts, will outline and emphasize the primary functions of our system. It starts with an Introduction, which provides a general overview of the document and our system.

The Overall Description is primarily intended for the customer's consumption. It includes a brief explanation of the system's operation, User Characteristics, and a class diagram that shows how the system is organized.

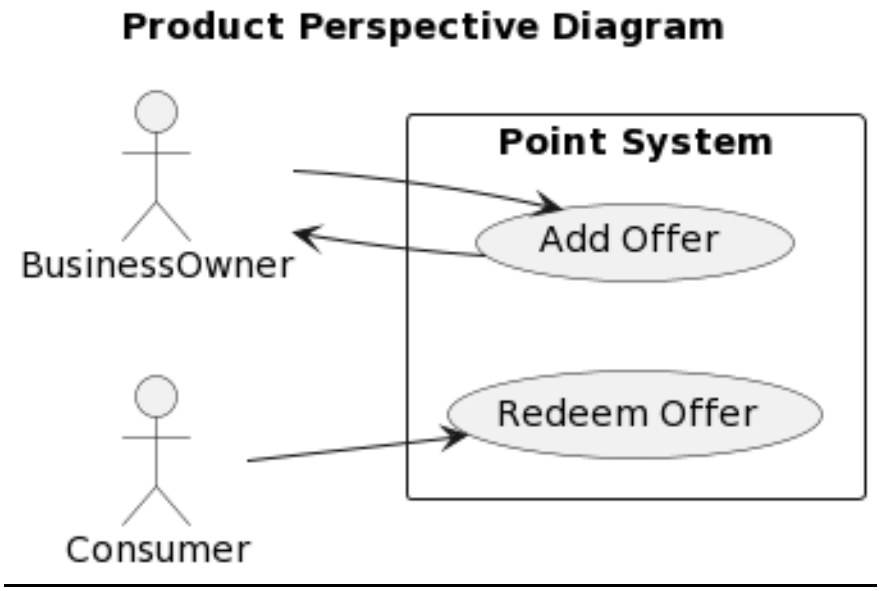
The last section, Specific Requirements, is mostly intended for developers and delves deeper into the system's functionality, providing specific technical descriptions using Use case diagram and Sequence diagram. Also, prototype system interfaces and non-functional requirements.

## 2. Overall Description

This section, which is written in the customer's language, contains a brief summary of various aspects of the project. The section will illustrate what systems our product interacts with, an overview of the system's functionality, how the classes interact with one another, user characteristics, and an outline of the non-functional requirements.

### 2.1 Product Perspective

**Figure 1:Product Perspective**




| ✕✓ | MyRewards | Drahim |
|---|---|---|
| Analyze your expenses | ✓ | ✓ |
| Managing spending | ✓ | ✓ |
| Getting rewards from purchasing | ✓ | ✕ |
| Giving business owners to make a marketing campaign | ✓ | ✕ |

## 2.2 Product Functions

The primary goal of the system is to provide a platform for customers to accumulate points by making purchases from the participating businesses, and to redeem the points for rewards in the form of offers provided by the businesses.

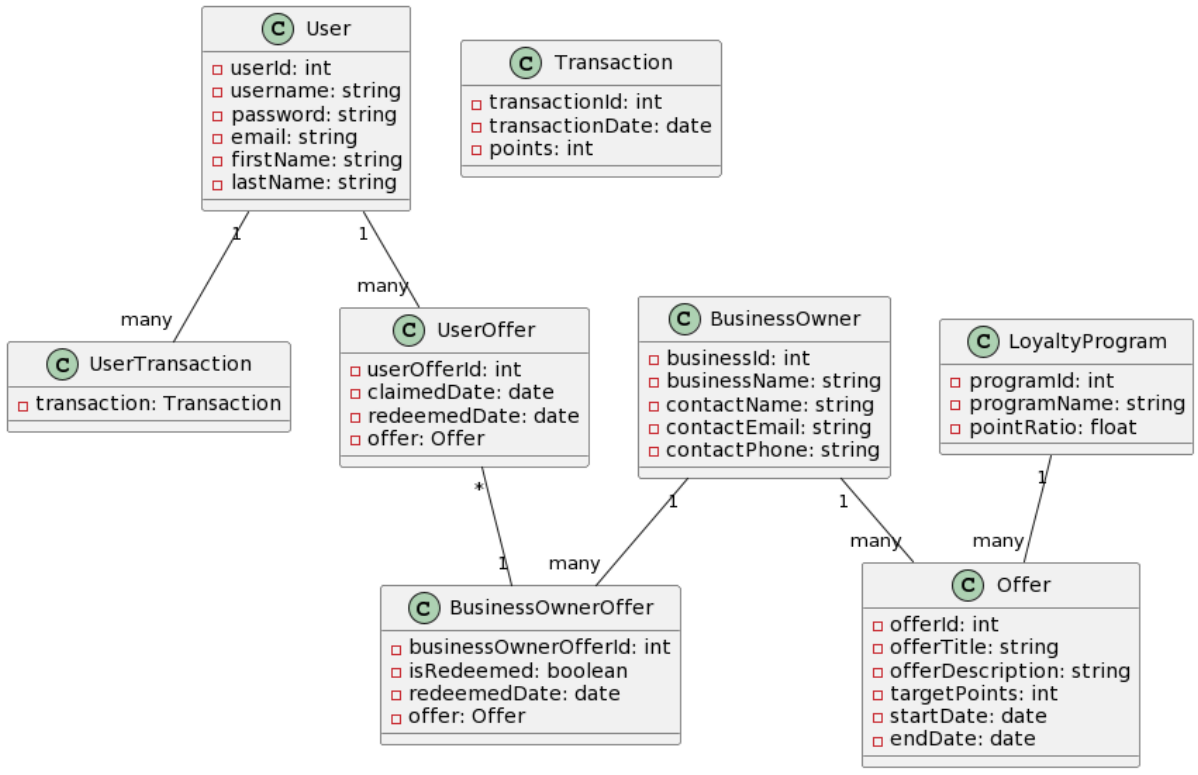
The system should allow business owners to create and manage their offers, set points thresholds for their customers to achieve, and track the performance of their offers. The system should also enable business owners to view customer profiles, transactions, and points histories.

Customers should be able to view their point balance and redeem points for available offers. Additionally, customers should be able to view and track their transaction histories, as well as update their profile information.

The system should also provide administrators with the ability to manage user accounts, handle technical issues, and generate reports on the system's performance.

Overall, the product functions should ensure that the system provides a seamless and efficient experience for both the customers and the participating businesses, and that the system is easy to use and navigate for all users.

### 2.2.1 Class Diagram



### 2.3 User Characteristics

The registered user does not have to be expert at managing spendings, because the application will simplify the process for him.

| **User Characteristics** | ***Registered User*** |
|---|---|
| *Age* | *15 years or older* |
| *Gender* | *male/female* |
| *Physical Limitations* | *No physical limitations* |
| *Educational level* | *Intermediate School or higher* |
| *Experience* | *Know how to use mobiles* |
| *Technical Experience* | *Novice* |
| *Frequent of Use* | *Almost Daily* |

Business Owner should be more familiar with dashboard system to analyze the displayed data and graphs.

| **User Characteristics** | ***Business Owner*** |
|---|---|
| *Age* | *18 years or older* |
| *Gender* | *male/female* |
| *Physical Limitations* | *No physical limitations* |
| *Educational level* | *High school or higher* |
| *Experience* | *Know how to use PC and Dashboards* |
| *Technical Experience* | *Expert* |
| *Frequent of Use* | *Almost Daily* |

### 2.4 Constraints

1. The system shall be developed and delivered within 9 months from the start of the project.
2. The design should support different layouts for English and Arabic languages, with the ability to switch between them within the app.
3. The design should be responsive and adaptive to different screen sizes and orientations, with proper font scaling and image rendering.
4. The minimum age requirement for users is 15 years old, and the system should have a validation mechanism to enforce this.
5. The user shall be able to access their wallet within 2 steps, without excessive clicks or pop-ups.
6. The business owner shall be able to modify an offer within 24 hours from its creation.
7. The business owner shall be able to delete an offer within 24 hours from its creation.

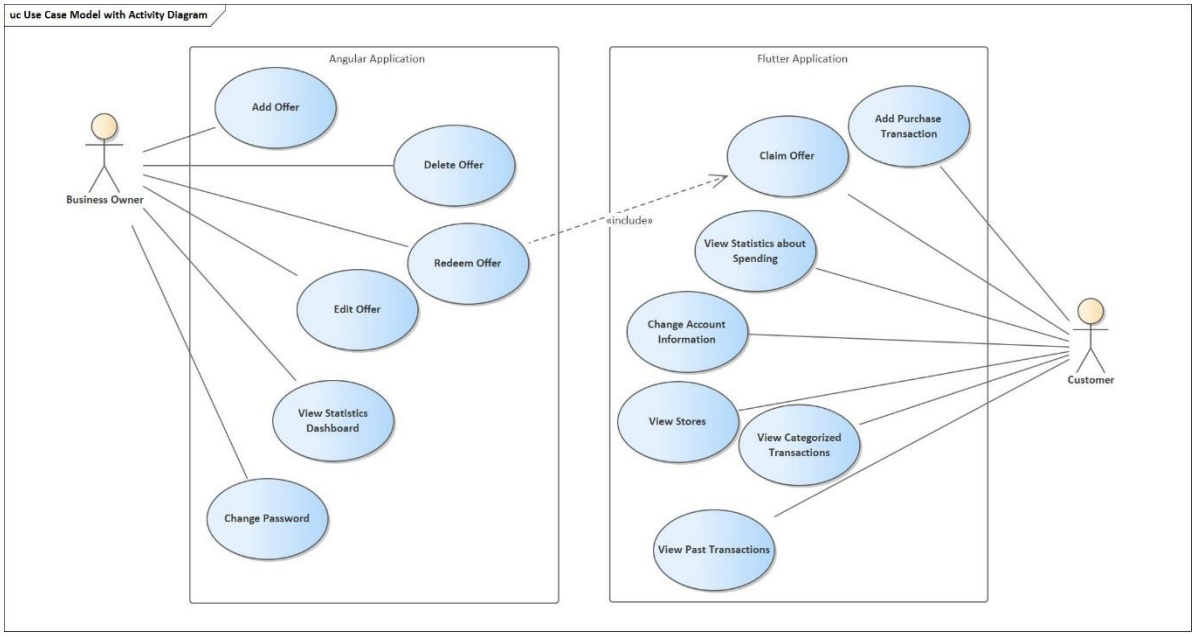
8. The client will have the option to report an infringement within 3 steps, with a clear reporting process and feedback mechanism.
9. The system shall use secure encryption and hashing algorithms to protect user data and login credentials.
10. The system requires a minimum of 2 GB of primary memory and 4 GB of secondary memory for installation and execution.
11. The system shall have a fast loading time, not exceeding 3 seconds for the initial screen and 1 second for subsequent screens.
12. The system shall use secure notification channels and protocols to send relevant notifications to users, without compromising their privacy or security.
13. The system shall use advanced analytics and data visualization tools to provide insights and recommendations to business owners and users, based on their behavior, preferences, and interests.
14. The system shall have a high level of compatibility and integration with popular third-party services and APIs, such as Google Maps, Firebase, and AWS.
15. The login verification process should take less than 5 seconds to process, with the ability to detect and prevent fraudulent login attempts or suspicious activities.
16. The system shall have a robust and scalable architecture, with high availability and performance, even under heavy loads or peak traffic.
17. The system shall have a comprehensive and user-friendly documentation and support system, with clear guidelines, tutorials, and FAQs.
18. The system shall have a flexible and customizable backend and dashboard, with the ability to customize and configure different settings, features, and permissions for different users and roles.
19. The system shall comply with all relevant laws, regulations, and standards, such as GDPR, PCI DSS, and ISO 27001, and have regular security audits and assessments to ensure compliance and mitigate risks.

## 3. Specific Requirements

### 3.1 Detailed Requirements

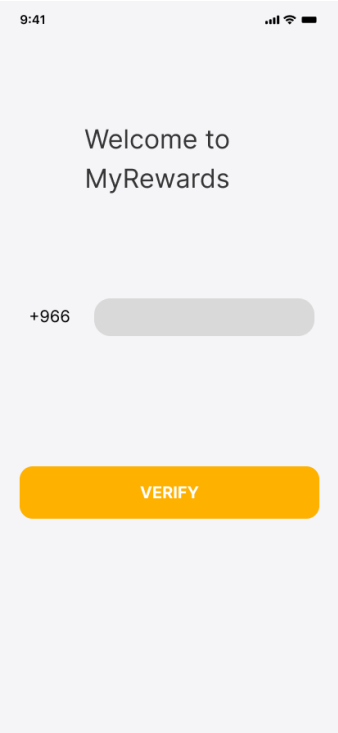
1. The system shall allow the user to view wallet balance
The system shall display a dashboard with users' rankings (XP points).
The system shall enable the BO to add a new offer
The BO should be able to register its store through the website
The system shall generate promo code for small/medium BOs
The system shall send users the generated promo code
2. The system shall enable BOs to validate promo code through the system.
The system shall collect and analyze incoming texts to user mobile and check if they are from a bank and it is a purchase transaction, extract the store name, amount, and date of transaction.
The system shall store data collected from the user’s SMS message
Users shall be able to view their monthly spending
The user shall be able to register by their phone number
The system shall prompt (optional) the user for birth date, sex.
The system shall give each account an associated points wallet
The system shall enable the user to change their phone number.
The BO shall register by their email and password.
The BO shall be able to change their password
The system shall authenticate the SMS sender.
The system shall validate phone number by OTP

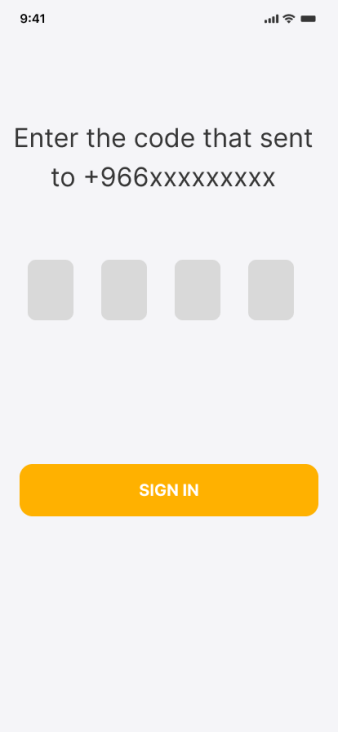
## 3.2 Use case Diagram

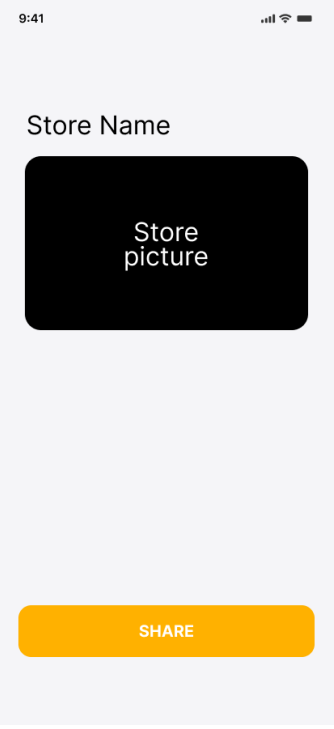


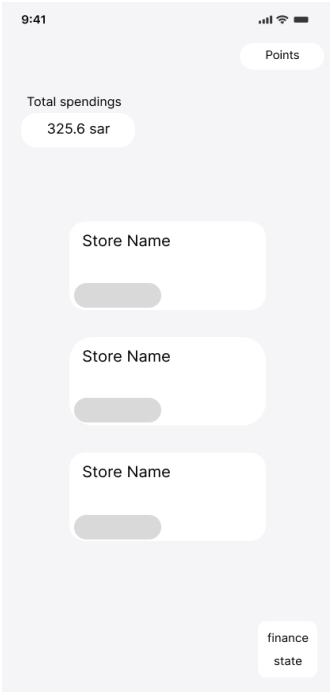

## 3.3 Interface Requirements

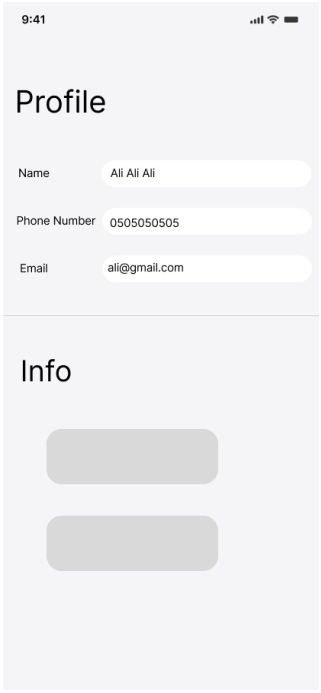
### 3.3.1 User Interfaces

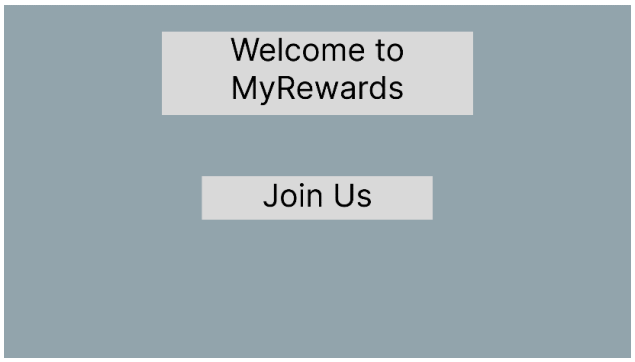


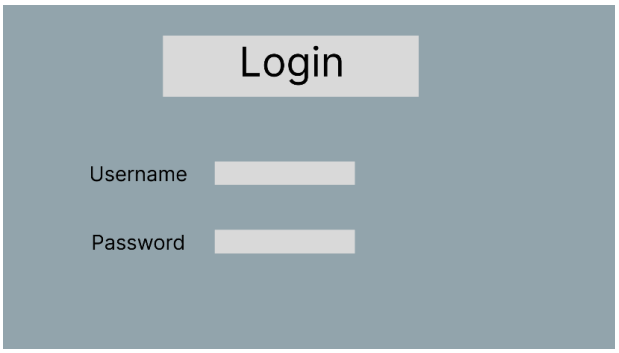



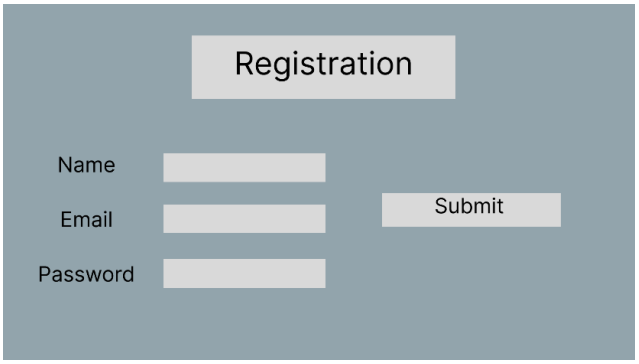



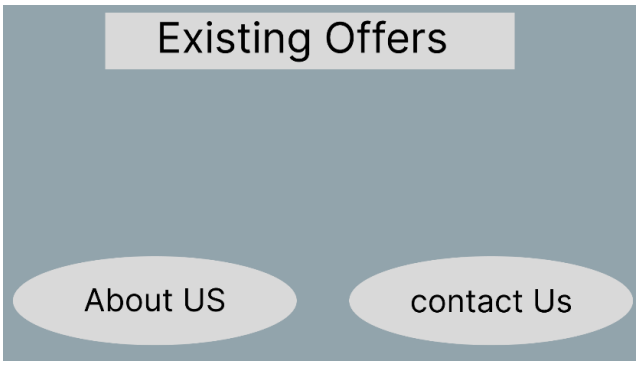

9:41
Points
Total spendings
325.6 sar
Store Name
Store Name
Store Name
finance
state

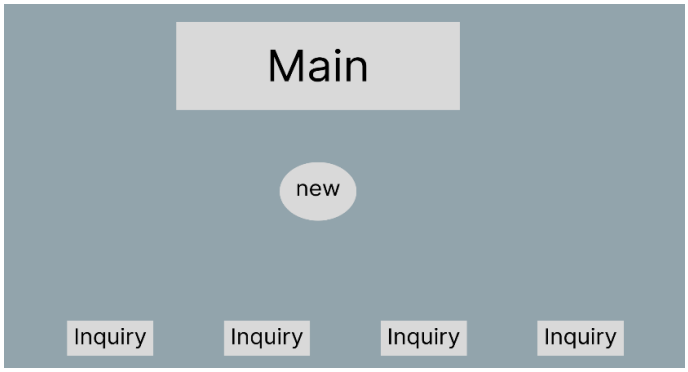

9:41
Profile
Name
Ali Ali Ali
Phone Number
0505050505
Email
ali@gmail.com
Info


Welcome to MyRewards
Join Us


Login
Username
Password


Registration
Name
Email
Password
Submit


Existing Offers
About US
contact Us


Main
new
Inquiry
Inquiry
Inquiry
Inquiry

### 3.3.2 Hardware Interfaces

The system will be deployed as a mobile application. So, there will not be any special hardware equipment needed for the application to run. Average smart phones with 2GB of ram is more than sufficient for this application to work.

As for the Business Owners they might need a QR Code Reader to read the generated QR code that will have the offers for the customers. However, it is not mandatory to for the business owner to have QR code reader, as this task can be done via a smart phone's camera.

### 3.3.3 Software Interfaces

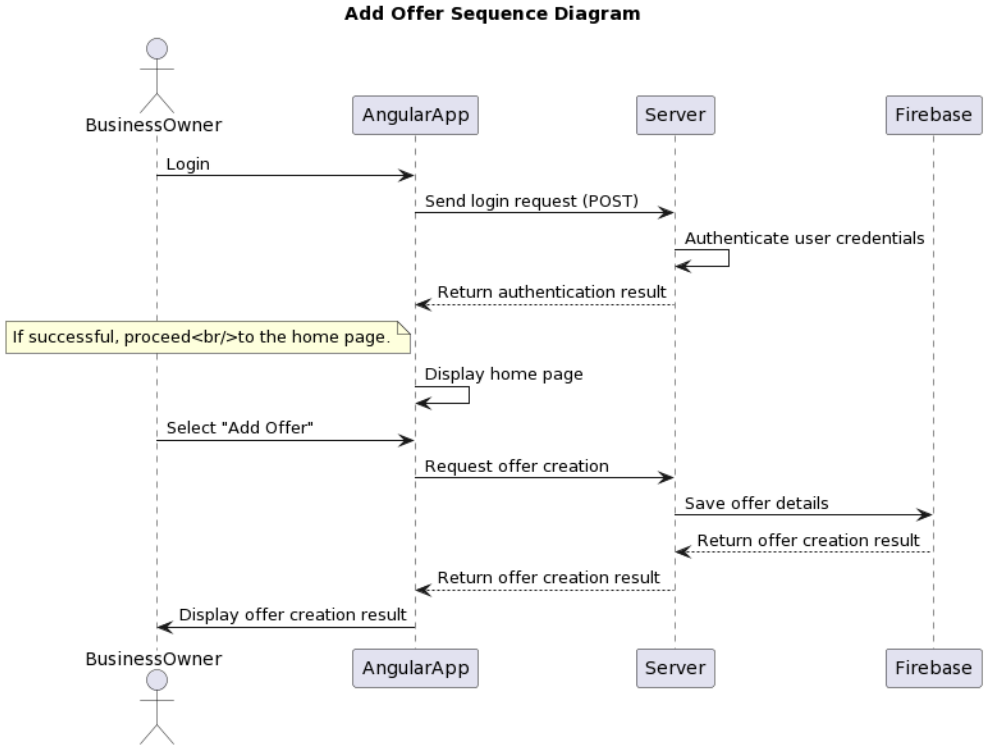
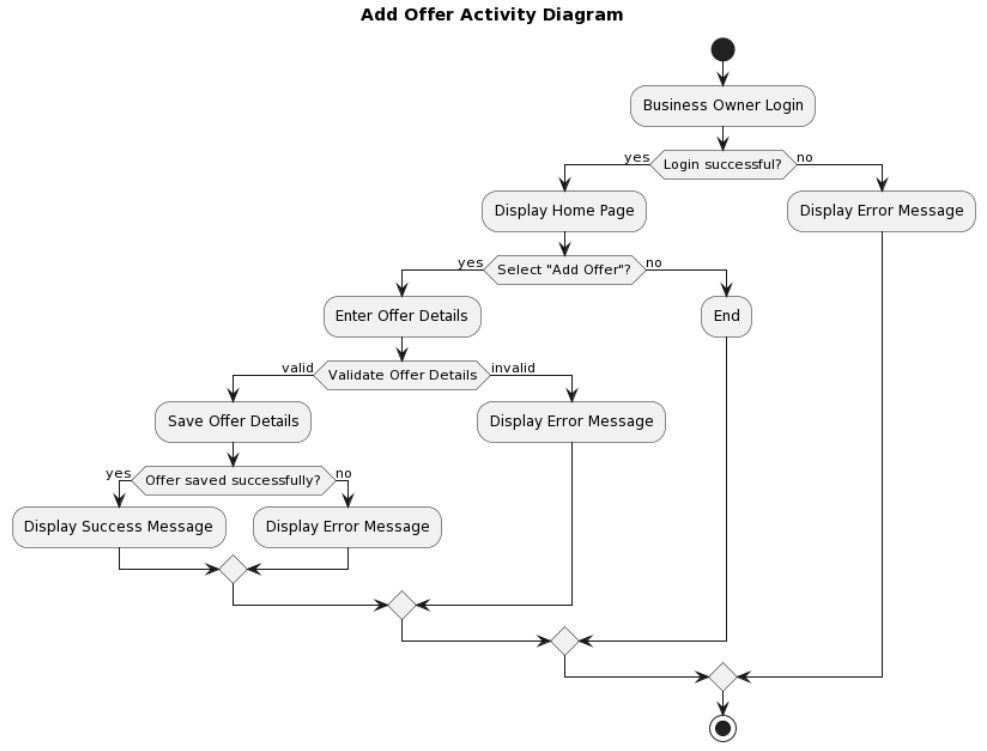
| Name | *MongoDB Atlas* |
|---|---|
| Mnemonic | *MongoDB* |
| Version number | *Version 6.0* |
| Source | *https://www.mongodb.com/* |
| Purpose of Interfacing | *The system must use a NoSQL database and the fastest and most efficient one and uses JSON-like documents.* |

| Name | *VueJS* |
|---|---|
| Mnemonic | *Vue* |
| Version number | *Version 3.2.41* |
| Source | *https://vuejs.org/* |
| Purpose of Interfacing | *The system must use a web framework to make the web portal for the Business Owners and also to make an easy communication and connection between the web portal and the database.* |

## 3.4 Functional Requirements Description

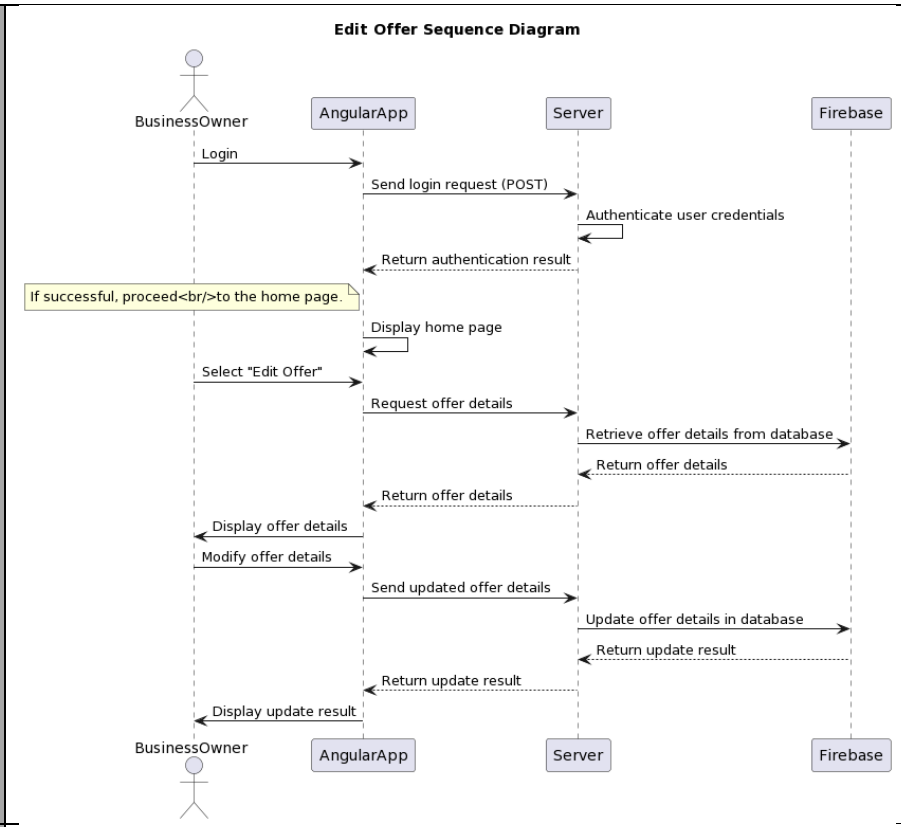
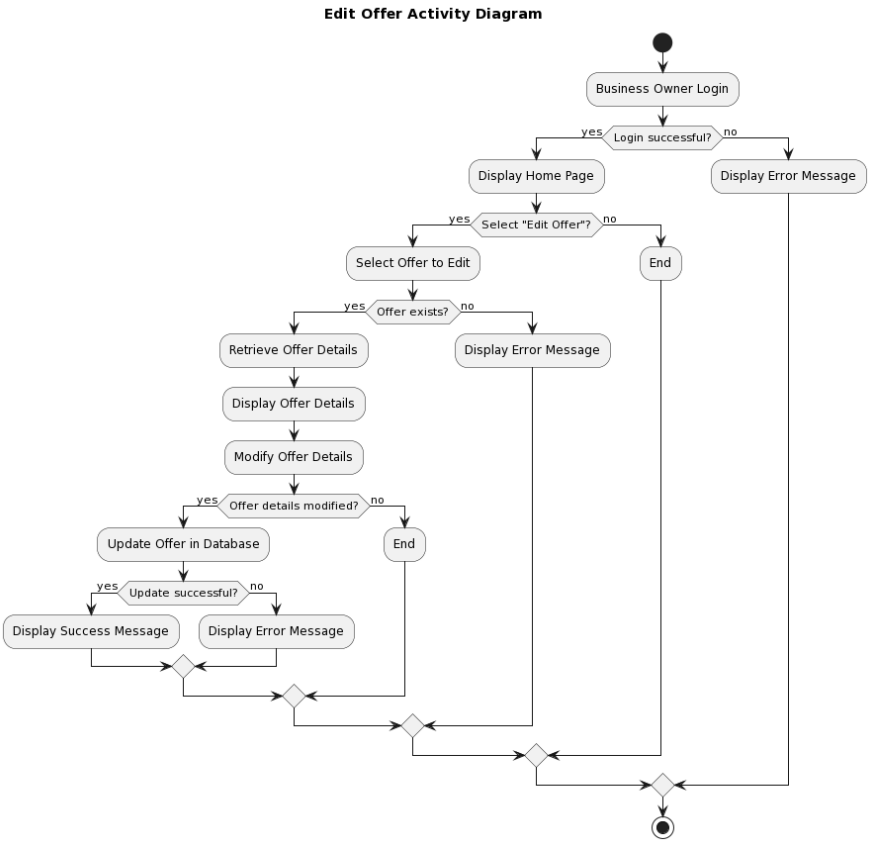
### 3.4.1 Add Offer

| UC-01: Add Offer | |
|---|---|
| Description: | This use case allows the Business Owner (BO) to create a new offer to be displayed on the mobile application. |
| Actors: | Business Owner |
| Pre-Condition(s): | • The BO must be logged into the mobile application.<br>• The BO must have the necessary permissions to create a new offer.<br>• The BO must have an active internet connection. |
| Post-Condition(s): | The new offer is saved to the database and displayed on the mobile application for users to view and redeem. |
| Main Flow: | 1- The BO navigates to the "Create Offer" section of the mobile application.<br>2- The BO enters the offer details, including the offer title, description, expiration date, and any other relevant information.<br>3- The BO selects the image or video to be associated with the offer.<br>4- The BO sets the number of points required for users to redeem the offer.<br>5- The BO reviews the offer details and confirms that they are correct.<br>6- The BO clicks the "Create" button to save the new offer to the database.<br>7- The mobile application displays a confirmation message that the offer has been successfully created. |
| Alternative(s): | 2a. Business Owner selects an existing offer to duplicate:<br>• System displays a list of existing offers.<br>• Business Owner selects the offer to duplicate.<br>• System pre-fills the offer details form with the selected offer's information.<br>3a. Business Owner selects to create a recurring offer:<br>• Business Owner selects "Recurring" option in the Create Offer form.<br>• Business Owner selects the frequency (daily, weekly, monthly) and the number of times the offer will be available.<br>• System generates a unique code for each instance of the recurring offer. |
| Exception(s): | 6a. Business Owner submits the form with missing or invalid information:<br>• System displays an error message next to the invalid field(s). |

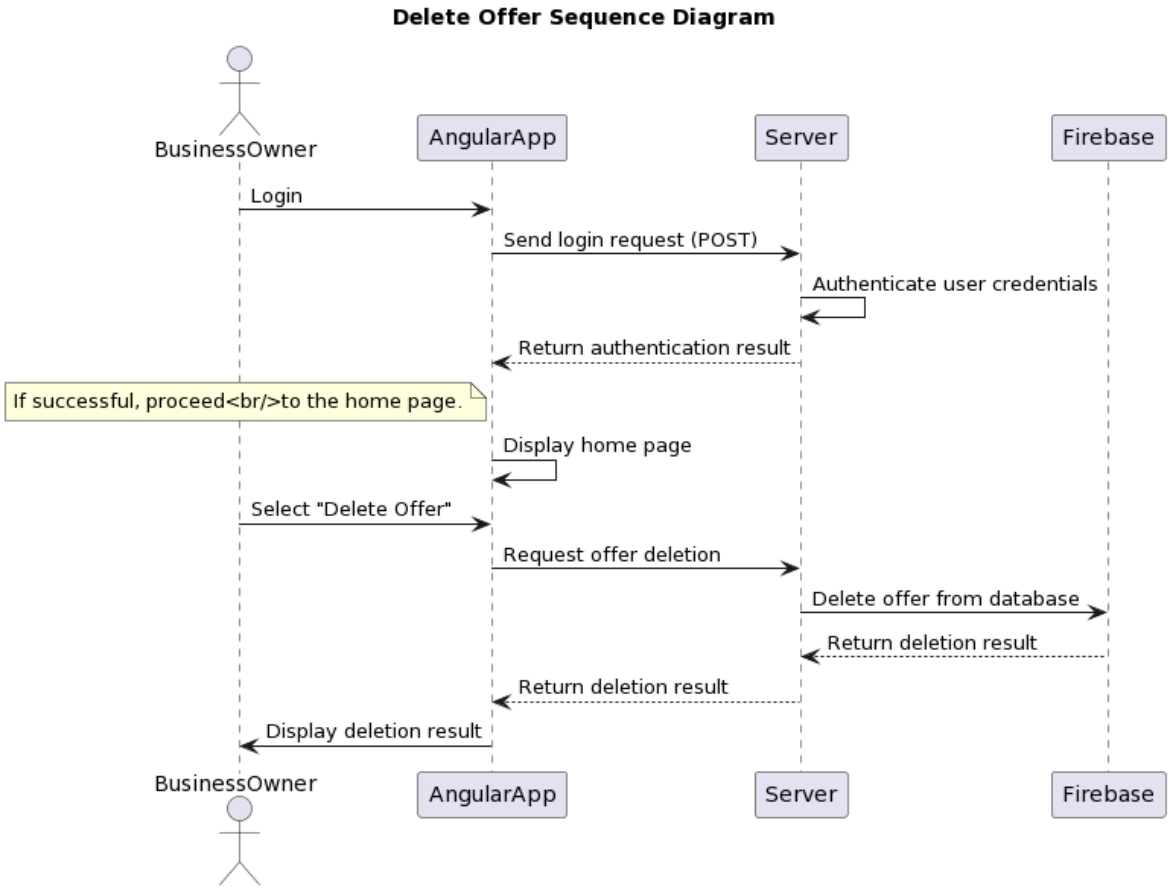
| | |
|---|---|
| | • Business Owner corrects the information and resubmits the form.<br>6b. Business Owner submits the form with an existing offer title:<br>• System displays an error message that the offer title already exists in the system.<br>• Business Owner selects a new title and resubmits the form.<br>7a. System fails to save the new offer:<br>• System displays an error message that the offer could not be saved.<br>• Business Owner can retry or contact technical support. |
| Dependency: | |
| Sequence Diagram |  |
| Activity Diagram |  |

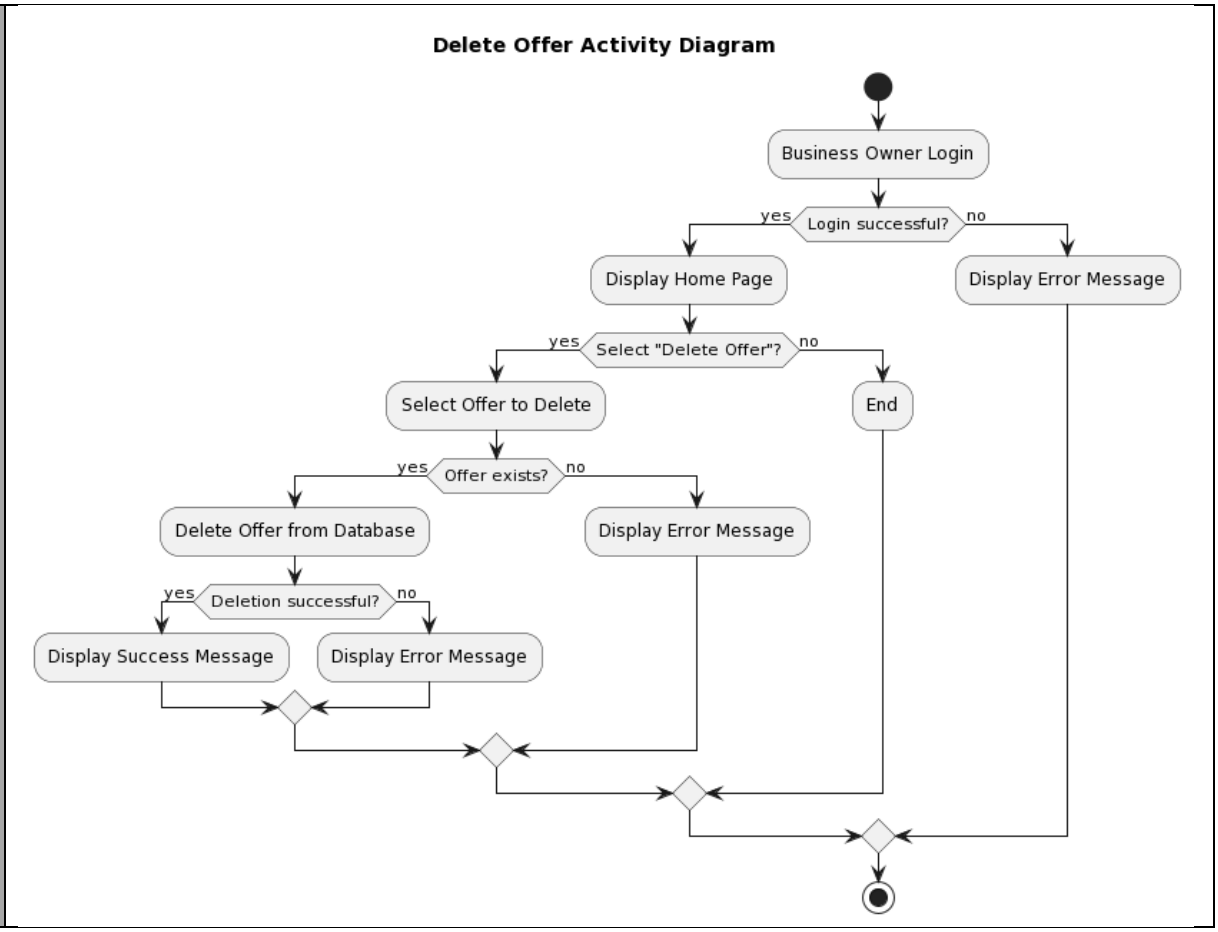
### 3.4.2 Edit Offer

| UC-02: Edit Offer | |
|---|---|
| Description: | The business owner updates an existing offer in the system. |
| Actors: | Business Owner |
| Pre-Condition(s): | • The business owner must be authenticated and authorized to edit offers.<br>• The business owner must have at least one existing offer in the system. |
| Post-Condition(s): | • The offer is updated in the system.<br>• The business owner receives a confirmation message. |
| Main Flow: | 1. The business owner logs into the system.<br>2. The system displays the business owner dashboard.<br>3. The business owner selects the offer they wish to edit.<br>4. The system displays the offer details.<br>5. The business owner makes the necessary changes to the offer.<br>6. The business owner confirms the changes.<br>7. The system updates the offer in the system.<br>8. The system displays a confirmation message to the business owner. |
| Alternative(s): | 5a. If the business owner decides not to edit the offer:<br>1. The business owner cancels the edit.<br>2. The system returns the business owner to the offer details page. |
| Exception(s): | 6a. If the business owner inputs invalid data:<br>1. The system displays an error message.<br>2. The system prompts the business owner to correct the invalid data.<br>3. The business owner corrects the invalid data.<br>4. The business owner confirms the changes.<br>5. The system updates the offer in the system.<br>6. The system displays a confirmation message to the business owner.<br><br>7a. If the system fails to update the offer:<br>1. The system displays an error message.<br>2. The system prompts the business owner to try again later.<br><br>8a. If the business owner cancels the confirmation message:<br>1. The system returns the business owner to the offer details page. |
| Dependency: | |

| | |
|---|---|
| Sequence Diagram |  |
| Activity Diagram |  |

### 3.4.3 Delete Offer

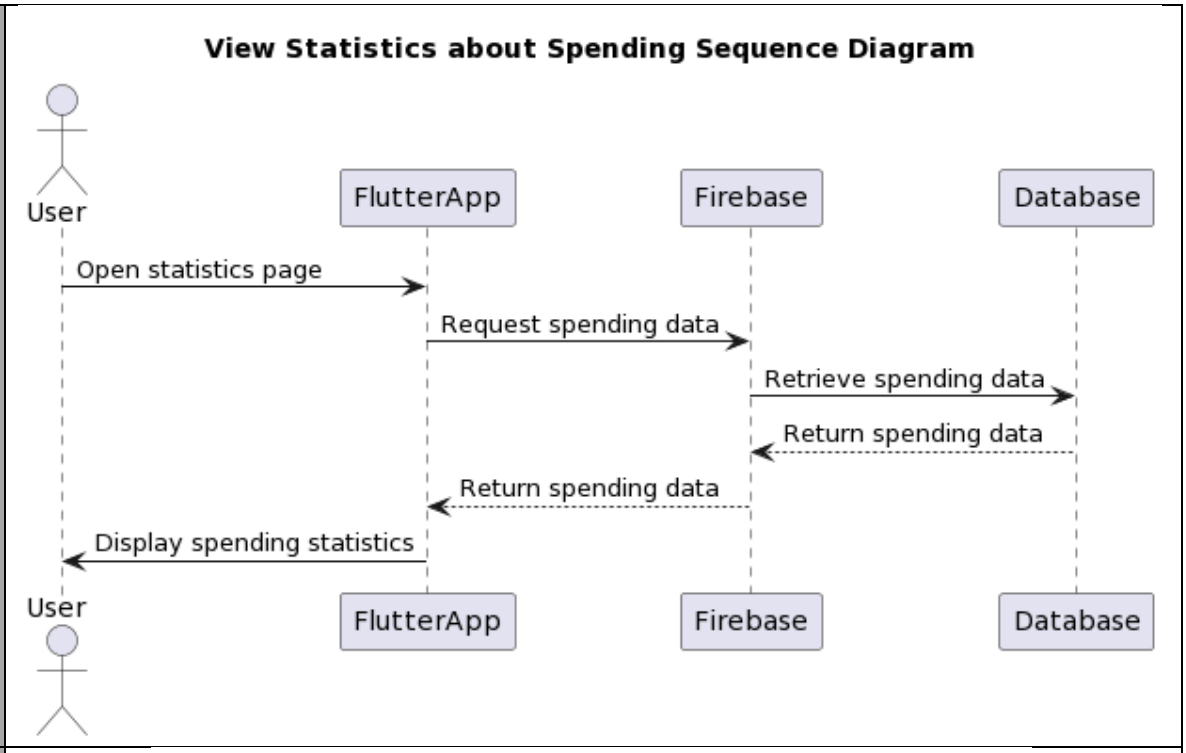
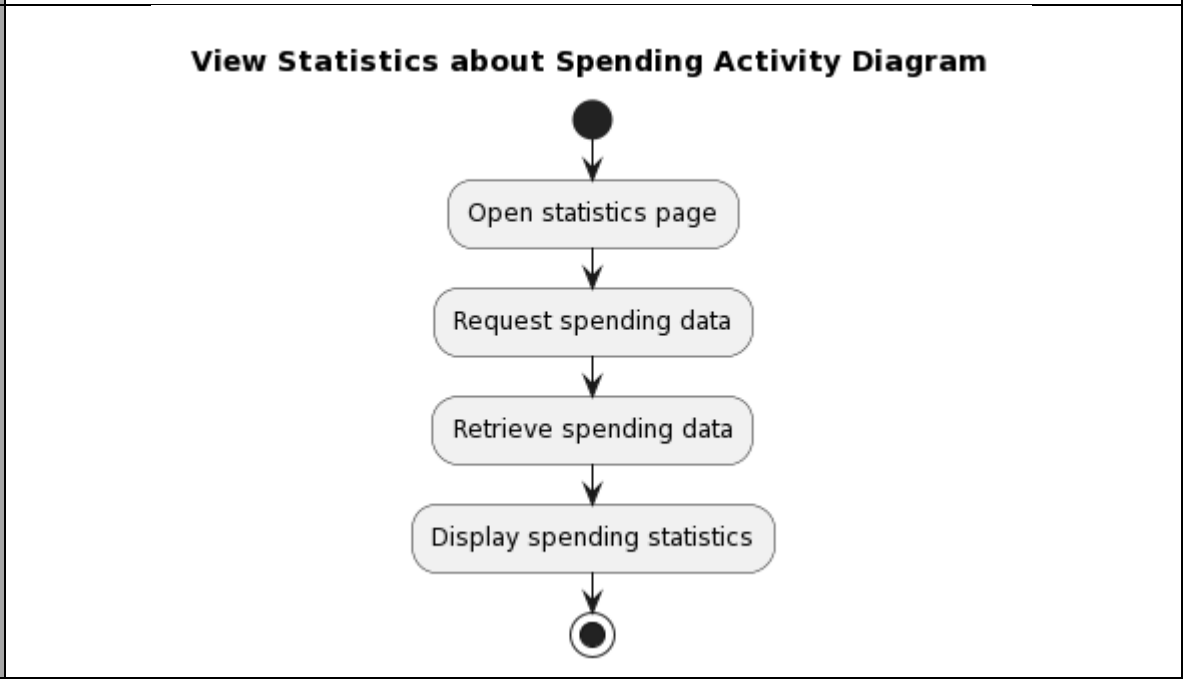
| UC-03: Delete Offer | |
|---|---|
| Description: | This use case allows the business owner to delete an existing offer from the system. |
| Actors: | Business Owner |
| Pre-Condition(s): | • The Business Owner must be authenticated and authorized to delete an offer.<br>• The offer must exist in the system. |
| Post-Condition(s): | • The offer is deleted from the system. |
| Main Flow: | 1. The Business Owner selects the option to delete an offer.<br>2. The system displays a list of all existing offers.<br>3. The Business Owner selects the offer to be deleted.<br>4. The system displays a confirmation message to ensure that the Business Owner wants to delete the selected offer.<br>5. The Business Owner confirms the deletion request.<br>6. The system deletes the offer from the system and displays a success message. |
| Alternative(s): | Step 2a: If there are no existing offers, the system displays a message indicating that there are no offers to delete.<br>Step 4a: If the Business Owner decides not to proceed with the deletion, the system cancels the deletion process and returns to the main menu. |
| Exception(s): | Step 6a: If the system fails to delete the offer from the system, an error message is displayed, and the Business Owner is prompted to try again or contact support. |
| Dependency: | |
| Sequence Diagram |  |

| | |
|---|---|
| Activity Diagram |  |



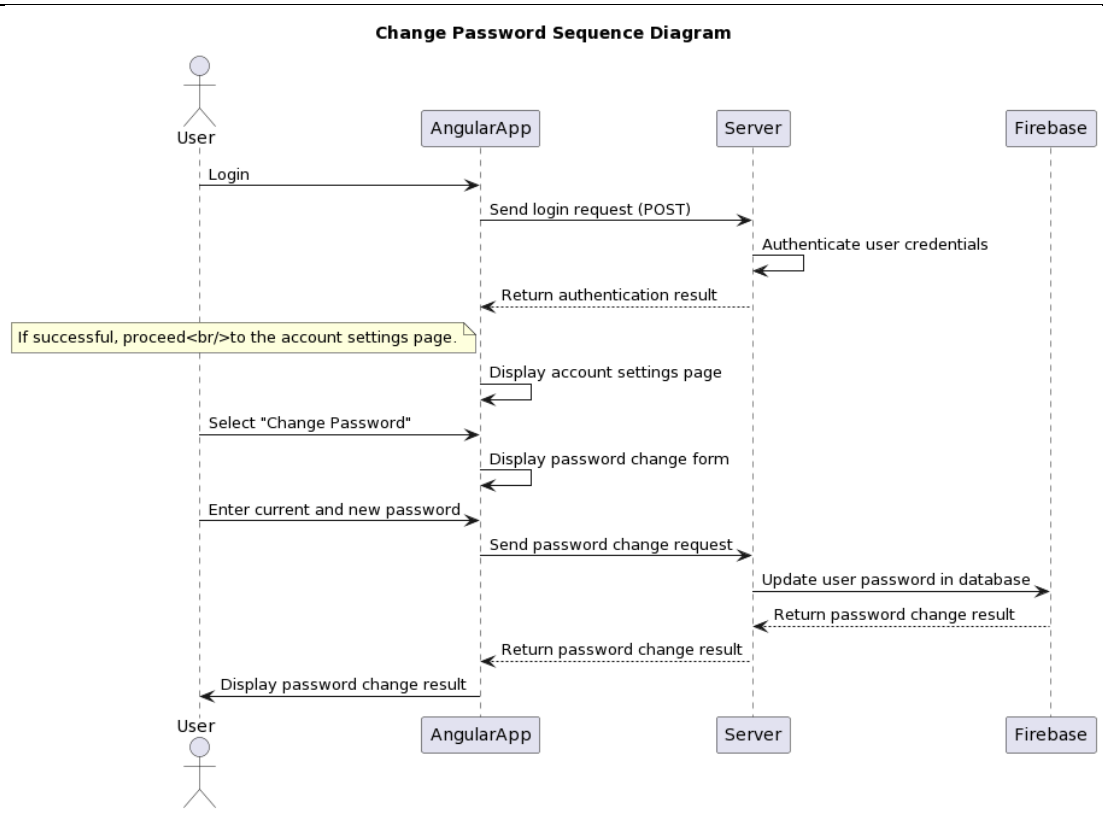
### 3.4.4 View Statistics Dashboard

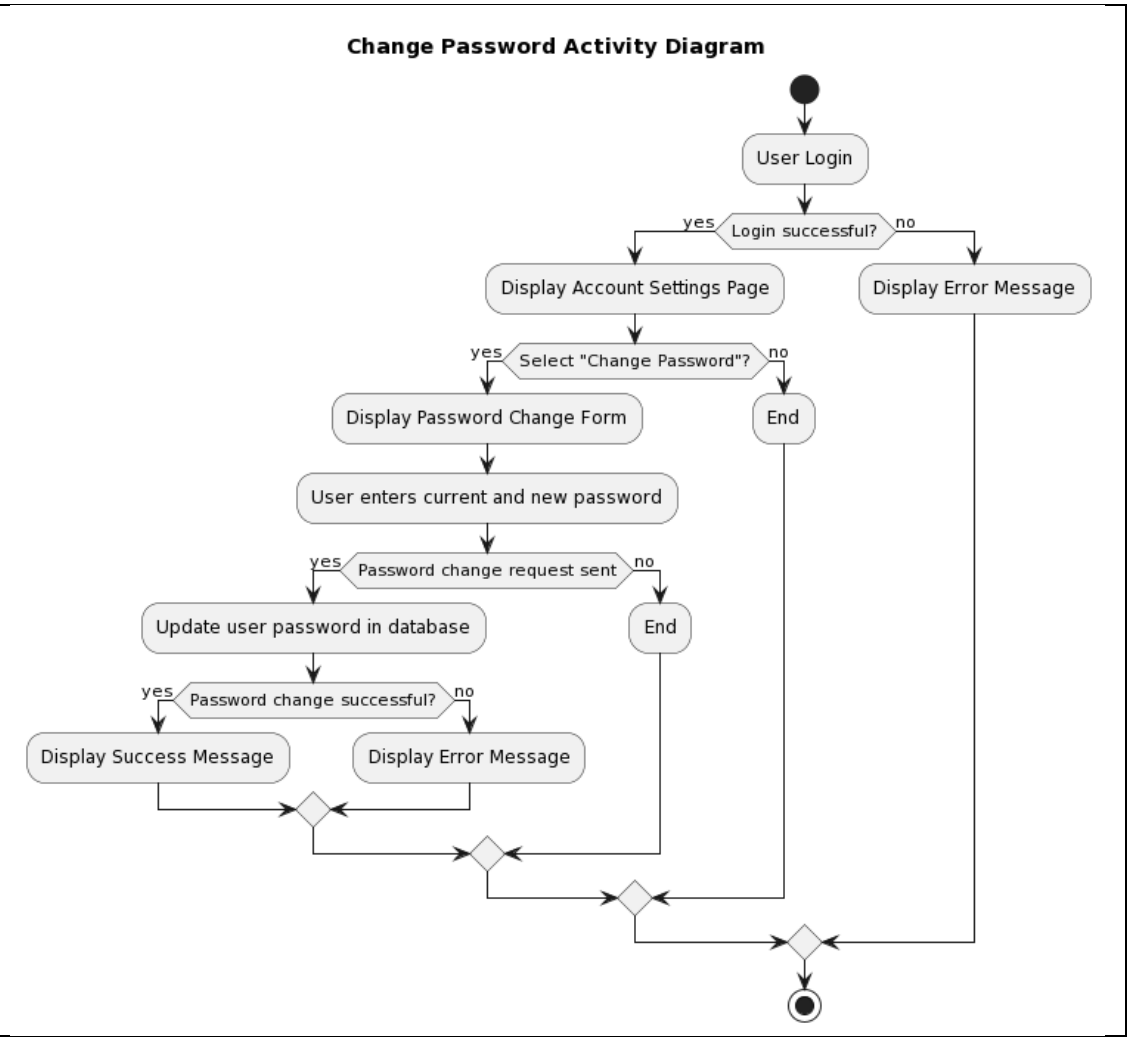
| UC-04: View Statistics Dashboard | |
|---|---|
| Description: | This use case allows the business owner to view statistics related to the offers, customers, and users of the system. The statistics dashboard includes graphical representations of the data in the form of charts, tables, and other data visualizations. |
| Actors: | Business Owner |
| Pre-Condition(s): | • The business owner must be logged into the system.<br>• The business owner must have appropriate permissions to view statistics. |
| Post-Condition(s): | • The business owner has viewed the statistics dashboard. |
| Main Flow: | 1. The business owner navigates to the statistics dashboard page.<br>2. The system presents the statistics dashboard, which includes various data visualizations, such as charts and tables.<br>3. The business owner can interact with the data visualizations to view more detailed information or filter the data.<br>4. The business owner can export the data to a CSV file. |
| Alternative(s): | |
| Exception(s): | 1a. If the business owner is not logged into the system, the system redirects them to the login page.<br>1b. If the business owner does not have appropriate permissions to view statistics, the system displays an error message and does not show the statistics dashboard. |
| Dependency: | |

| | |
|---|---|
| Sequence Diagram |  |
| Activity Diagram |  |

### 3.4.5 Change Password

| UC-05: Change Password | |
|---|---|
| Description: | The Change Password use case describes the steps that a registered user must follow in order to change their account password. |
| Actors: | Business Owner |
| Pre-Condition(s): | • The user must be registered and logged into the application.<br>• The user must have a current password. |
| Post-Condition(s): | • The user's password is updated and saved in the database. |
| Main Flow: | 1. The user navigates to the account settings page.<br>2. The user selects the "Change Password" option.<br>3. The system prompts the user to enter their current password and new password. |

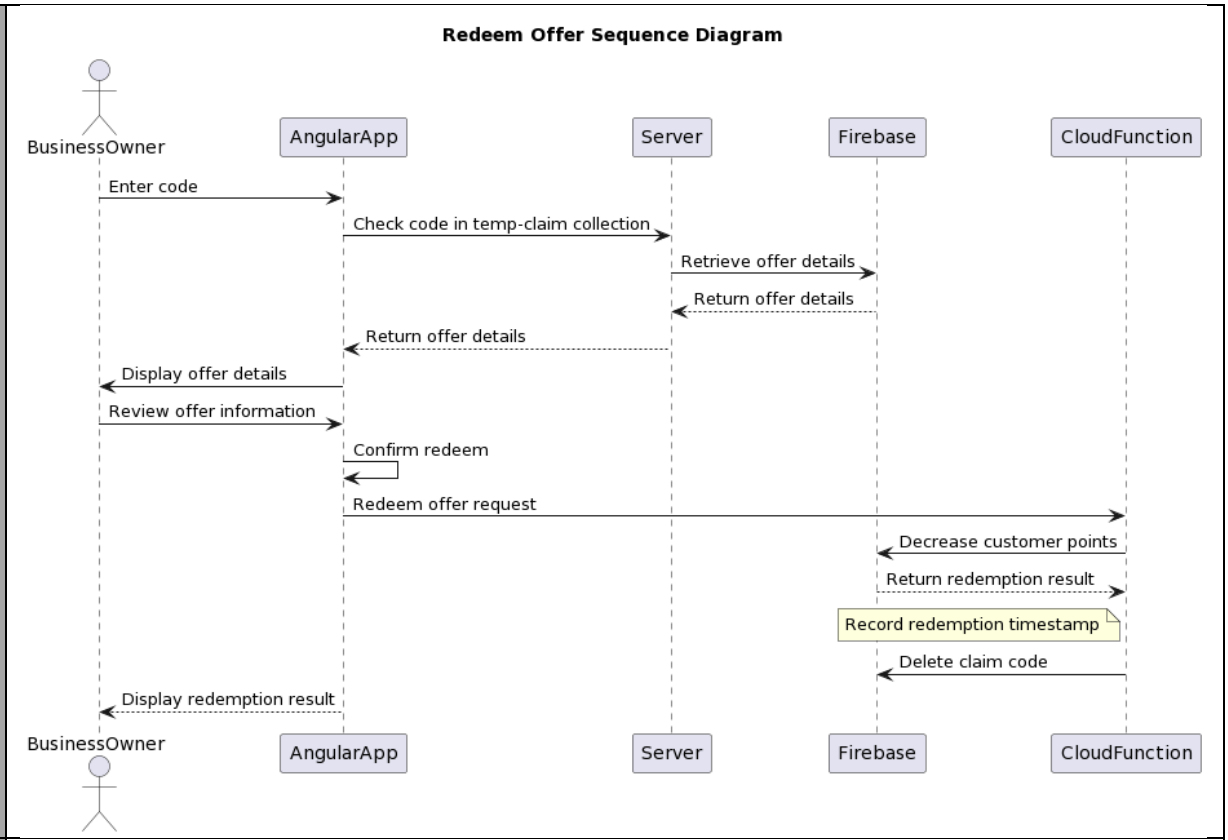
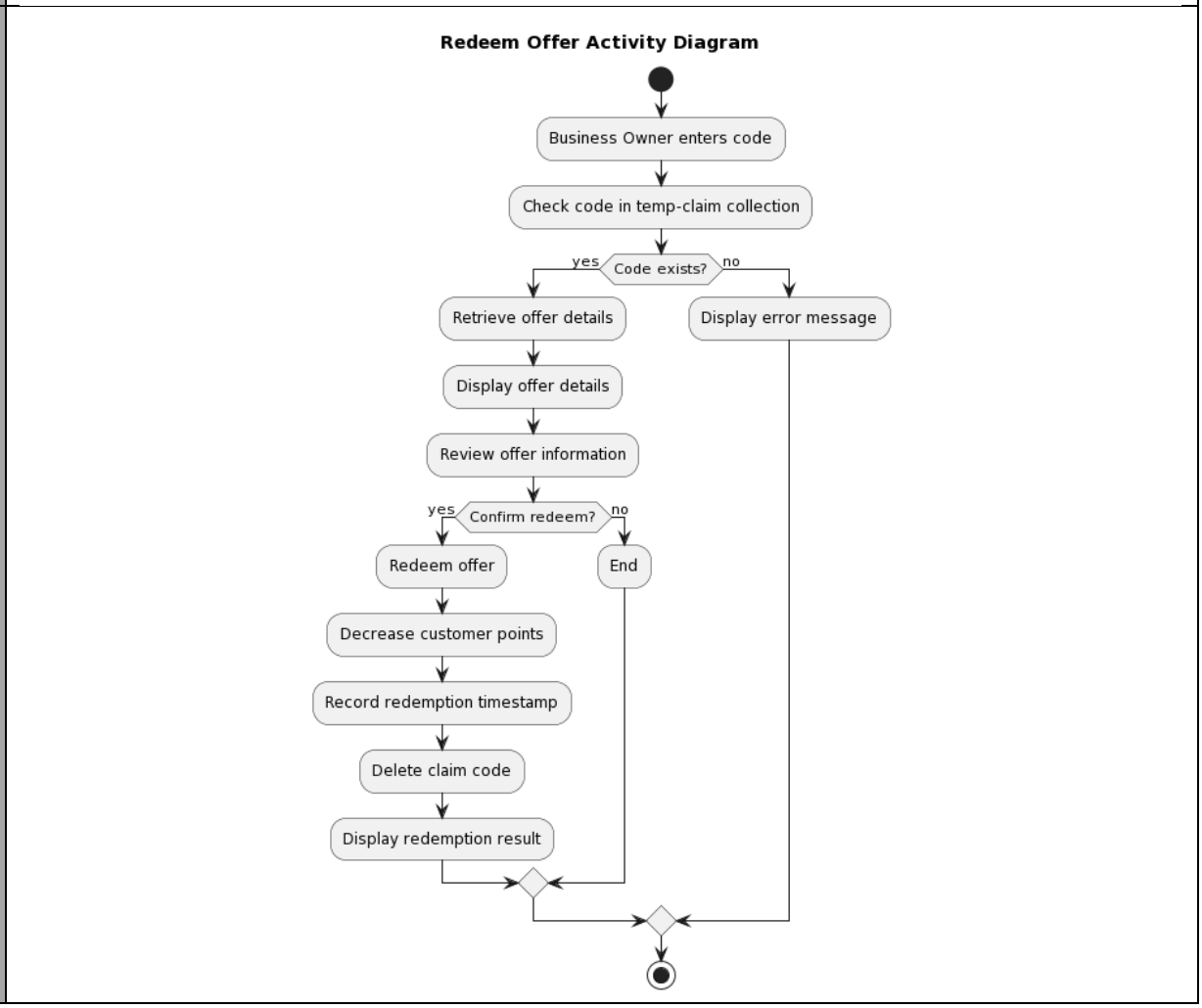
| | |
|---|---|
| | 4. The user enters their current password and new password.<br>5. The system verifies that the entered current password matches the user's current password in the database.<br>6. The system verifies that the new password meets the password complexity requirements.<br>7. The system saves the new password in the database.<br>8. The system displays a confirmation message that the password has been changed. |
| Alternative(s): | Step 5: If the entered current password does not match the user's current password in the database, the system displays an error message and prompts the user to re-enter their current password.<br>Step 6: If the new password does not meet the password complexity requirements, the system displays an error message and prompts the user to enter a new password that meets the requirements. |
| Exception(s): | If the system is unable to save the new password in the database due to technical errors, the system displays an error message and prompts the user to try again later. |
| Dependency: | |
| Sequence Diagram |  |

| | |
|---|---|
| Activity Diagram |  |



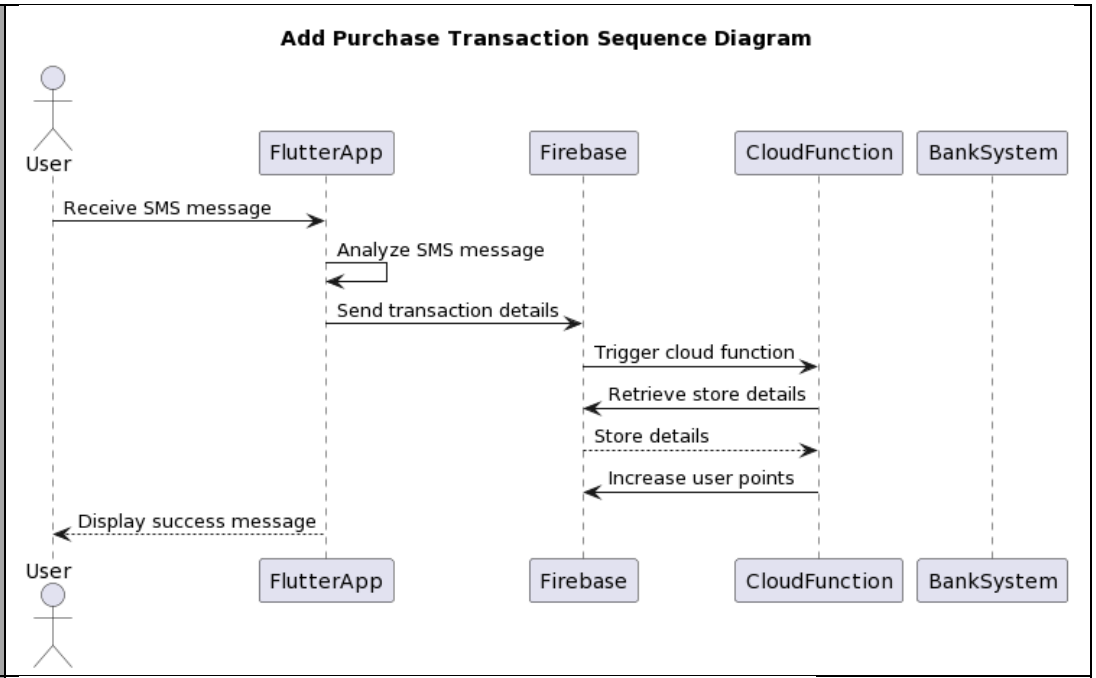
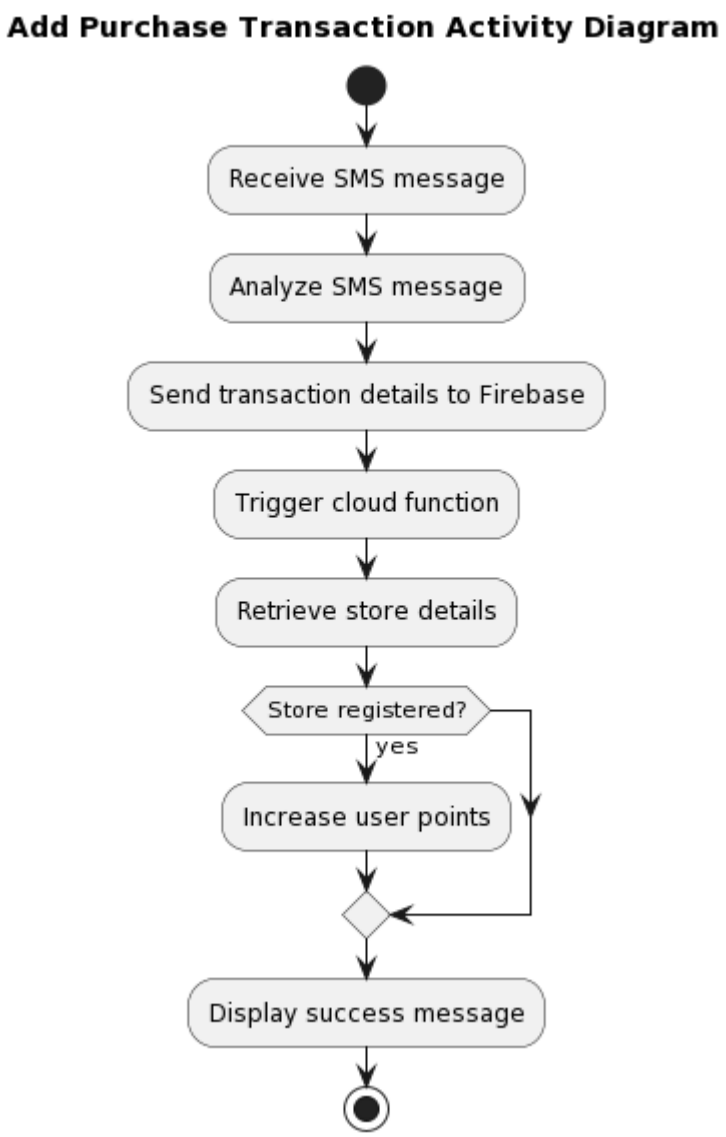
### 3.4.6 Redeem Offer

| UC-06: Redeem Offer | |
|---|---|
| Description: | This use case allows the business owner to redeem an offer for a customer by entering a code. |
| Actors: | Business Owner |
| Pre-Condition(s): | • The business owner must be authenticated and have the necessary permission to redeem offers.<br>• A valid code should be provided by the customer. |
| Post-Condition(s): | • The customer's points should be decreased by the redeemed points value.<br>• The claim code should be deleted. |
| Main Flow: | 1. The business owner enters the claim code provided by the customer.<br>2. The system checks the temp-claim collection to see if the code exists.<br>3. If the code exists, the system displays the offer details to the business owner.<br>4. The business owner reviews the offer details and decides to redeem it.<br>5. The business owner clicks on the "Redeem" button.<br>6. The system calls a Firebase cloud function to decrease the customer points and records the timestamp of this redeem.<br>7. The system deletes the claim code.<br>8. The system displays a confirmation message to the business owner. |
| Alternative(s): | 3a. If the code does not exist, the system displays an error message and prompts the business owner to re-enter the code. |
| Exception(s): | 6a. If there is an error in calling the Firebase cloud function, the system displays an error message and prompts the business owner to try again. |
| Dependency: | UC-08 |

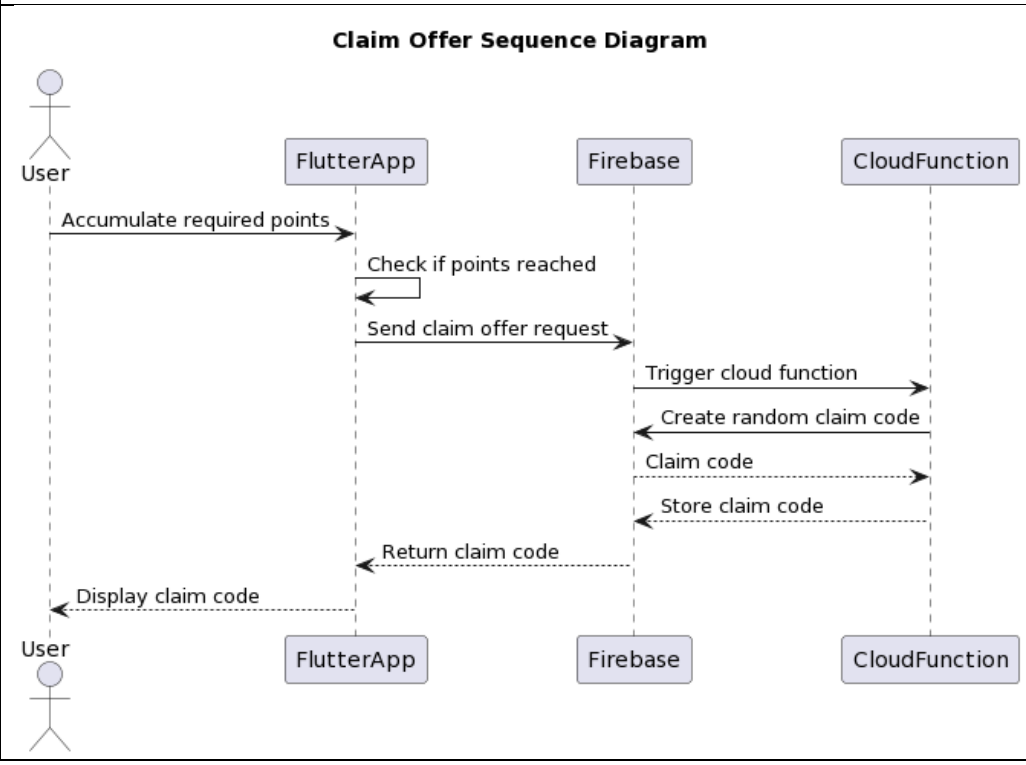
| | |
|---|---|
| Sequence Diagram |  |
| Activity Diagram |  |

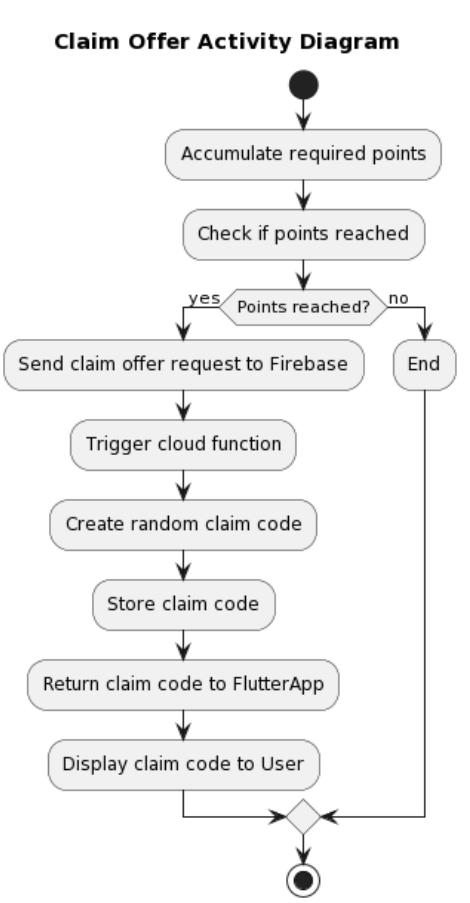
### 3.4.7 Add Purchase Transactions

| UC-07: Add Purchase Transactions | |
|---|---|
| Description: | The application analyzes the message to check if it contains a purchase transaction details from a bank. If the message contains a purchase transaction, the application extracts the date, amount, and store name. |
| Actors: | Consumer |
| Pre-Condition(s): | • The user must have installed the application on their mobile device and enabled the SMS listener service.<br>• The user must have received an SMS message from the bank containing a purchase transaction detail.<br>• The Firebase database and cloud functions must be properly set up. |
| Post-Condition(s): | • The purchase transaction details are stored in the system<br>• The user points are increased if the store name is registered in the system. |
| Main Flow: | 1. The application listens to incoming SMS messages.<br>2. The application analyzes the message to check if it contains a purchase transaction detail from the bank.<br>3. If the message contains a purchase transaction, the application extracts the date, amount, and store name and sends the data to the Firebase database.<br>4. A cloud function is triggered to link the transaction and find whether the store name is registered with our system or not.<br>5. If the store name is registered, the cloud function increases the points of the user with the associated store. |
| Alternative(s): | 3a. If the message does not contain a purchase transaction detail, the application discards the message and returns to listening mode. |
| Exception(s): | 4a. If there is an error connecting to the Firebase database, the application displays an error message to the user and logs the error.<br>4b. If there is an error triggering the cloud function, the application displays an error message to the user and logs the error.<br>4c. If the store name is not registered, the cloud function does not increase the user points. |
| Dependency: | |

| | |
|---|---|
| Sequence Diagram |  |
| Activity Diagram |  |

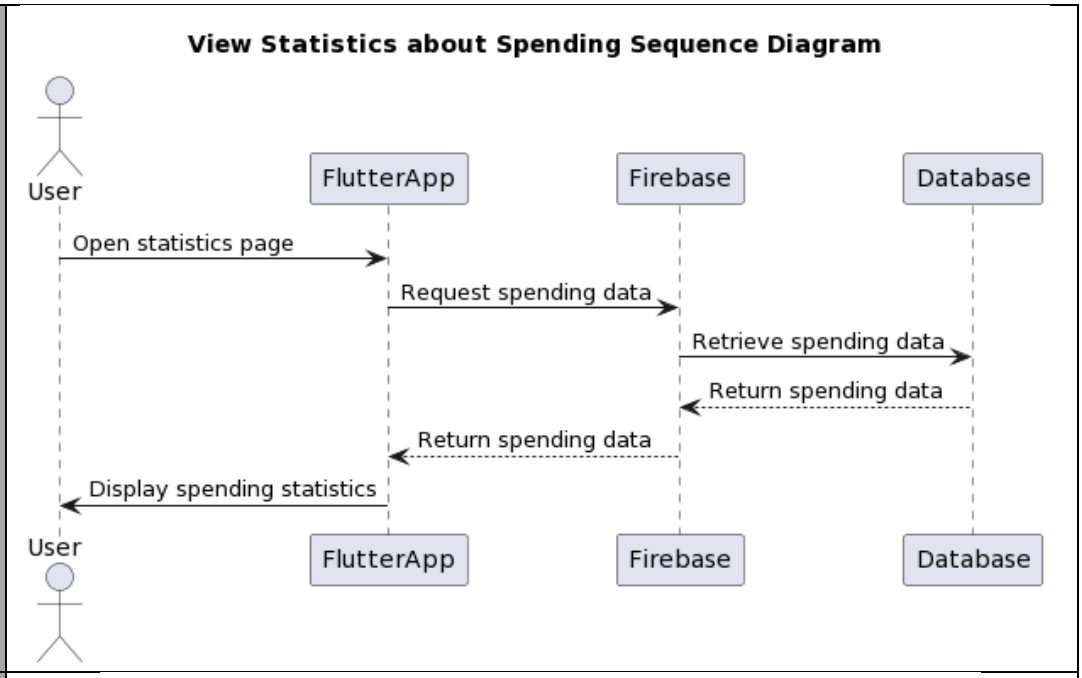
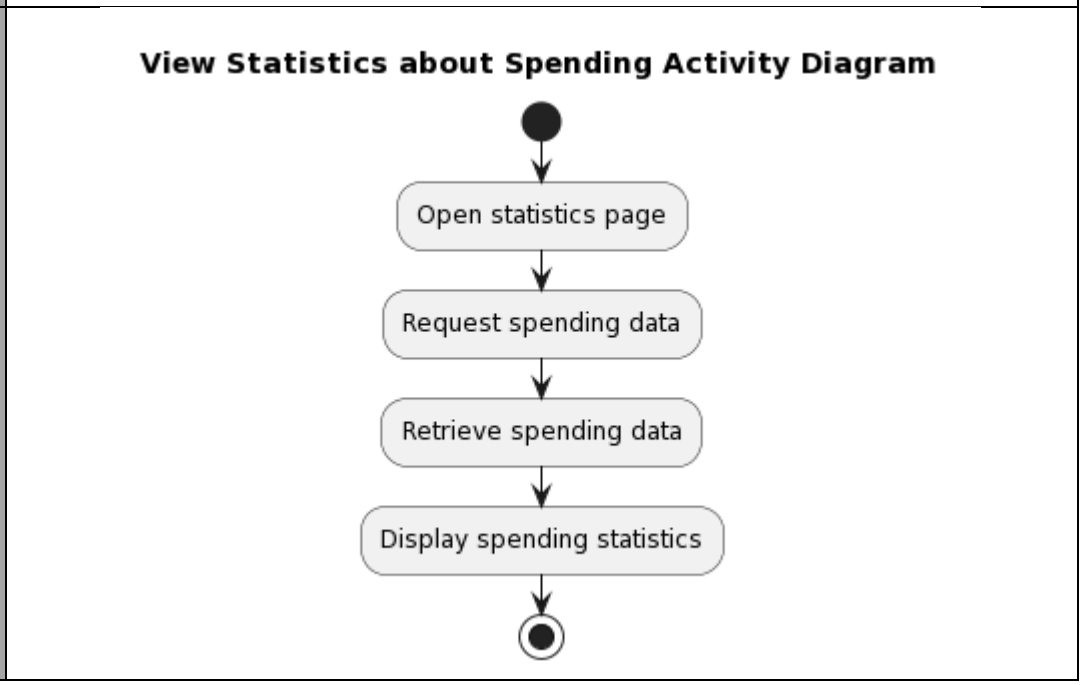
### 3.4.8 Claim Offer

| UC-08: Claim Offer | |
|---|---|
| Description: | his use case involves the process of a user claiming an offer that has been published by a business owner. |
| Actors: | Consumer |
| Pre-Condition(s): | • The user must be logged in to the system.<br>• The user must have a valid claim code provided by the business owner.<br>• The user must have sufficient points to claim the offer. |
| Post-Condition(s): | • The user's points balance is updated to reflect the deduction of points.<br>• The user can view the claimed offer in their wallet. |
| Main Flow: | 1. The user opens the "Claim Offer" screen in the app.<br>2. The user enters the claim code provided by the business owner.<br>3. The system validates the claim code against the temporary claims collection.<br>4. If the claim code is valid, the system presents the offer details to the user.<br>5. The user can choose to claim the offer.<br>6. The system checks the user's points balance.<br>7. If the user has sufficient points, the system deducts the required number of points.<br>8. The system displays a confirmation message to the user.<br>9. The user can view the claimed offer in their wallet. |
| Alternative(s): | If the claim code is invalid, the system displays an error message and prompts the user to enter a valid claim code (Step 3).<br>If the user does not have sufficient points, the system displays an error message and prompts the user to earn more points (Step 7). |
| Exception(s): | |
| Dependency: | |
| Sequence Diagram |  |

| | |
|---|---|
| Activity Diagram |  |

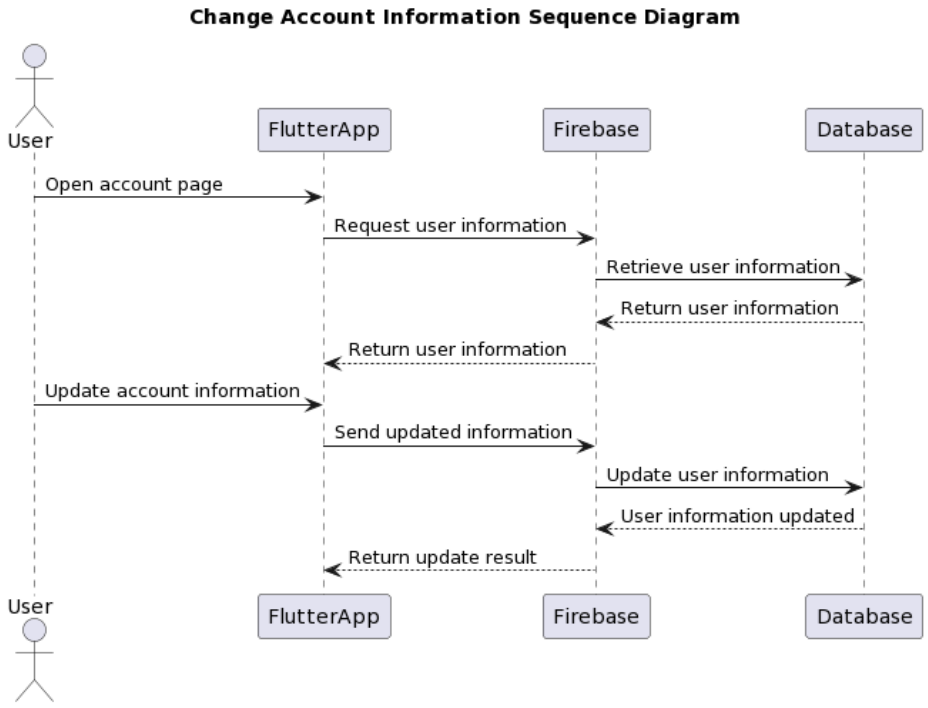
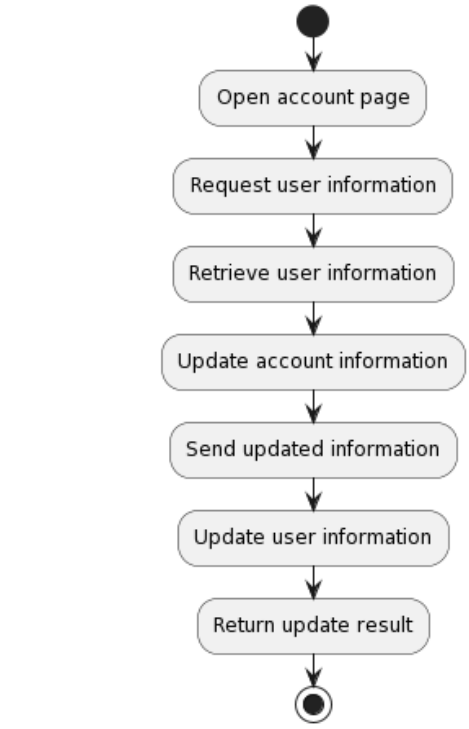
### 3.4.9 View Statistics about Spending

| UC-09: View Statistics about Spending | |
|---|---|
| Description: | This use case allows the user to view statistics about their spending on stores. |
| Actors: | Consumer |
| Pre-Condition(s): | • The user must be logged in to their account.<br>• The user must have redeemed at least one offer. |
| Post-Condition(s): | • The user can view their spending statistics. |
| Main Flow: | 1. The user navigates to the "View Statistics" page.<br>2. The system retrieves the user's spending statistics from the database.<br>3. The system displays the statistics to the user.<br>4. The user can filter the statistics by offer or store.<br>5. The user can sort the statistics by date or amount. |
| Alternative(s): | • If the user has not redeemed any offers, the system displays a message indicating that there is no data to show.<br>• If the user filters the statistics by offer, the system displays the spending statistics for that offer only.<br>• If the user filters the statistics by store, the system displays the spending statistics for that store only. |
| Exception(s): | If there is an error retrieving the spending statistics from the database, the system displays an error message and prompts the user to try again later. |
| Dependency: | |

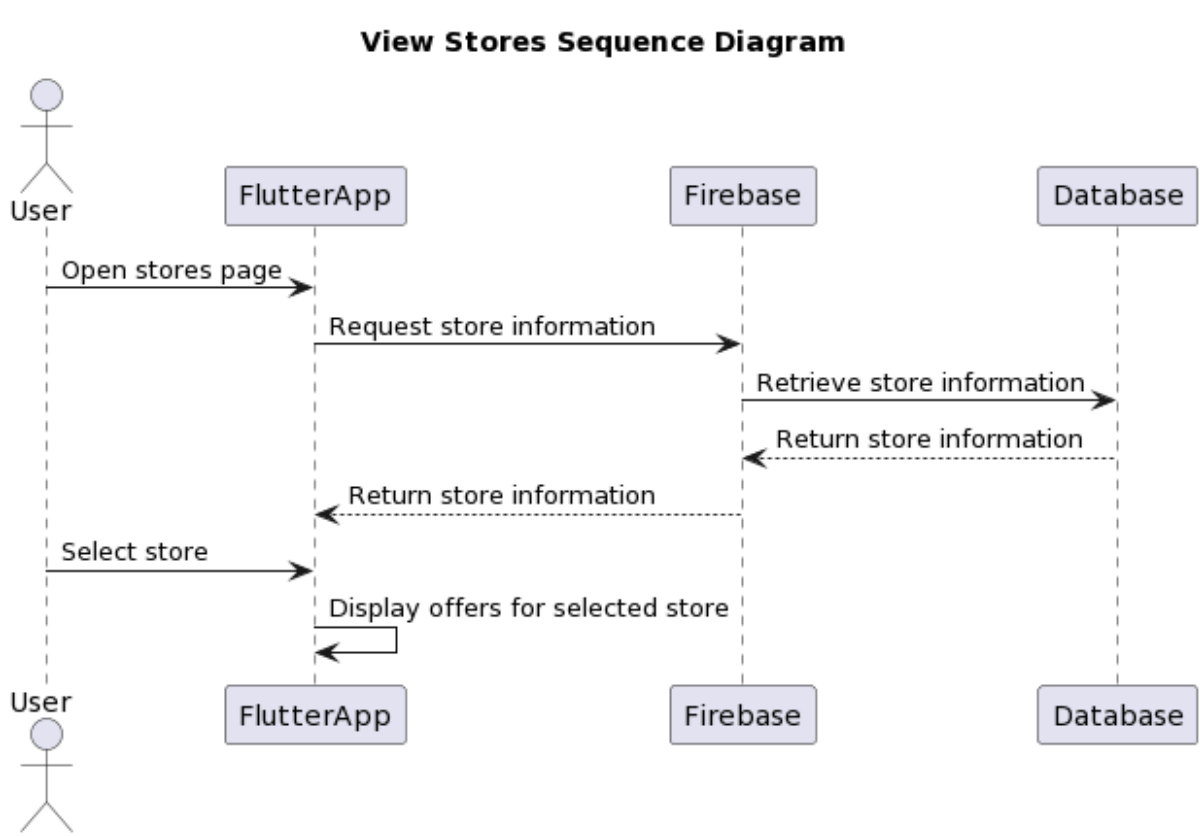
| | |
|---|---|
| Sequence Diagram |  |
| Activity Diagram |  |

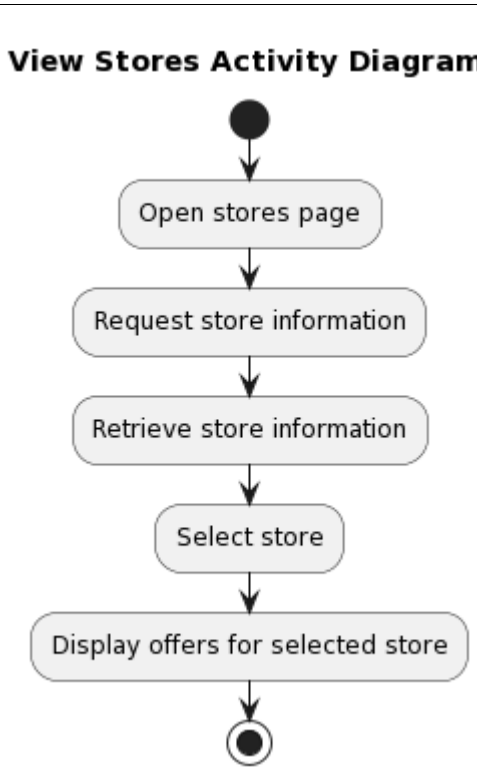
### 3.4.10 Change Account Information

| UC-10: Change Account Information | |
|---|---|
| Description: | This use case allows the user to change their account information, such as name, email, password, and profile picture. |
| Actors: | Consumer |
| Pre-Condition(s): | • The user must be logged in.<br>• The user must have access to their account information. |
| Post-Condition(s): | • The user's account information is updated in the system. |
| Main Flow: | 1. The user navigates to their account settings.<br>2. The user selects the "Edit Account" option.<br>3. The system displays the current account information.<br>4. The user makes the necessary changes to their account information.<br>5. The user saves the changes. |

| | |
|---|---|
| | 6. The system validates the updated information.<br>7. The system updates the user's account information in the database.<br>8. The system displays a message confirming that the changes have been saved. |
| Alternative(s): | If the user decides not to save the changes, they can click on the "Cancel" button and the system will not update their account information. |
| Exception(s): | If the user enters invalid or incomplete information, the system will display an error message and prompt the user to correct the errors. |
| Dependency: | |
| Sequence Diagram |  |
| Activity Diagram |  |

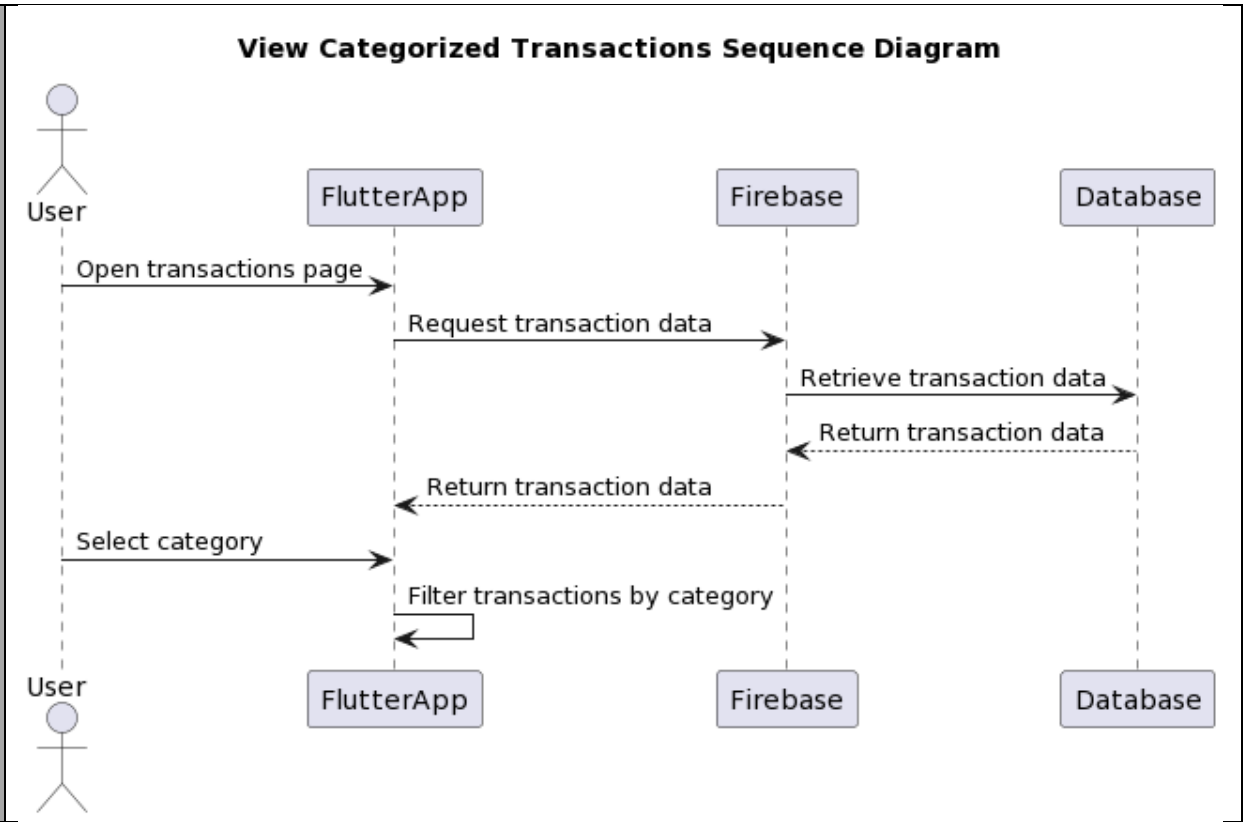
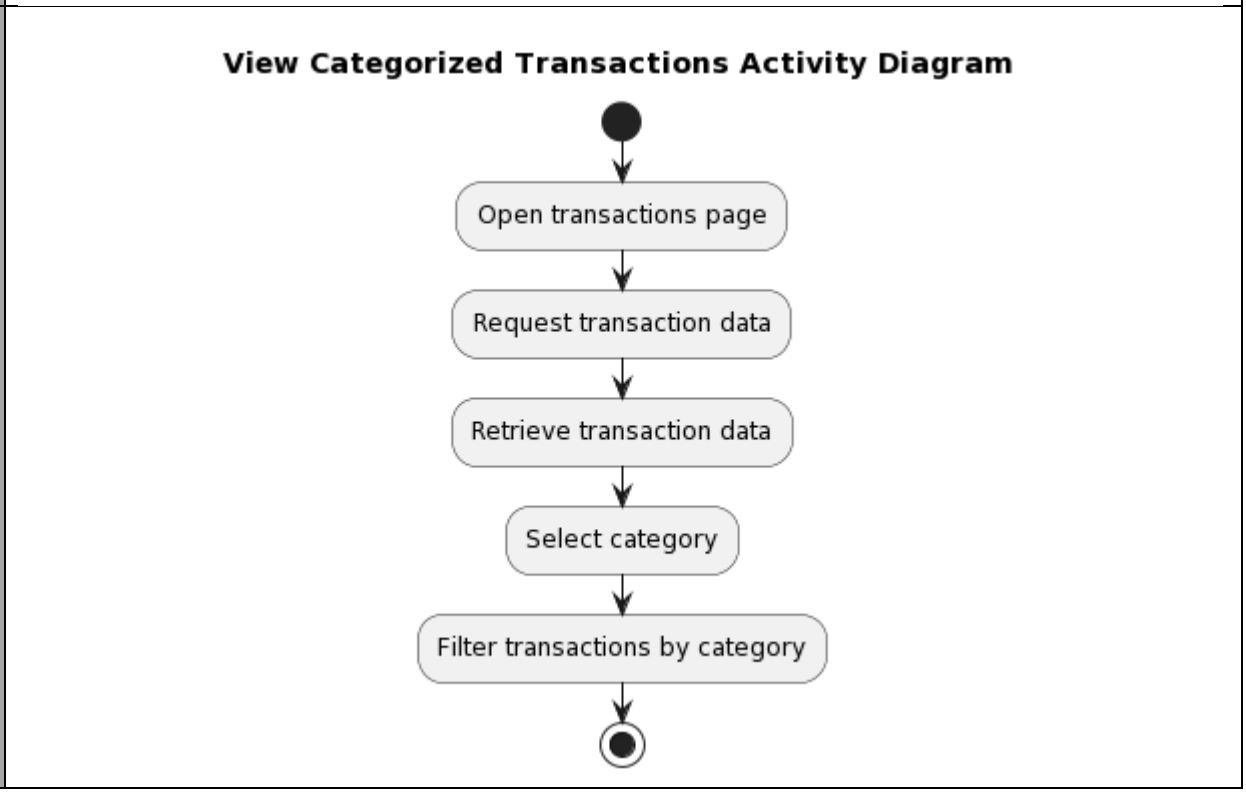
### 3.4.11 View Stores

| UC-11: View Stores | |
|---|---|
| Description: | This use case allows the user to view a list of all the stores registered in the system. The list of stores will be displayed on the user's screen. |
| Actors: | Consumer |
| Pre-Condition(s): | • The user is logged into the system.<br>• There are stores registered in the system. |
| Post-Condition(s): | • The user can view the list of stores. |
| Main Flow: | 1. The user clicks on the "View Stores" option in the navigation menu.<br>2. The system displays a list of all the stores registered in the system.<br>3. The user can scroll through the list of stores to view their names and other information.<br>4. The user can click on a store name to view more details about that store. |
| Alternative(s): | If there are no stores registered in the system, the system will display a message stating that there are no stores to view. |
| Exception(s): | If there is an error retrieving the list of stores, the system will display an error message and prompt the user to try again later. |
| Dependency: | |
| Sequence Diagram |  |

| | |
|---|---|
| Activity Diagram |  |

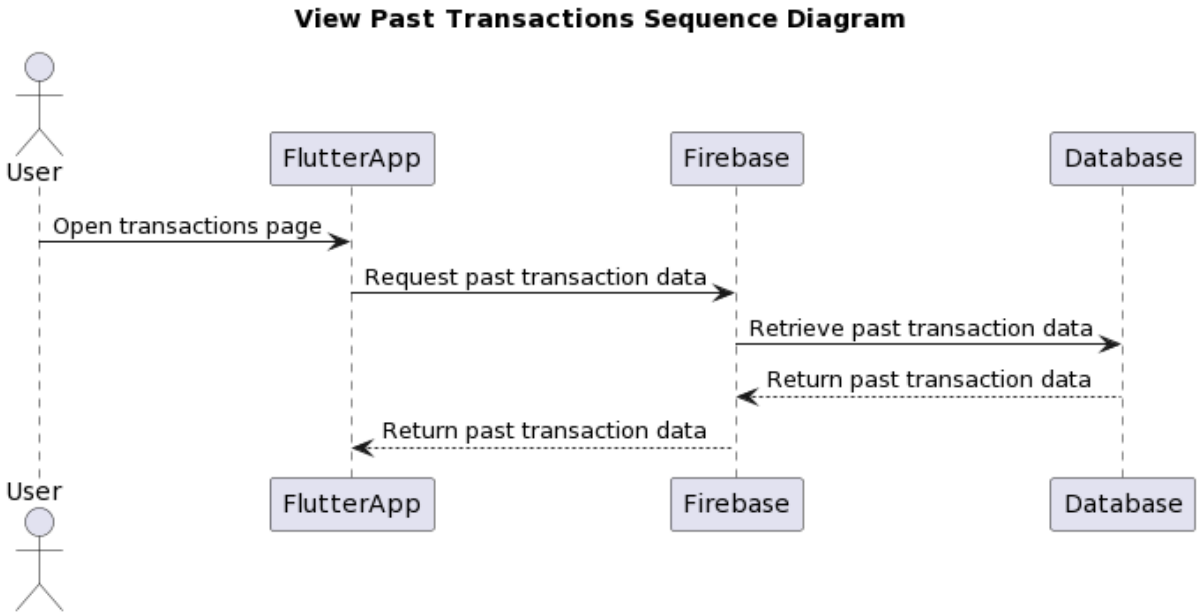
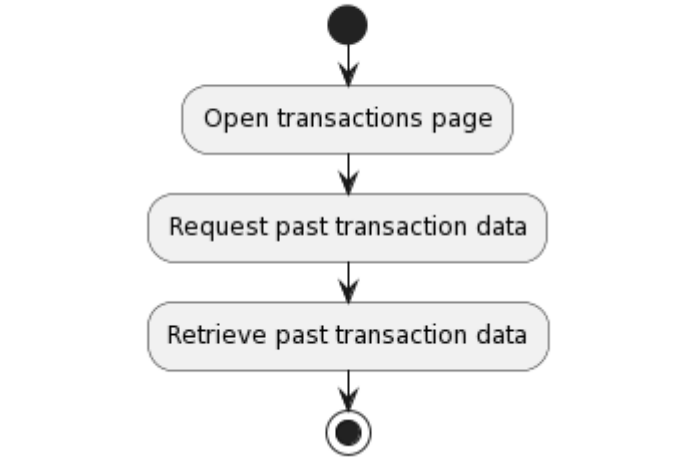
### 3.4.12 View Categorized Transactions

| UC-12: View Categorized Transactions | |
|---|---|
| Description: | This use case allows users to view their transactions categorized by type, such as food, clothing, entertainment, etc. |
| Actors: | Consumer |
| Pre-Condition(s): | • User has logged into their account.<br>• User has made transactions that are stored in the system. |
| Post-Condition(s): | • User is able to view their transactions organized by category. |
| Main Flow: | 1. User selects the "View Categorized Transactions" option from the main menu.<br>2. The system retrieves the user's transaction history from the database.<br>3. The system categorizes the transactions based on their type (food, clothing, entertainment, etc.).<br>4. The system displays the categorized transactions to the user, showing the amount spent on each category. |
| Alternative(s): | If there are no transactions available for the user, the system displays a message indicating that there are no transactions to display. |
| Exception(s): | If there is an error retrieving the user's transaction history from the database, the system displays an error message and prompts the user to try again later. |
| Dependency: | |

| | |
|---|---|
| Sequence Diagram |  |
| Activity Diagram |  |

### 3.4.13 View Past Transactions

| UC-13: View Past Transactions | |
|---|---|
| Description: | This use case allows the user to view their past transactions. |
| Actors: | Consumer |
| Pre-Condition(s): | • The user is logged in to the system.<br>• The user has made at least one transaction. |

| | |
|---|---|
| Post-Condition(s): | • The system displays the list of transactions that match the selected criteria. |
| Main Flow: | 1. The user selects the "View Past Transactions" option from the menu.<br>2. The system displays a list of available filters, such as date, store, and category.<br>3. The user selects the desired filter and enters the required information.<br>4. The system displays a list of transactions that match the selected criteria.<br>5. The user can view the details of each transaction, such as the date, store, amount, and category. |
| Alternative(s): | If the user enters invalid information in the filter fields, the system displays an error message and prompts the user to enter valid information. |
| Exception(s): | If the system encounters an error while retrieving the transaction data, it displays an error message and prompts the user to try again later. |
| Dependency: | |
| Sequence Diagram |  |
| Activity Diagram |  |

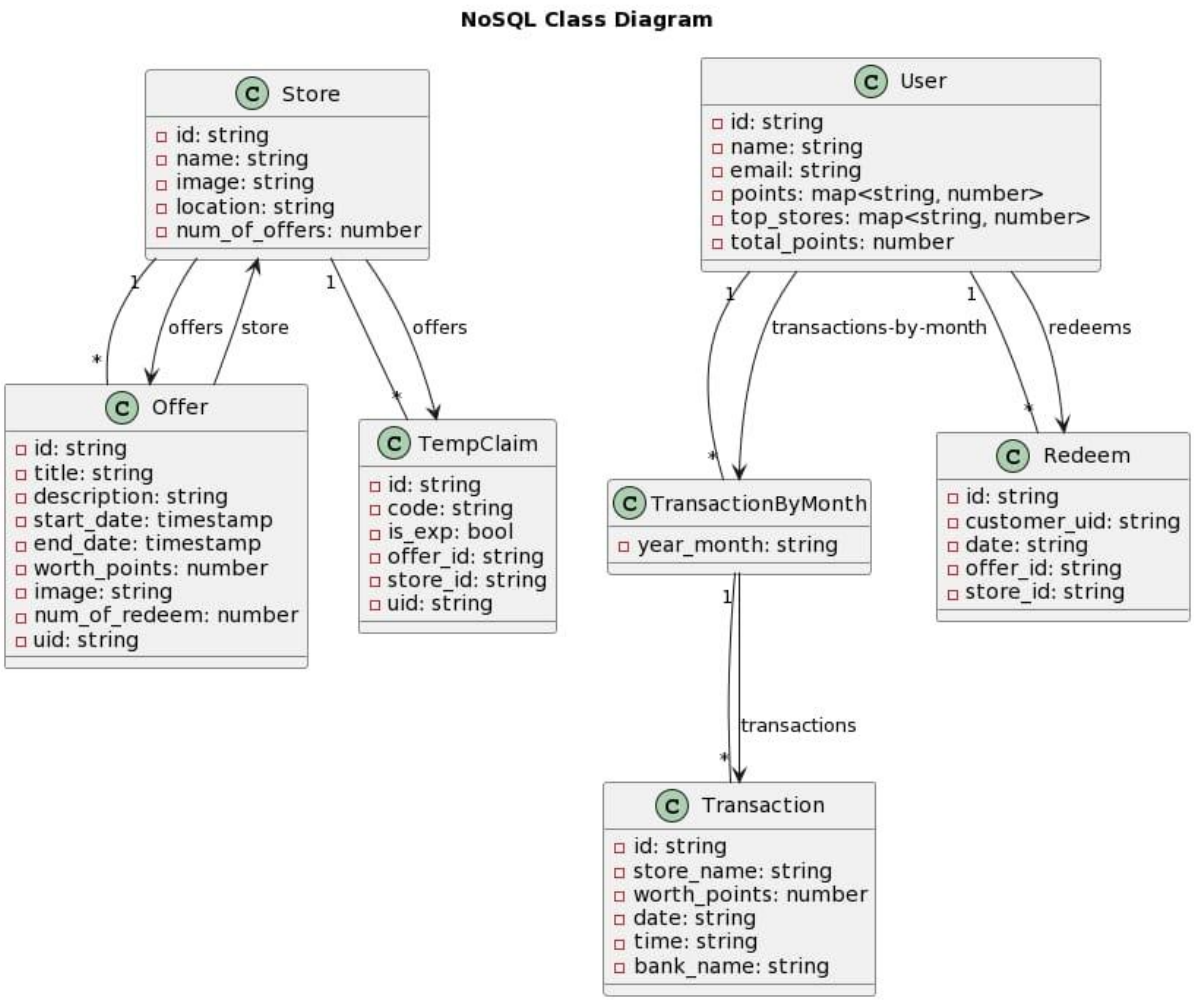
### 3.5 Performance Requirements

1. the app should be no more than 300MB in size.
2. Within 1 minute after receiving a transaction SMS message, the system should be able to analyze it and verify its sender.

3. The system should respond with the invitation link within 10 seconds after a user creates an invitation code.
4. The system should not take more than 10 seconds to show new silent notification after the user opens the app.
5. The BO dashboard should be refreshed every 10 minutes.
6. The system should be able to verify a store in less than 2 minutes using the Ministry of Commerce's API (Wathq).
7. The user should be able to generate a new invitation code with two keystrokes.
8. The system should validate a review using NLP within 3 minutes.
9. The system should be able to recommend a store that is 90% based on user purchases.
10. The login authentication process should not take more than 5 seconds.
11. The user should be able to review a store after purchase within 2 steps.
12. The BO should be able to add a new offer within 4 steps.
13. The process of submitting a new inquiry should not exceed 6 seconds
14. The time required to get the user data from the database after signing in should not exceed 12
15. seconds.
16. The number of steps to update one certain information in the user's profile should be 3 or less.
17. The number of steps for the user to select an answer for his/her inquiry should be 4 steps or less.
18. The system should complete the validation of redeeming coins in less than 12 seconds.
19. The system should complete the offer modification process within 10 seconds.
20. The BO should be able to remove an offer in 6 steps.

### 3.6 Logical Database Requirements

Our application makes use of a NoSQL Firebase database to store and manage data. Due to the dynamic nature of NoSQL databases, an Entity-Relationship Diagram (ERD) is not applicable for this type of database. However, to illustrate the structure of the database, a Class Diagram is provided in this section, which outlines the different collections and their associated attributes. The Class Diagram provides a clear understanding of the database schema and can be used as a reference for software development and maintenance. The use of a NoSQL database allows for flexible data modeling and efficient query performance, making it a suitable choice for our application's needs.




## 3.7 Software System Attributes

### 3.7.1 Security

1. 2-Step Authentication should be implemented for the login process.
2. The system should encrypt the login password for security, the encryption method will be Firebase's own Authentication method, which uses an internally modified version of script to hash account passwords.
3. The minimum password length shall be 6 characters.
4. The login verification process will not take longer than 2 seconds to process.
5. The system will verify phone numbers by OTP.
6. The system shall revoke all links associated with a store when the store closes.

### 3.7.2 Reliability

1. the application shall not consume more than 30% of the memory during the running.
2. When the system fails it should refresh itself inside the application.
3. The number of crashes must not exceed 10 for every 6 months.
4. The application shall show the user the errors if the user had any.
5. The application should do a backup every 1 hour.

### 3.7.3 Availability

1. The application will be available to the user 24/7.
2. In case of downtime, the downtime must not exceed 10 minutes.
3. The users shall be able to do anything in the application during the day.
4. All the services of the application should be available during the day.
5. Valuable information should be recovered when any failure happens.

### 3.7.4 Maintainability

1. The code should be clear and well documented when the user visits it another time.
2. The system should be high cohesion and loos coupling.
3. The subsystems of the application shall be separated to ease the maintenance.
4. The maintenance team should do maintenance every 2 months.
5. The maintenance of an urgent problem must not exceed 10 minutes.
6. The code should follow SOLID principles to ease the maintenance.
7. The code should use standard API formats.
8. Adding any additional features should be fast and the code should be easy to extend for the developers.
9. The testing shall be broken into units to ease the testing.
10. The testing for each unit should not take more than 1 hour.

### 3.7.5 Portability

1. The system shall be developed the promo Code by JSON format.
2. The system shall process reviews by using “NLP”.
3. The system shall be available to IOS devices and Android devices.
4. The application should be able to run when an update for any operating system is installed on the user’s device.
5. The web portal shall be able to run on all browsers on any operating systems.

# 4. Future Work

1. Wallet coins shall expire within a year.
   1.1. The system shall notify the user when their coins are near expiring
2. The system shall define a classification structure based on users’ XP points
   2.1. The system shall decrease the rank class if the user is inactive for a specific period.
3. The offer details should include the following:
   3.1. Type (Item, service, spending)
4. Offers are associated with rankings (labels).
5. User shall be able to see his spending in each which category (entertainment, coffee, ….).
6. The system shall allow users to add a new purchase detail.
   6.1. Manually added purchases shall not be counted within the loyalty system.
   6.2. The details need is (Date, Store Name, Purchase amount)