جامعة الملك فهد للبترول والمعادن
King Fahd University of Petroleum & Minerals

Software Design Document for MyRewards

Team#15

Ali Alsayafi
Ali Al-Aqel
Ali Albannay
Mohammed Almohammedsaleh
Emad Alomran
Abbas Sheif

Date: 3rd May 2023

Table of Contents

Introduction

Purpose

The document will explain in detail how the system is going to be established in a physical environment. The physical environment consists of the computing nodes in an internet environment, as well as other hardware and network devices. All developers such as the developing team and non-developers such as clients would read the document.

Scope

MyRewards system will create a connection between the consumer (Customer) and the producer (Business Owner).

MyRewards' website will offer a range of services to business owners so they may achieve the greatest results possible when it comes to managing sales and offers. For instance, using the charts and graphs given in the services, he or she might monitor the sales of his or her stores. The system is limited to summaries and totals and cannot display specifics; therefore, it cannot be utilized as the primary system for managing sales. The offer subsystem can be used as a tool to gather information from customers and display it on the dashboard, such as the effectiveness of the offer and whether it attracted new clients.

From the perspective of the customer, the MyRewards application will gather spending information from them to create a spending system. A combination of charts and summaries of the customer's spending will be provided by the expenditure system. The consumer, for instance, can see how much money was spent on groceries in the previous month. The consumer can browse and take advantage of a selection of offers that the Business Owner or MyRewards owner would make available. There will be various methods for claiming an offer, including spending, purchasing, and using services.

All data collecting processes will be mostly automated.

References

- SRS document
- SWE 417 instructor

Overview

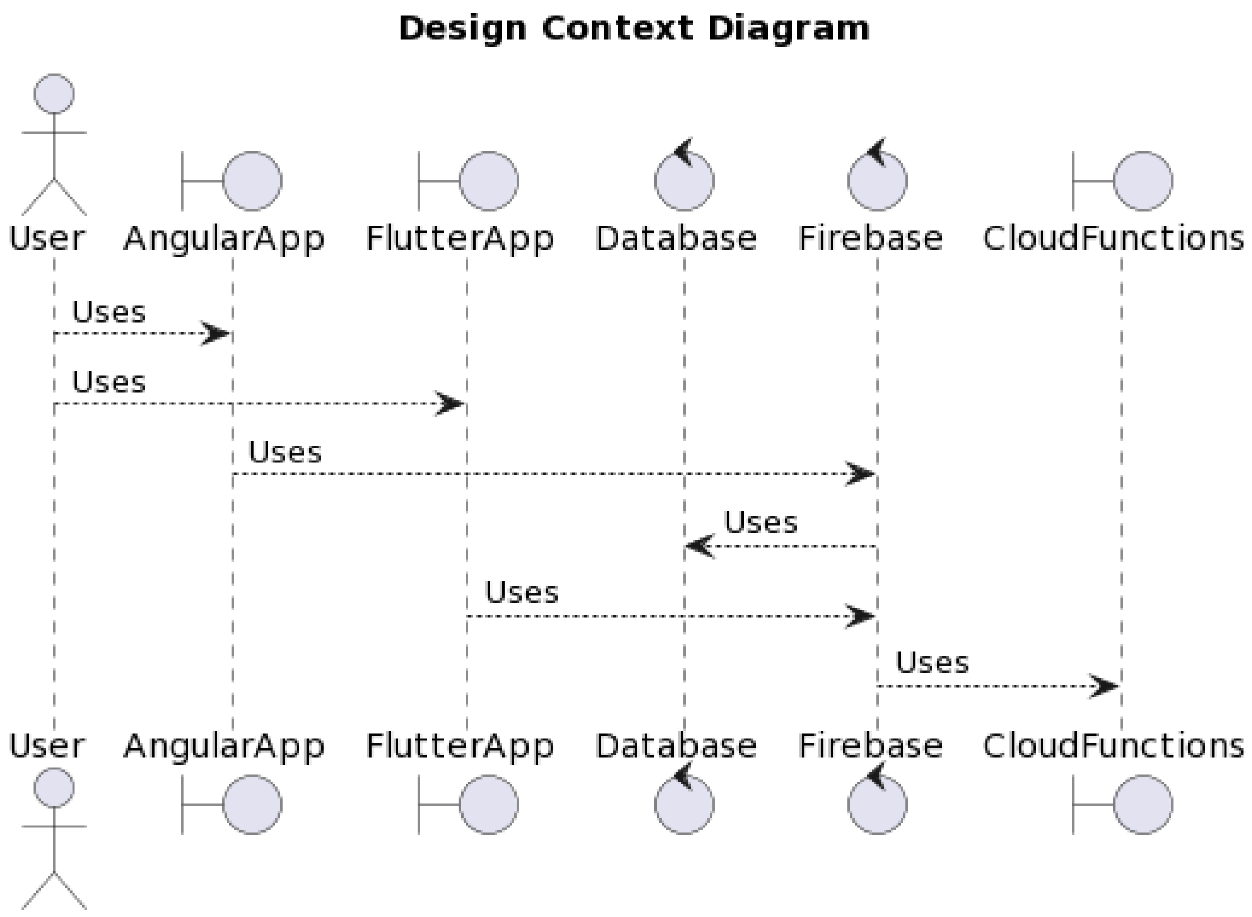
The other sections will focus on the architecture and the design of the system. includes functional specification, architectural design properties and requirements, data structures, algorithms, and methodologies. The diagrams will be used to provide summarize the information about the system.

Constraints

- The content of the payment from the SMS inbox could cause an issue. The name of the store could not be declarative, such as name of the owner and name has no relation with the store.
- The IOS system will not allow the application to read from the SMS inbox.

System Overview

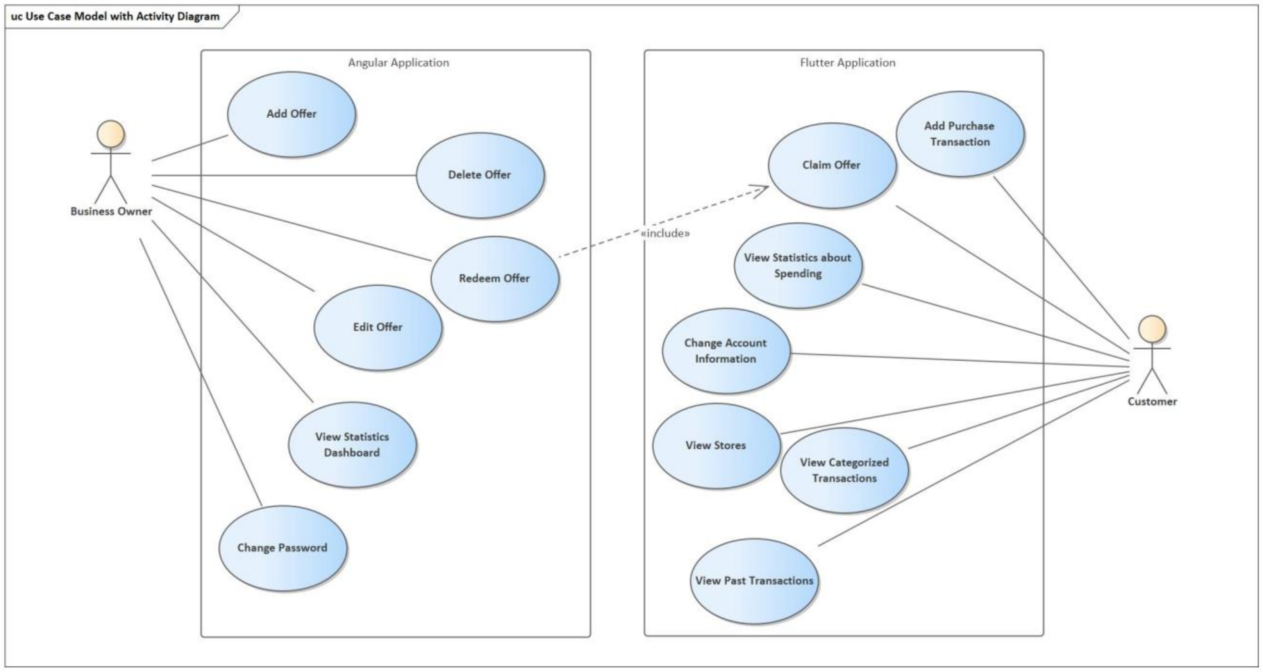
Design Context



In this diagram, the system is represented by various actors and components:
User: Represents the end-user interacting with the system.
AngularApp: Represents the Angular web application component of the system.
FlutterApp: Represents the Flutter mobile application component of the system.
Firebase: Represents the Firebase backend service used for database and cloud functions.
Database: Represents the database component within Firebase.
CloudFunctions: Represents the cloud functions component within Firebase.

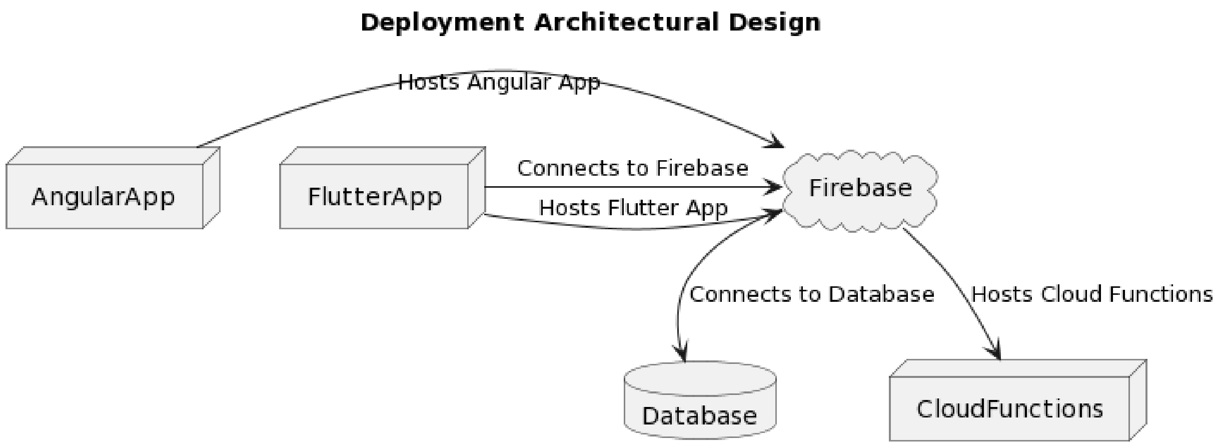
The arrows indicate the interaction and dependency between the actors and components. Users interact with the AngularApp and FlutterApp, which in turn communicate with the Firebase backend. The Firebase backend utilizes the Database and CloudFunctions components to store data and execute server-side logic.

Use Case Diagram



System Architecture

Deployment Architectural Description

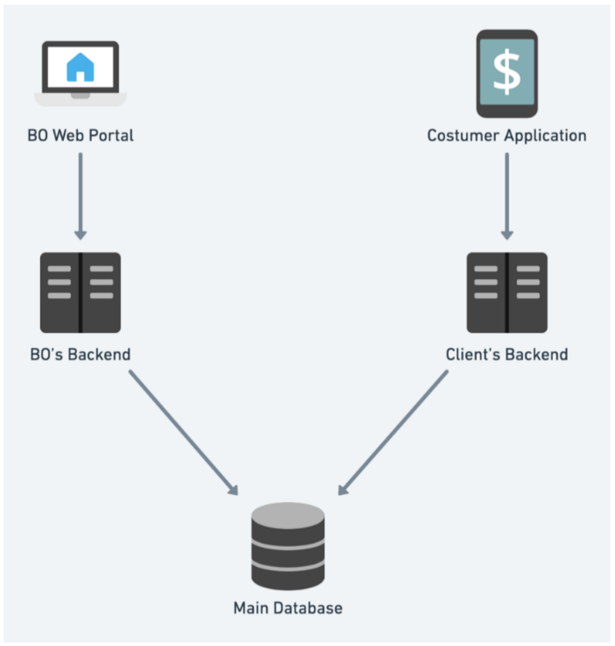


In this deployment diagram:

- AngularApp: Represents the Angular web application component.
- FlutterApp: Represents the Flutter mobile application component.
- Firebase: Represents the Firebase platform.
- FirebaseHosting: Represents the Firebase Hosting service, responsible for hosting the Angular web application.
- FirebaseFunctions: Represents the Firebase Functions service, responsible for hosting the server-side logic for the Flutter mobile application.
- Browser: Represents a web browser, which accesses the Angular web application hosted on Firebase Hosting.
- Mobile: Represents a mobile device, which interacts with the Flutter mobile application served by Firebase Functions.

The arrows represent the deployment flow, indicating how the Angular web application is delivered to the browser via Firebase Hosting, and how the Flutter mobile application is served to mobile devices through Firebase Functions.
Please note that this deployment diagram provides a simplified illustration of the deployment architecture, focusing on the cloud-based deployment using Firebase services. The actual deployment architecture may involve additional components or services depending on your specific deployment requirements and infrastructure setup.

Alternative Deployment Architectural Design Description



This architecture separates the client completely from the BO and all the sub systems that are needed inside the backend [for the sake of simplification].

Design Rationale

1. Angular Web Application:

- The decision to use Angular as the framework for the web application was based on its robustness, scalability, and extensive community support. Angular provides a structured and component-based approach to building complex web applications, which aligns with our project's requirements.
- Angular offers a wide range of built-in features and libraries for handling user interfaces, authentication, and data management, allowing for rapid development and maintenance of the application.
- Angular's ability to seamlessly integrate with Firebase as the backend platform offers a convenient and efficient way to handle real-time data synchronization, user authentication, and cloud functions.

2. Flutter Mobile Application:

- Flutter was chosen as the framework for the mobile application due to its cross-platform capabilities, enabling us to develop a single codebase that runs on both Android and iOS devices. This reduces development time and effort while ensuring a consistent user experience across platforms.
- Flutter provides a rich set of UI components and a reactive framework, enabling the development of visually appealing and performant mobile applications.
- The integration of Flutter with Firebase allows seamless data synchronization and real-time updates, providing a smooth and interactive user experience.

3. Firebase Backend:

- Firebase was selected as the backend platform for its comprehensive suite of cloud services, including real-time database, authentication, and cloud functions.
- Firebase Realtime Database offers a scalable and flexible NoSQL database solution that enables real-time synchronization and offline data access, ensuring data consistency across devices and platforms.
- Firebase Authentication provides a secure and reliable authentication system, allowing users to securely log in, sign up, and manage their accounts.
- Firebase Cloud Functions allow for the implementation of server-side logic and custom business rules, enabling the execution of complex operations and integrations with external services.

4. Database Design:

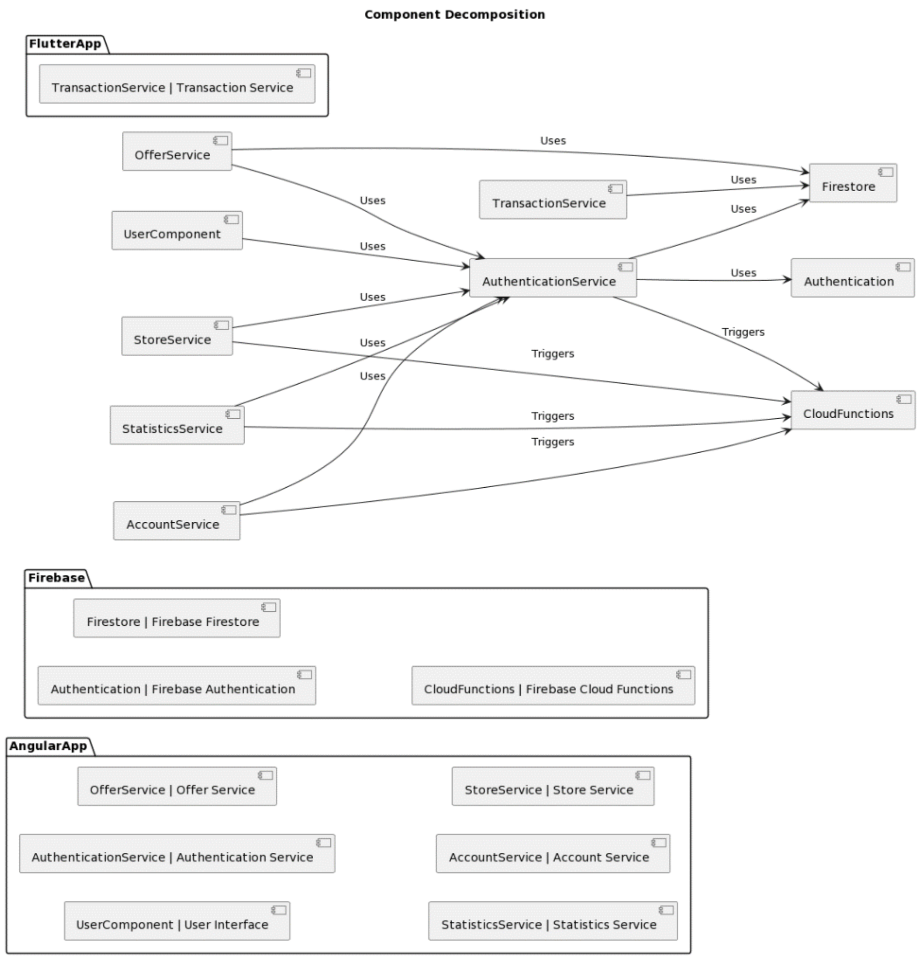
- The decision to use a NoSQL database, such as Firebase Realtime Database, was based on the project's requirements for real-time data synchronization and scalability.
- The database schema is designed to efficiently store and retrieve user data, offers, transactions, and store information while maintaining data integrity and performance.

- The use of denormalized data structures, such as storing offer information within stores and associating transactions with users and stores, allows for efficient querying and retrieval of data.

5. Cloud Functions:

- Cloud Functions are used to implement custom business logic and perform backend operations triggered by specific events, such as claim requests and transaction updates.
- By leveraging Cloud Functions, we can offload computationally intensive tasks from the client-side applications and ensure consistent and secure processing of business operations.
- The use of Cloud Functions enables the integration of external services, such as sending notifications or processing third-party APIs, providing extensibility and flexibility to the system.

Component Decomposition Description


Component Decomposition
FlutterApp
TransactionService | Transaction Service
OfferService
UserComponent
StoreService
StatisticsService
AccountService
TransactionService
AuthenticationService
Firestore
Authentication
CloudFunctions
Uses
Triggers
Firebase
Firestore | Firebase Firestore
Authentication | Firebase Authentication
CloudFunctions | Firebase Cloud Functions
AngularApp
OfferService | Offer Service
StoreService | Store Service
AuthenticationService | Authentication Service
AccountService | Account Service
UserComponent | User Interface
StatisticsService | Statistics Service

In this component diagram:

- The system is divided into three main packages: "Angular Web Application," "Flutter Mobile Application," and "Firebase Backend."
- The "Angular Web Application" package includes components related to the Angular web application, such as user interface components and various service components (authentication, offer, statistics, account, and store services).
- The "Flutter Mobile Application" package includes components related to the Flutter mobile application, such as user interface components and service components (authentication, offer, and transaction services).
- The "Firebase Backend" package represents the backend infrastructure components provided by Firebase, including Firebase Authentication, Firebase Firestore (database), and Firebase Cloud Functions.
- The arrows represent the dependencies and interactions between the components, indicating the usage and

Moreover, the detailed class diagram that shows all classes' attributes and function will be shown in section 5.1

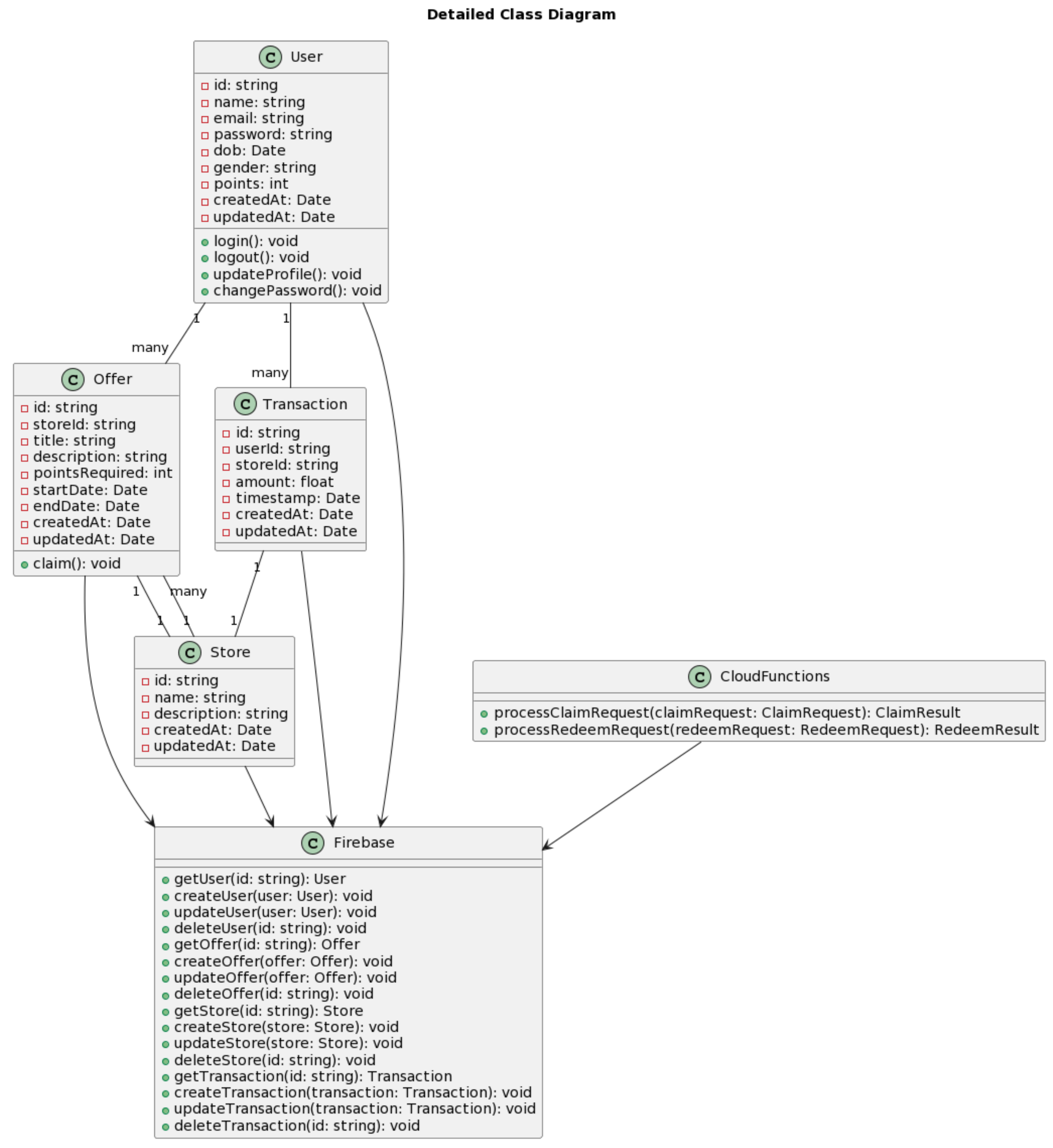
Technology selection

1- Angular: We will use it to build the webpage for the BO, adding that it is the most experienced among the team members. Moreover, it will make it easier for us while developing because we will divide the pages into components which Angular provides.
2- Flutter: It is a google framework and it has a big community that will help us when we face issues during the development of the application. Also, it has libraries that will help us to extract SMS text which is our main feature in the application.
3- Firebase

There is no hardware.

Component Design

System Components – Detailed Class Diagram


Detailed Class Diagram
User
id: string
name: string
email: string
password: string
dob: Date
gender: string
points: int
createdAt: Date
updatedAt: Date
login(): void
logout(): void
updateProfile(): void
changePassword(): void
Offer
id: string
storeId: string
title: string
description: string
pointsRequired: int
startDate: Date
endDate: Date
createdAt: Date
updatedAt: Date
claim(): void
Transaction
id: string
userId: string
storeId: string
amount: float
timestamp: Date
createdAt: Date
updatedAt: Date
Store
id: string
name: string
description: string
createdAt: Date
updatedAt: Date
CloudFunctions
processClaimRequest(claimRequest: ClaimRequest): ClaimResult
processRedeemRequest(redeemRequest: RedeemRequest): RedeemResult
Firebase
getUser(id: string): User
createUser(user: User): void
updateUser(user: User): void
deleteUser(id: string): void
getOffer(id: string): Offer
createOffer(offer: Offer): void
updateOffer(offer: Offer): void
deleteOffer(id: string): void
getStore(id: string): Store
createStore(store: Store): void
updateStore(store: Store): void
deleteStore(id: string): void
getTransaction(id: string): Transaction
createTransaction(transaction: Transaction): void
updateTransaction(transaction: Transaction): void
deleteTransaction(id: string): void
1
many
1
many
1
many
1
1
1
1
1

Functions – Sequence Diagrams

Business owner use cases

1. Add Offer

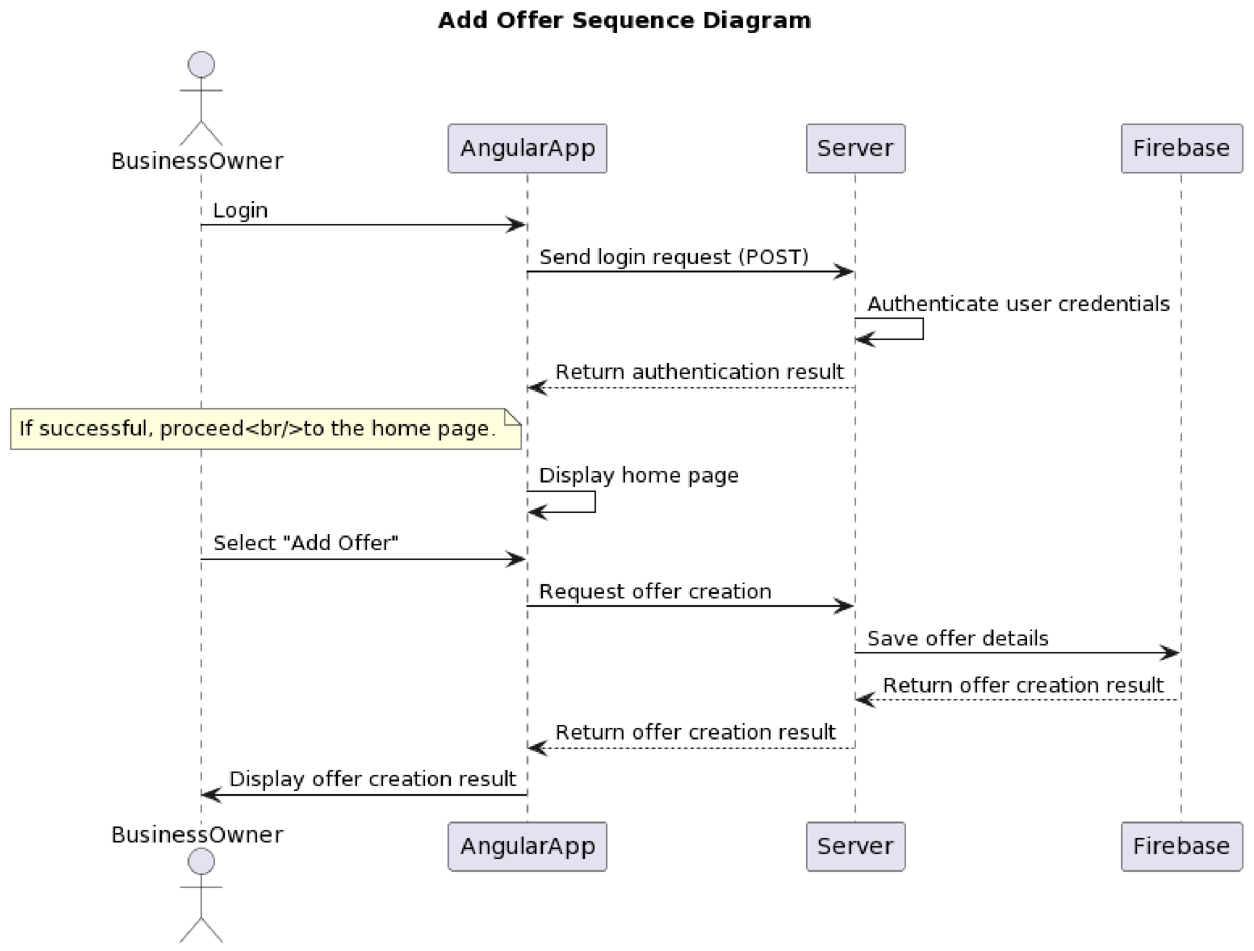

Add Offer Sequence Diagram
BusinessOwner
AngularApp
Server
Firebase
Login
Send login request (POST)
Authenticate user credentials
Return authentication result
If successful, proceed<br/>to the home page.
Display home page
Select "Add Offer"
Request offer creation
Save offer details
Return offer creation result
Return offer creation result
Display offer creation result
BusinessOwner
AngularApp
Server
Firebase

Figure 1: Add Offer Sequence Diagram

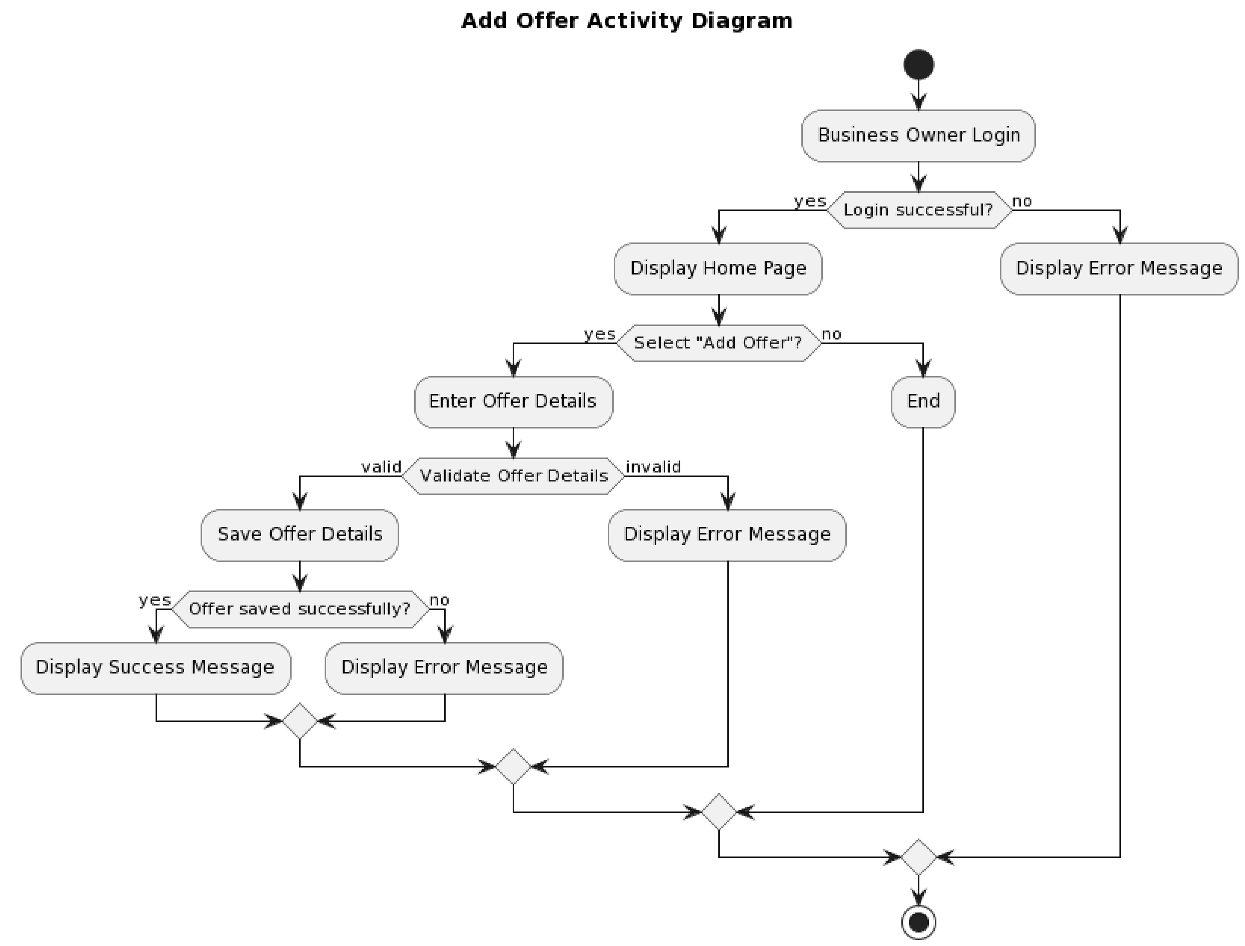
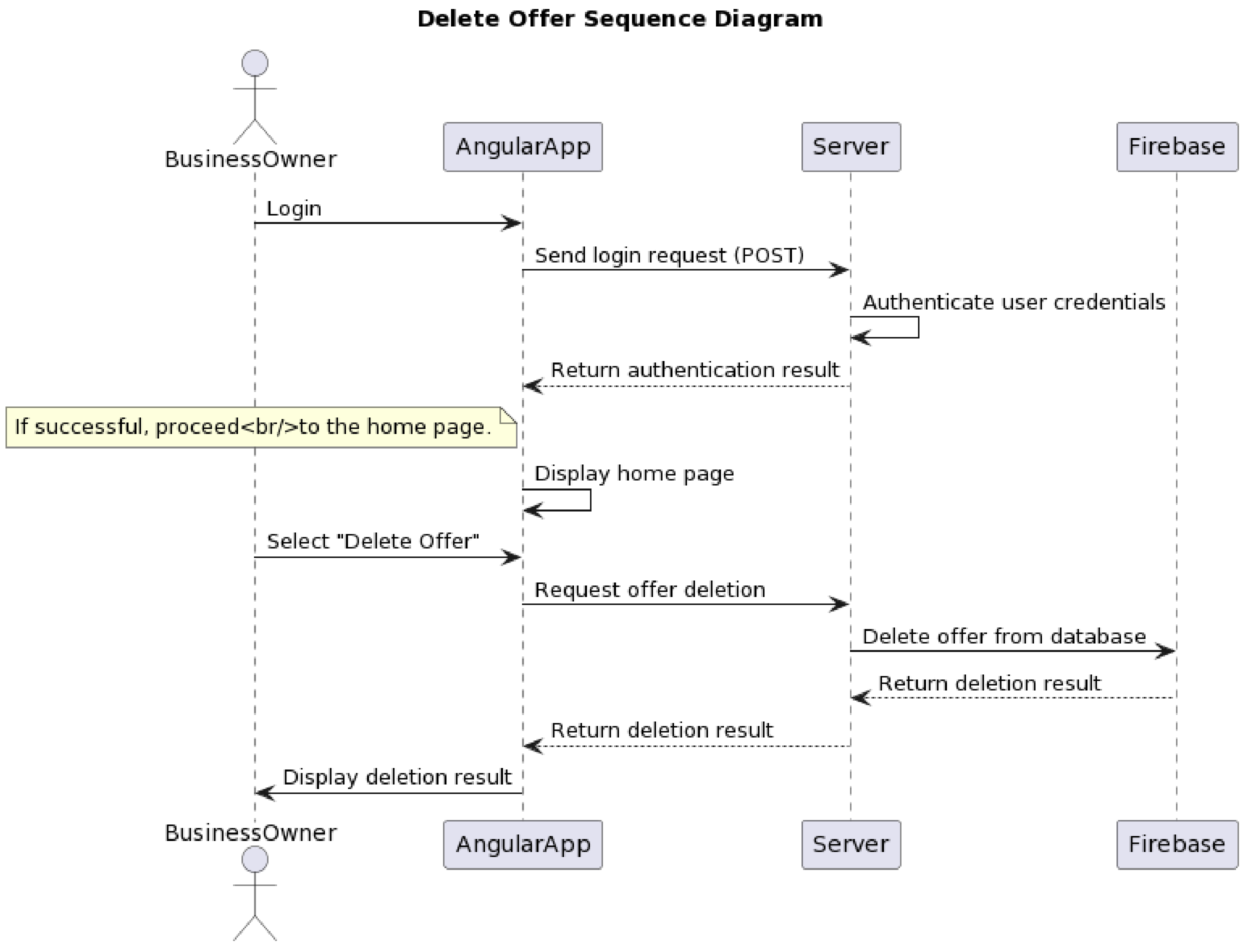


Figure 2: Add Offer Activity Diagram

2. Delete Offer


Delete Offer Sequence Diagram
BusinessOwner
AngularApp
Server
Firebase
Login
Send login request (POST)
Authenticate user credentials
Return authentication result
If successful, proceed<br/>to the home page.
Display home page
Select "Delete Offer"
Request offer deletion
Delete offer from database
Return deletion result
Return deletion result
Display deletion result
Figure 3: Delete Offer Sequence Diagram

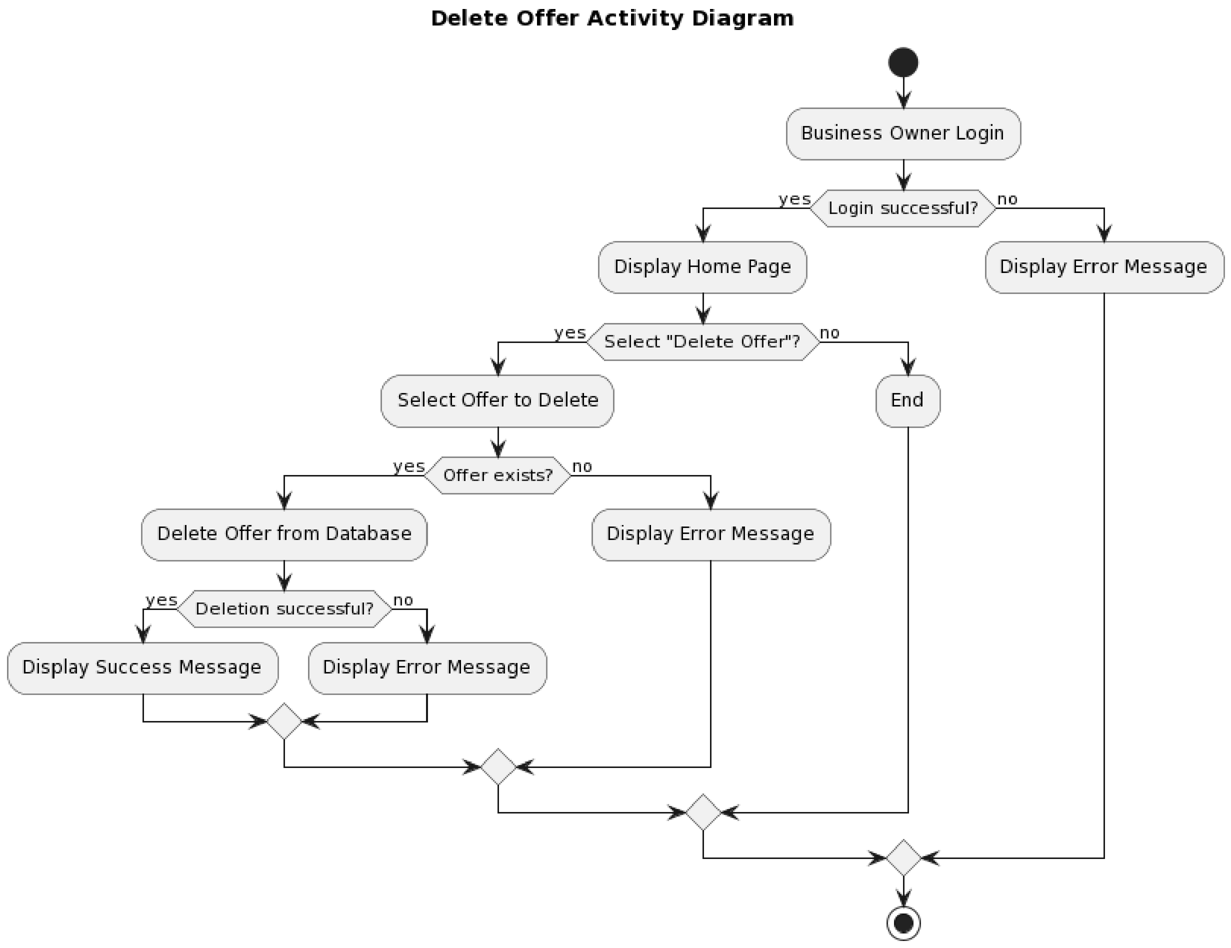


Figure 4: Delete Offer Activity Diagram

3. Edit Offer

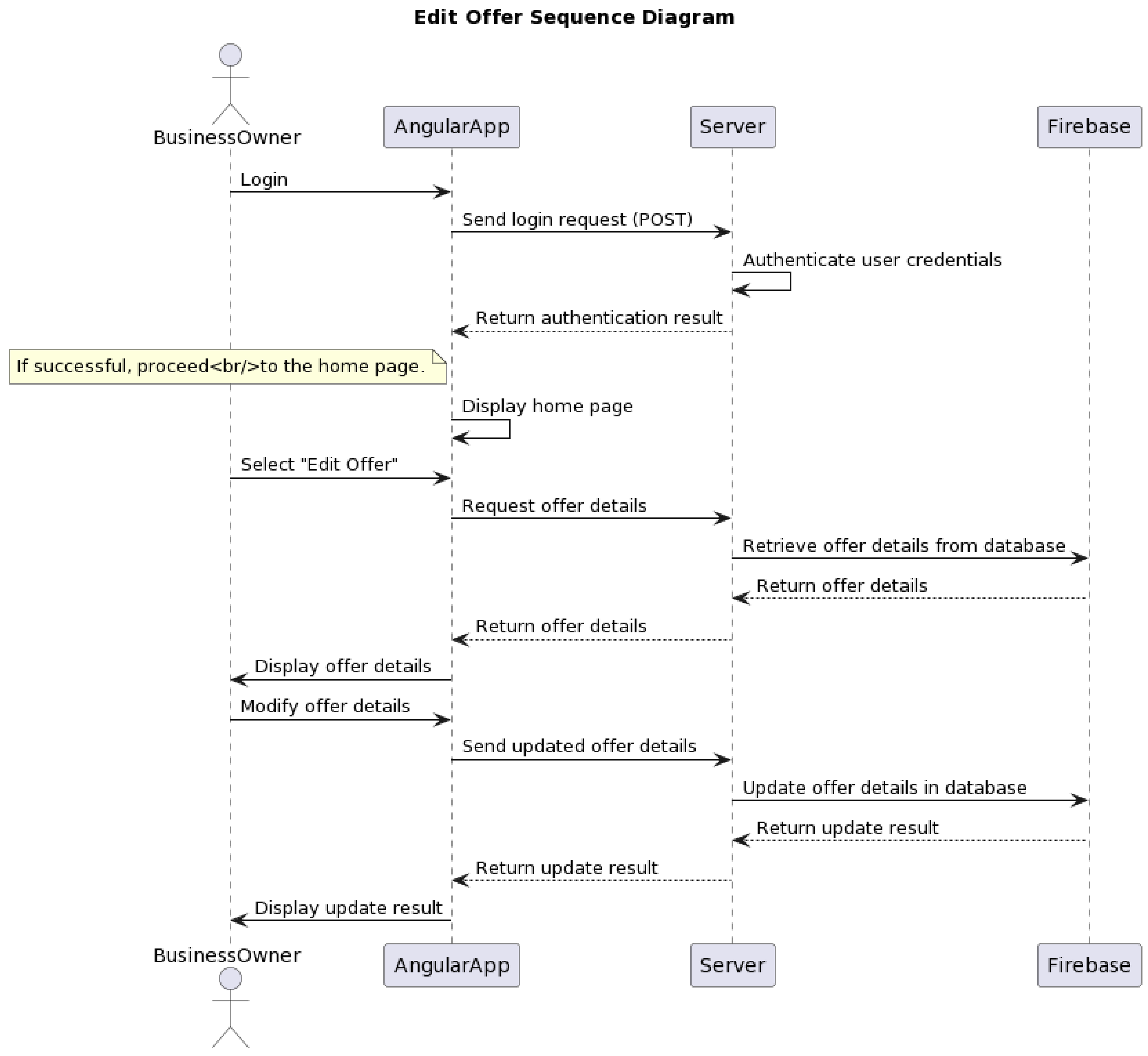

Edit Offer Sequence Diagram
BusinessOwner
AngularApp
Server
Firebase
Login
Send login request (POST)
Authenticate user credentials
Return authentication result
If successful, proceed<br/>to the home page.
Display home page
Select "Edit Offer"
Request offer details
Retrieve offer details from database
Return offer details
Return offer details
Display offer details
Modify offer details
Send updated offer details
Update offer details in database
Return update result
Return update result
Display update result

Figure 5: Edit Offer Sequence Diagram

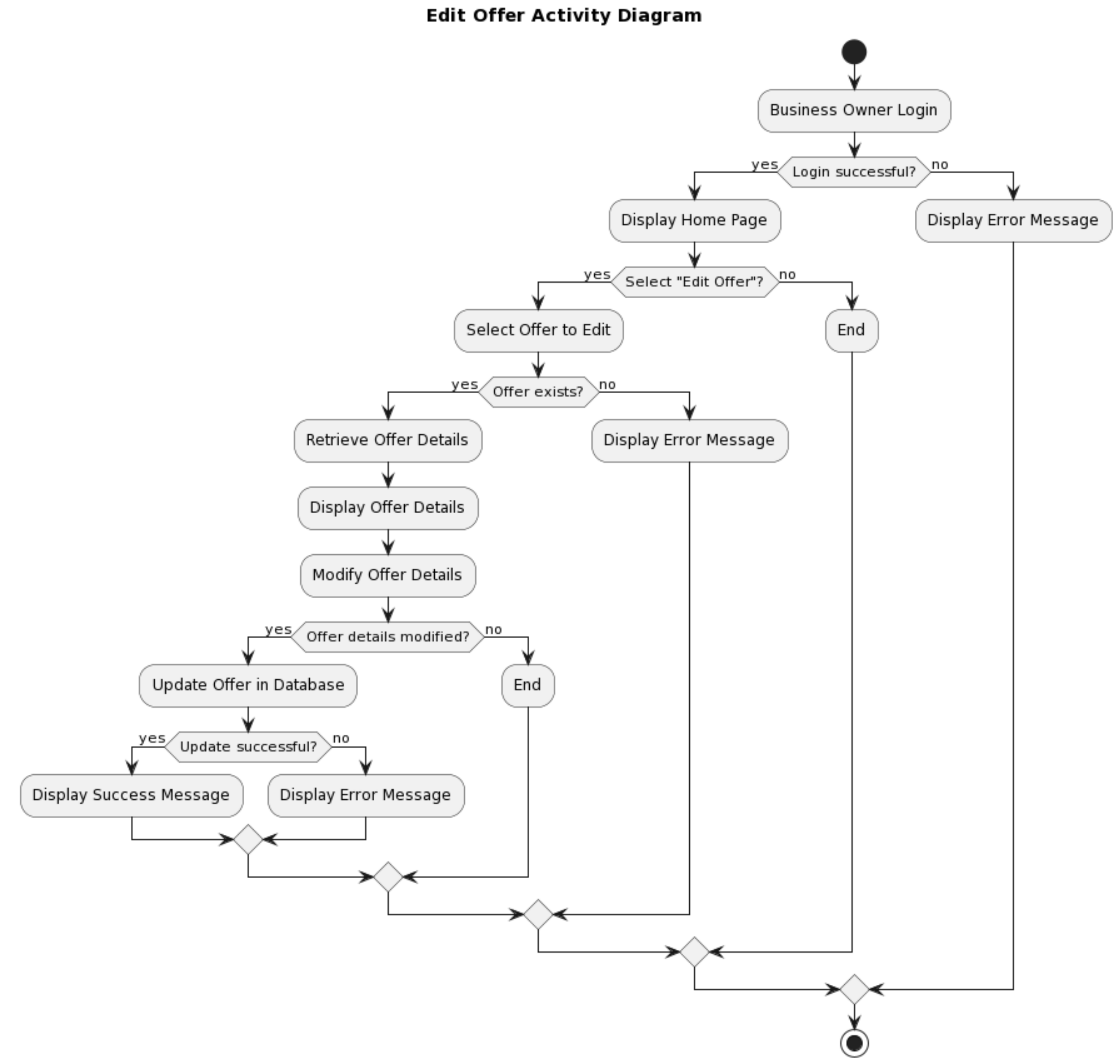


Figure 6: Edit Offer Activity Diagram

4. View Statistics Dashboard

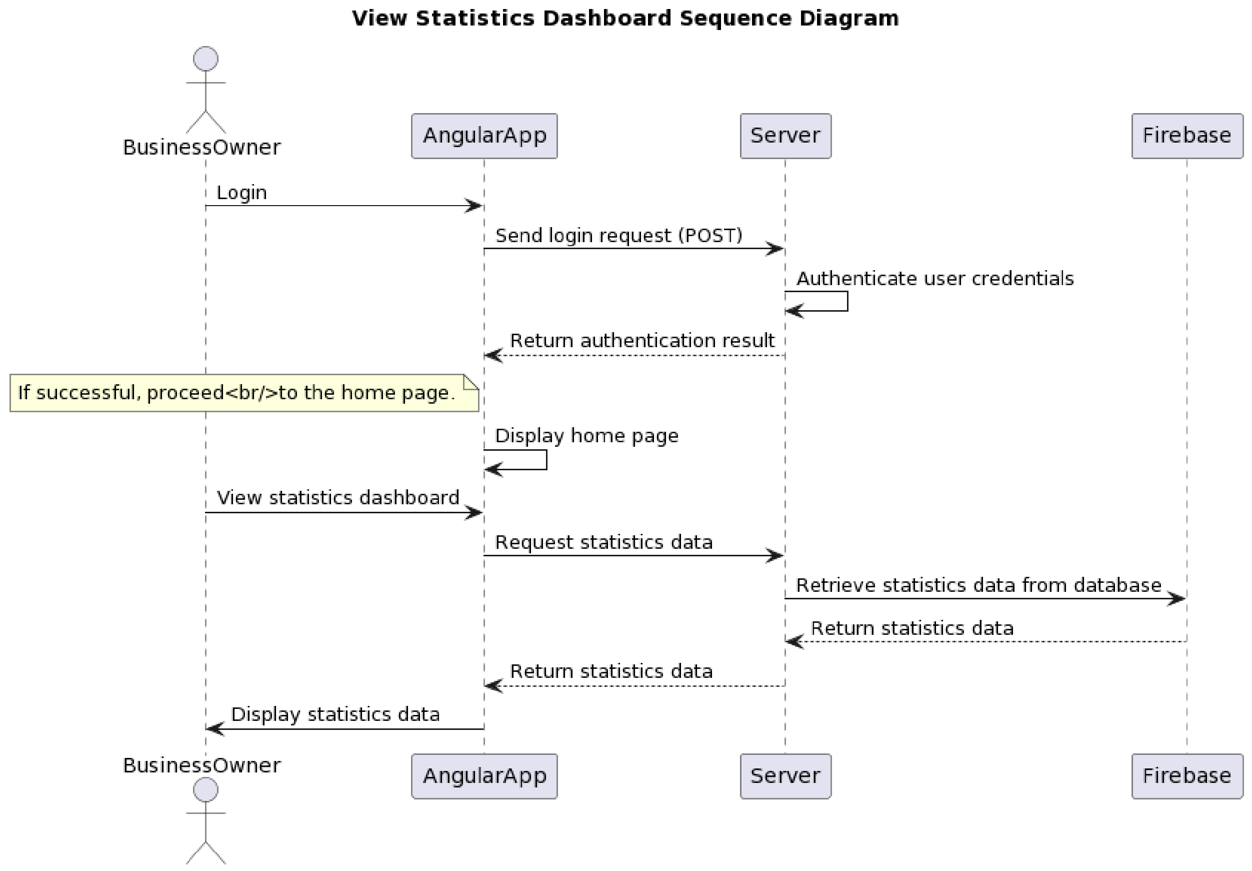

View Statistics Dashboard Sequence Diagram
BusinessOwner
AngularApp
Server
Firebase
Login
Send login request (POST)
Authenticate user credentials
Return authentication result
If successful, proceed<br/>to the home page.
Display home page
View statistics dashboard
Request statistics data
Retrieve statistics data from database
Return statistics data
Return statistics data
Display statistics data
BusinessOwner
AngularApp
Server
Firebase

Figure 7: View Statistics Dashboard Sequence Diagram

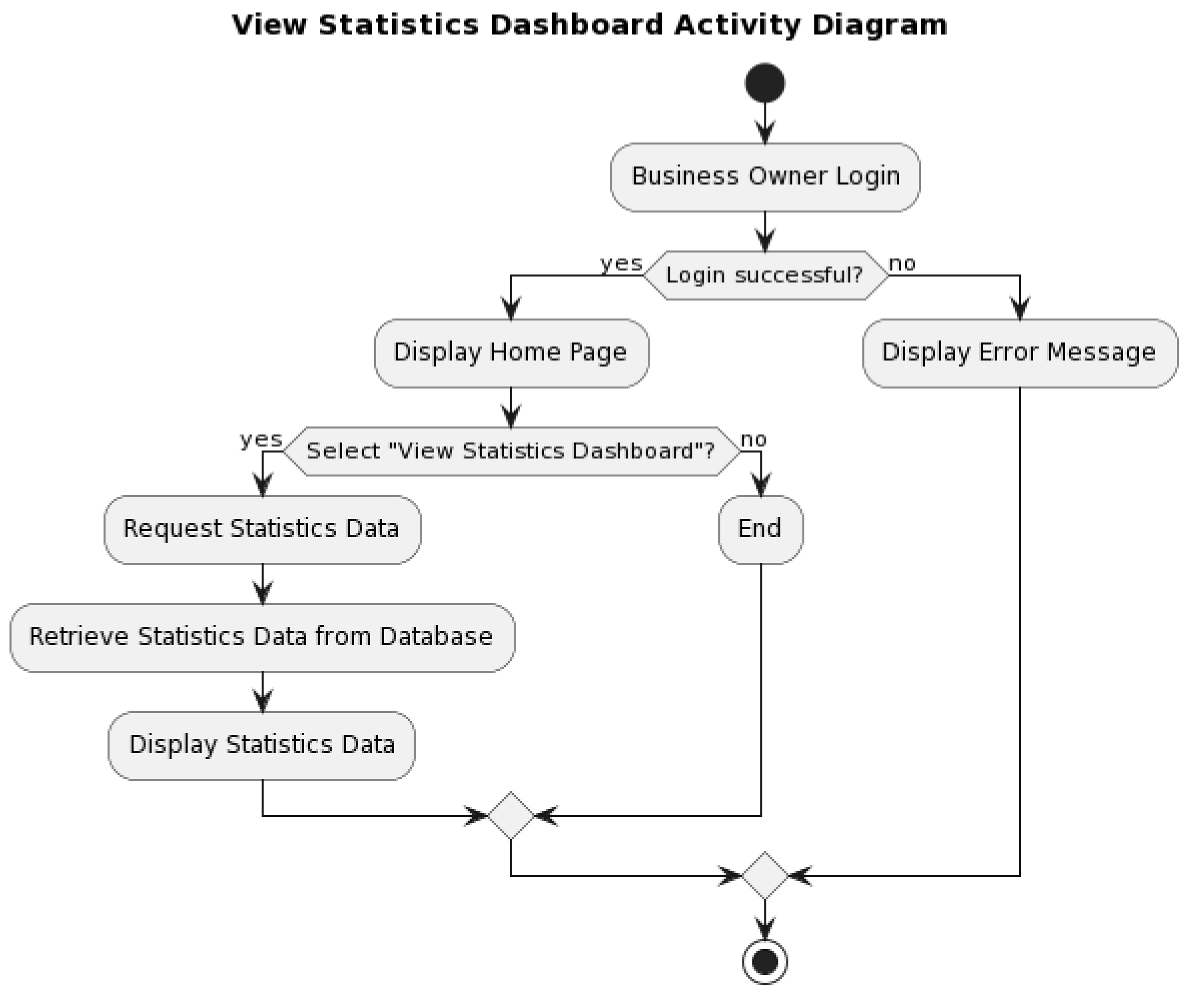


Figure 8: View Statistics Dashboard Activity Diagram

5. Redeem Offer

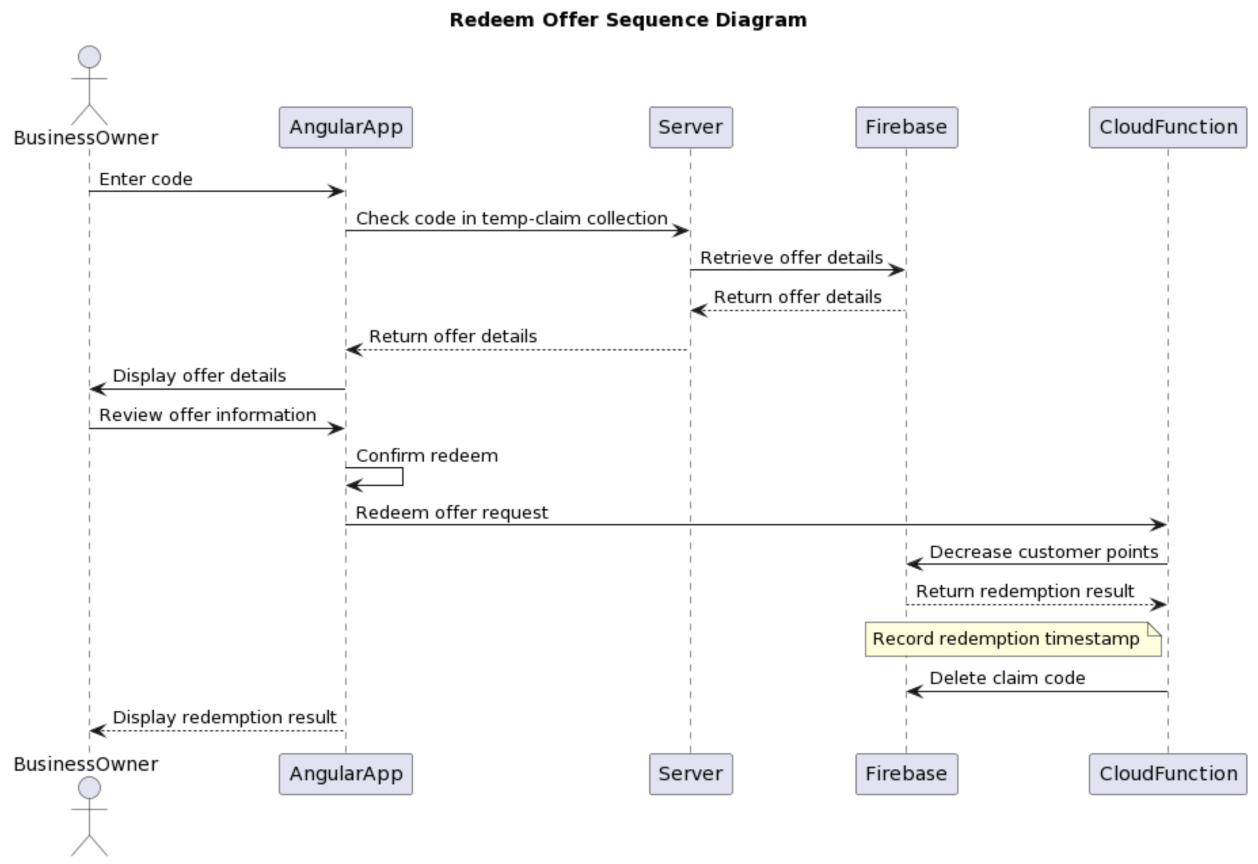


Figure 9: Redeem Offer Sequence Diagram

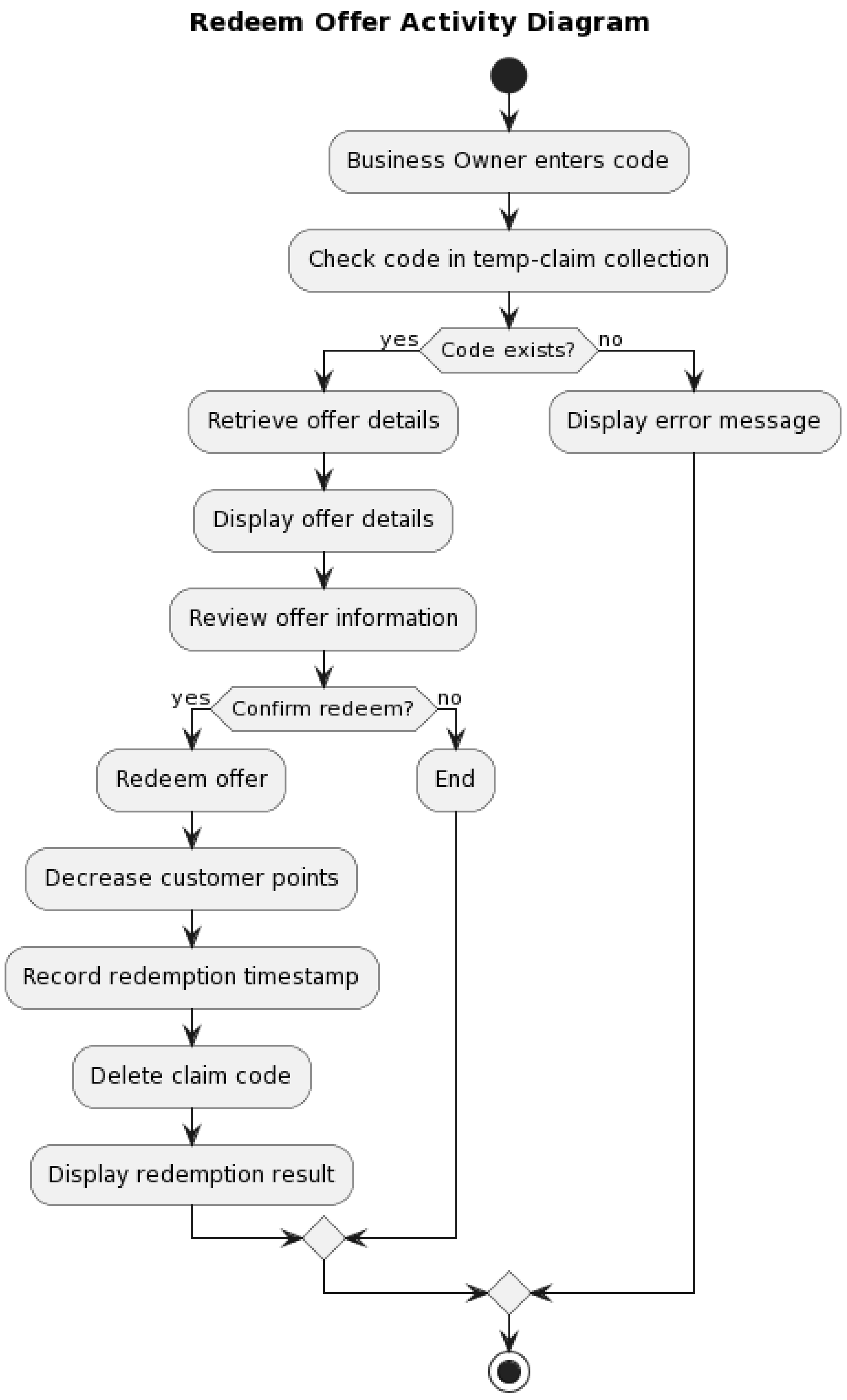
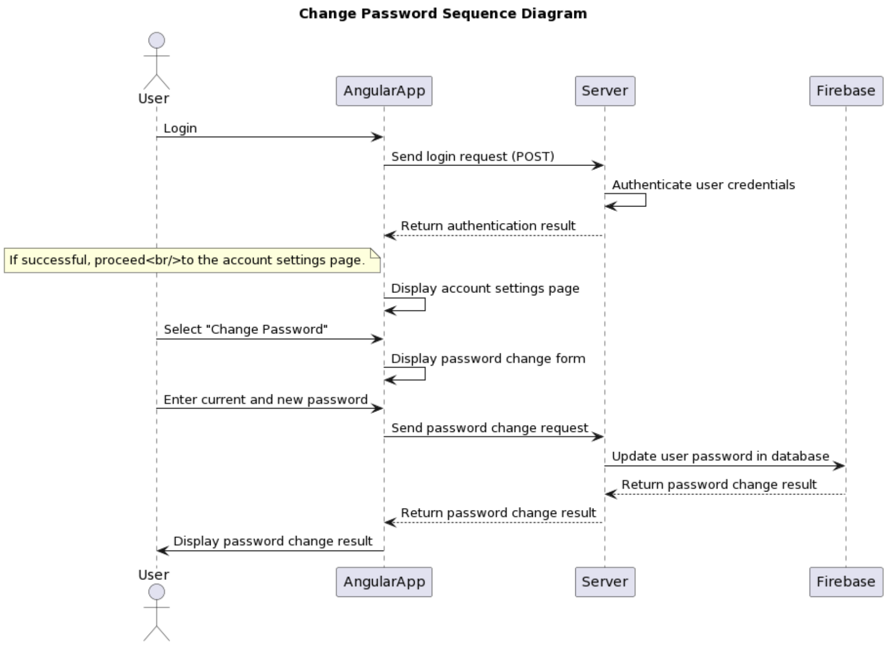
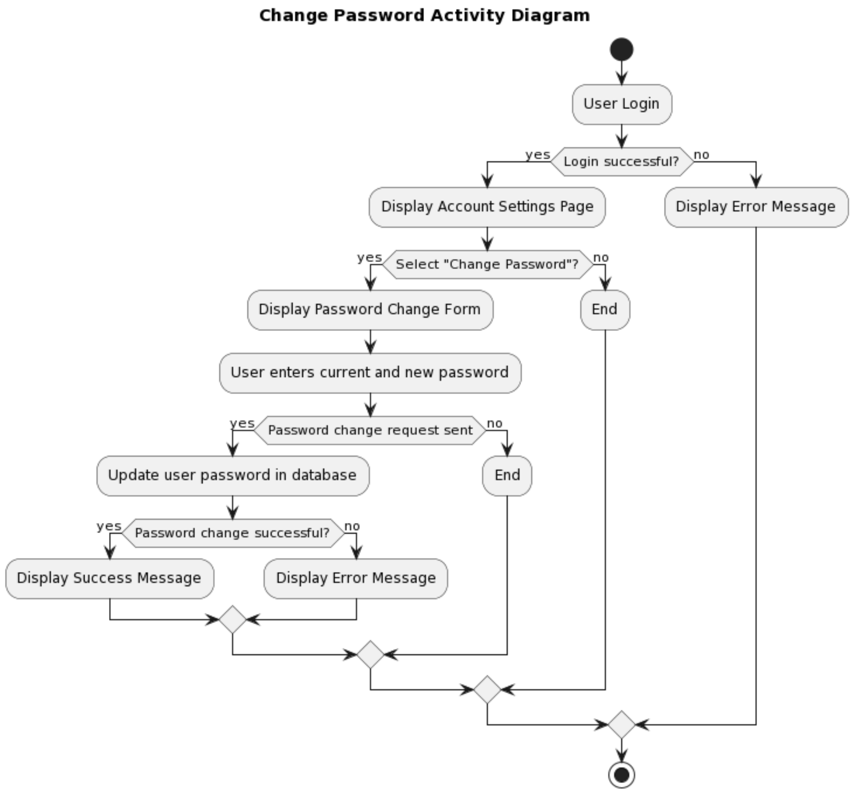


Figure 10: Redeem Offer Activity Diagram

6. Change Password

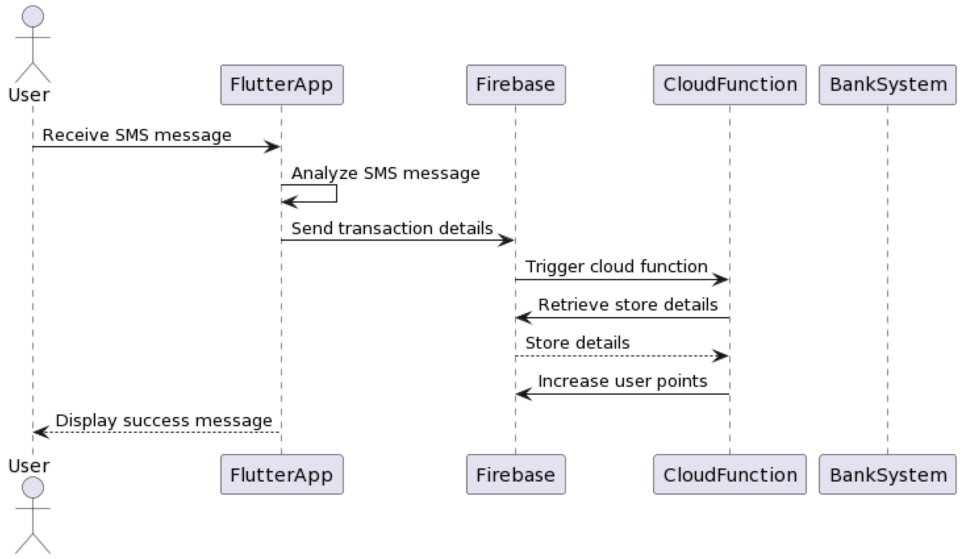

Change Password Sequence Diagram
User
AngularApp
Server
Firebase
Login
Send login request (POST)
Authenticate user credentials
Return authentication result
If successful, proceed<br/>to the account settings page.
Display account settings page
Select "Change Password"
Display password change form
Enter current and new password
Send password change request
Update user password in database
Return password change result
Return password change result
Display password change result
User
AngularApp
Server
Firebase


Change Password Activity Diagram
User Login
Login successful?
yes
no
Display Account Settings Page
Display Error Message
Select "Change Password"?
yes
no
Display Password Change Form
End
User enters current and new password
Password change request sent
yes
no
Update user password in database
End
Password change successful?
yes
no
Display Success Message
Display Error Message

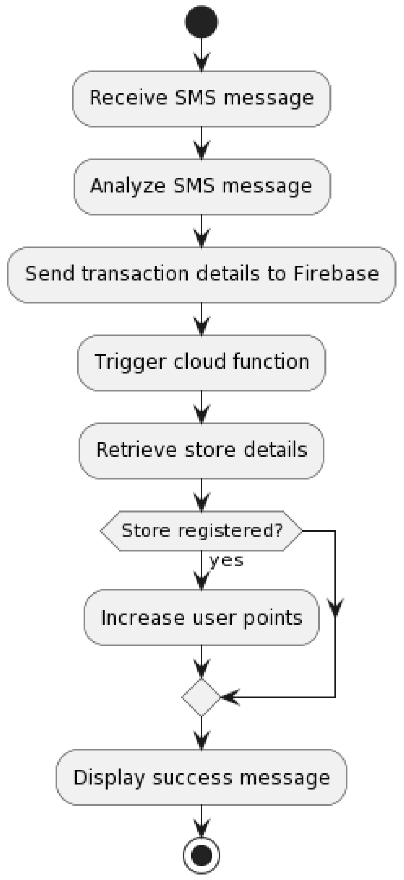
Customer use cases

1. Add Purchase Transaction

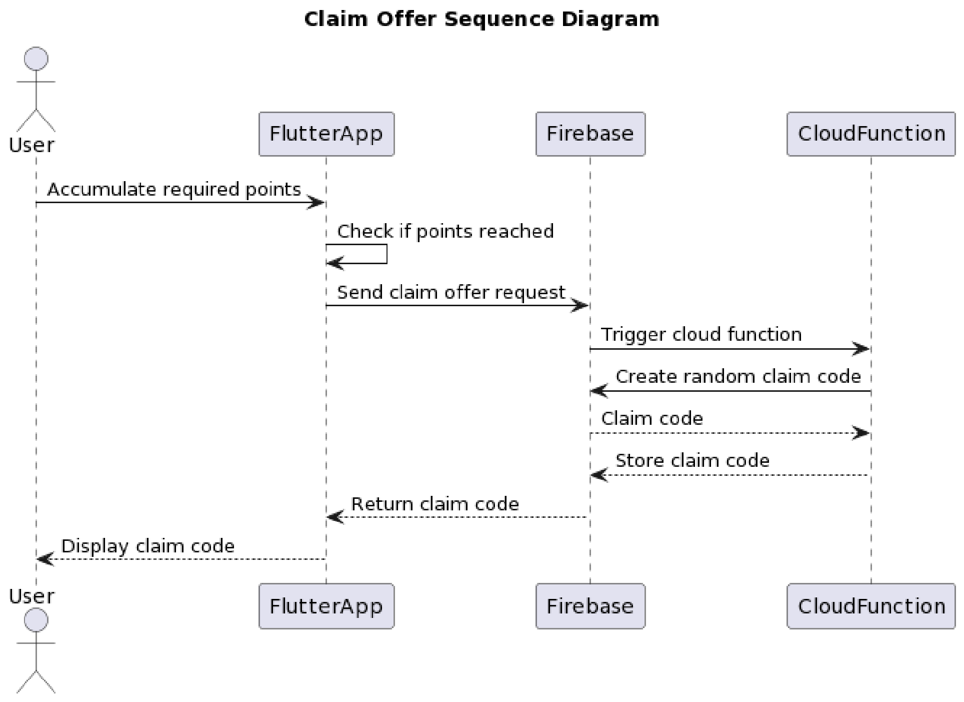
Add Purchase Transaction Sequence Diagram

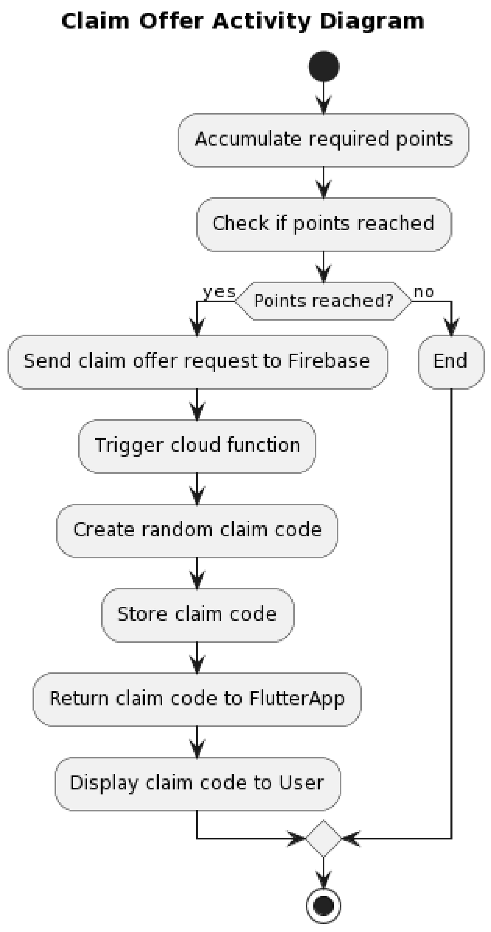


Add Purchase Transaction Activity Diagram

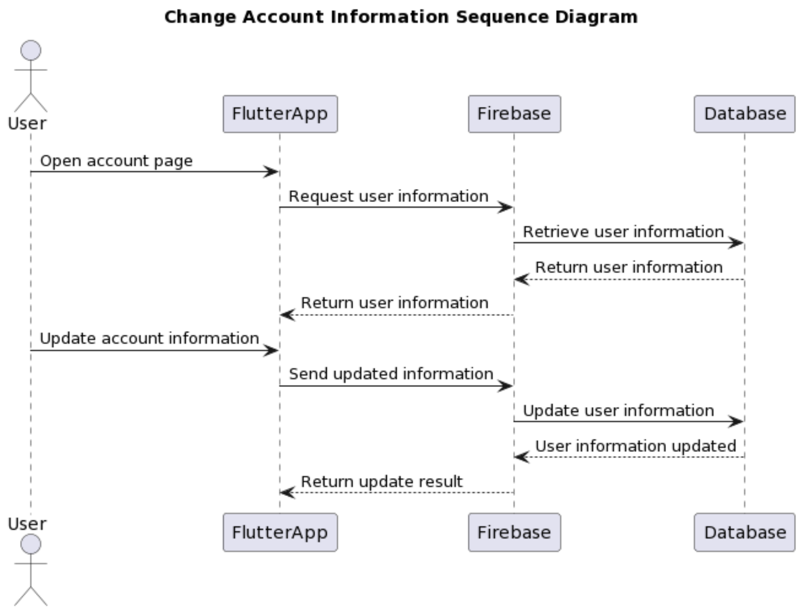


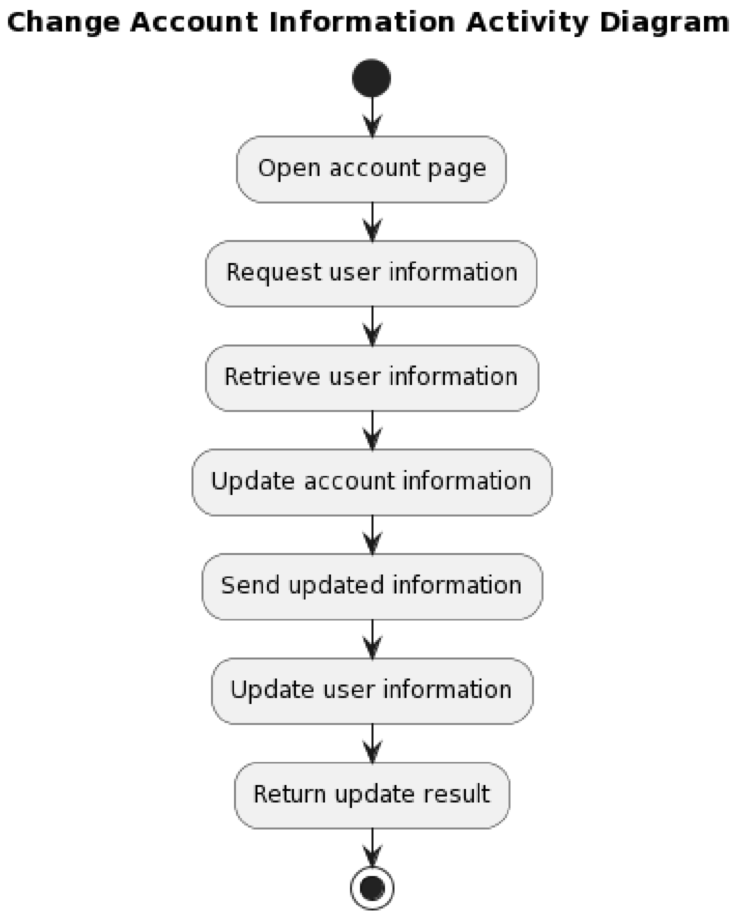
2. Claim Offer




Claim Offer Activity Diagram
Accumulate required points
Check if points reached
Points reached?
yes
no
Send claim offer request to Firebase
End
Trigger cloud function
Create random claim code
Store claim code
Return claim code to FlutterApp
Display claim code to User

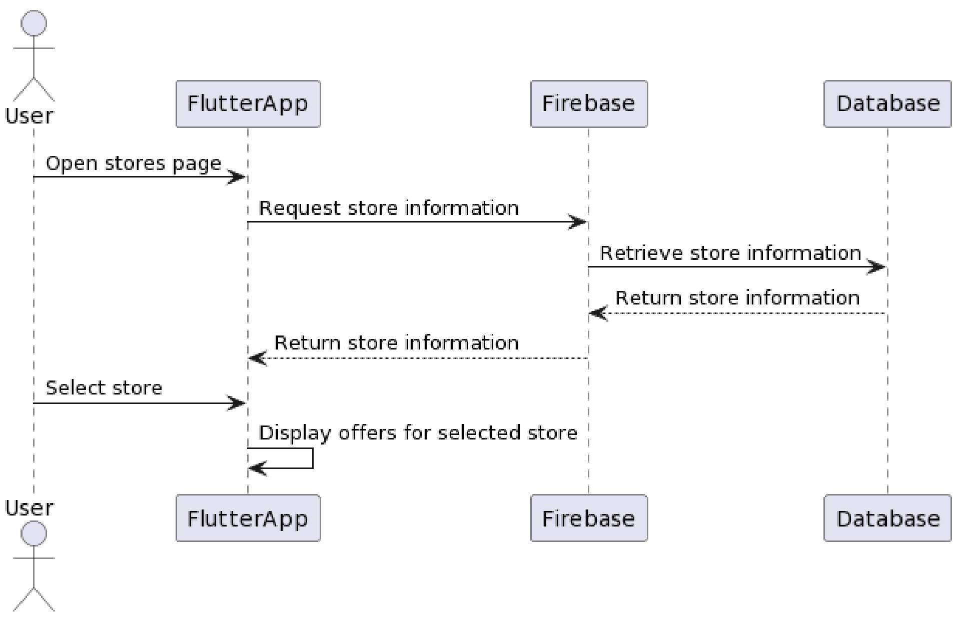
3. Change Account Information



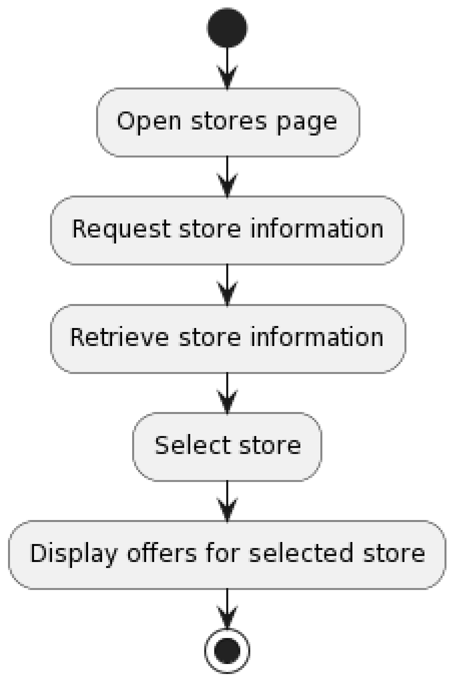


4. View Stores

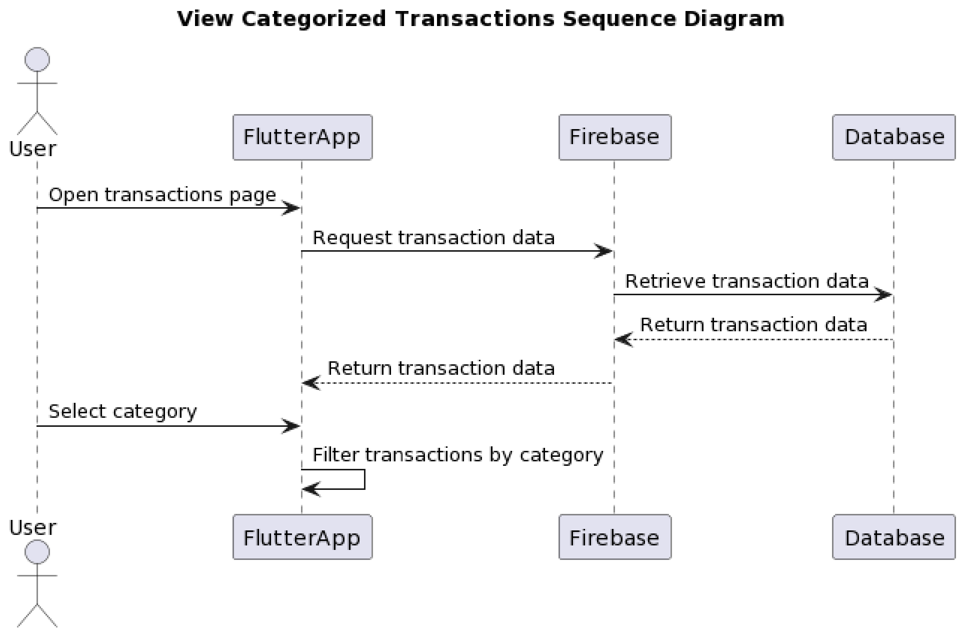
View Stores Sequence Diagram

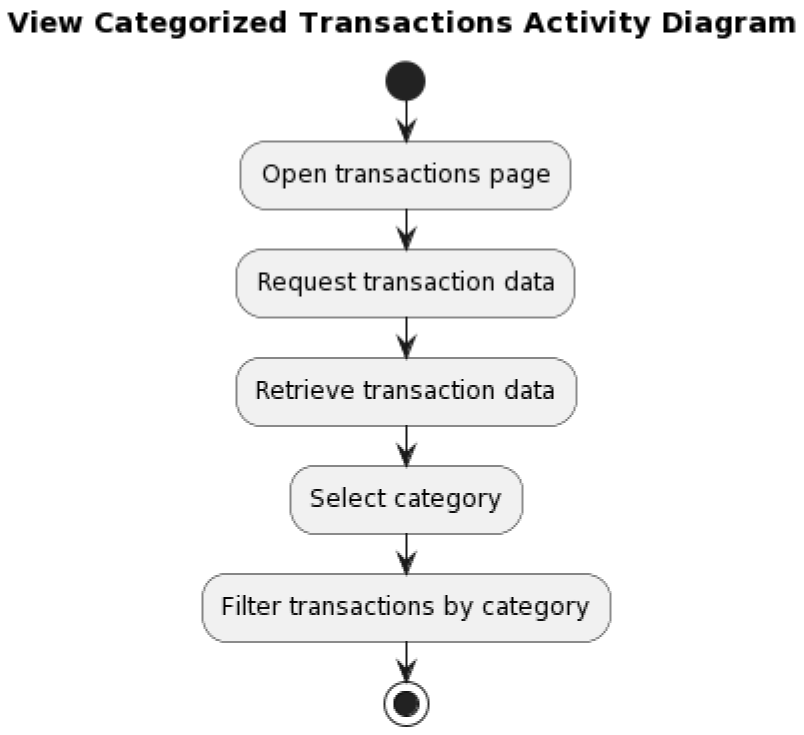


View Stores Activity Diagram

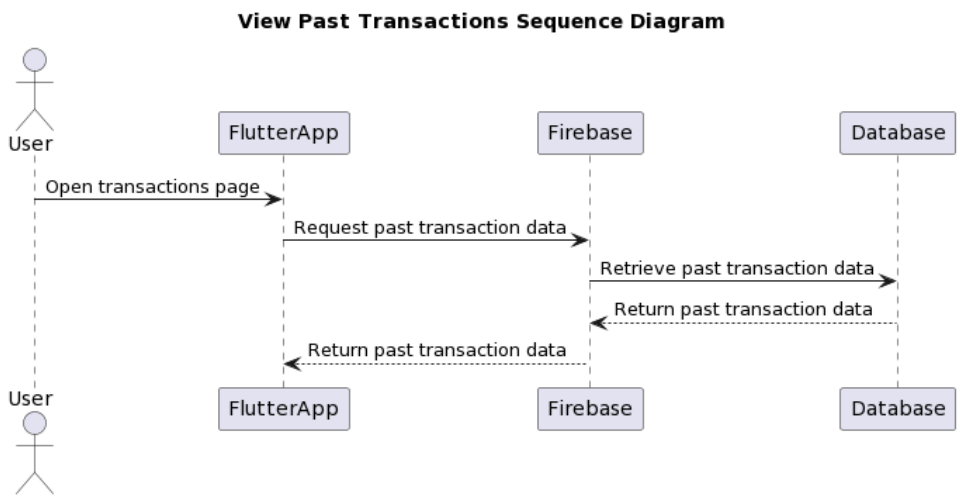

Open stores page
Request store information
Retrieve store information
Select store
Display offers for selected store

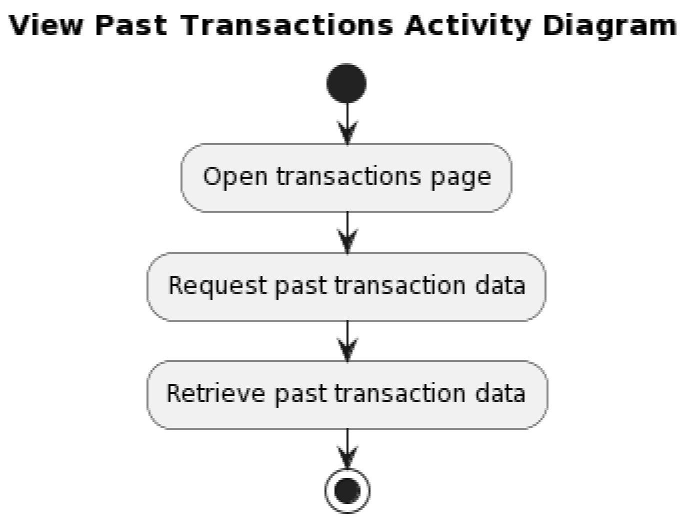
5. View Categorized Transactions




View Categorized Transactions Activity Diagram
Open transactions page
Request transaction data
Retrieve transaction data
Select category
Filter transactions by category

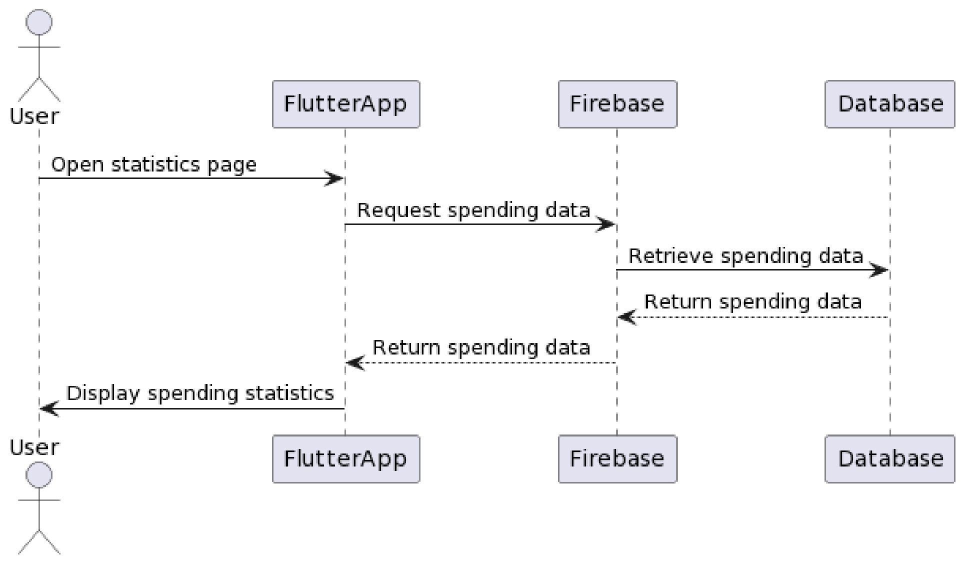
6. View Past Transactions


View Past Transactions Sequence Diagram
User
FlutterApp
Firebase
Database
Open transactions page
Request past transaction data
Retrieve past transaction data
Return past transaction data
Return past transaction data

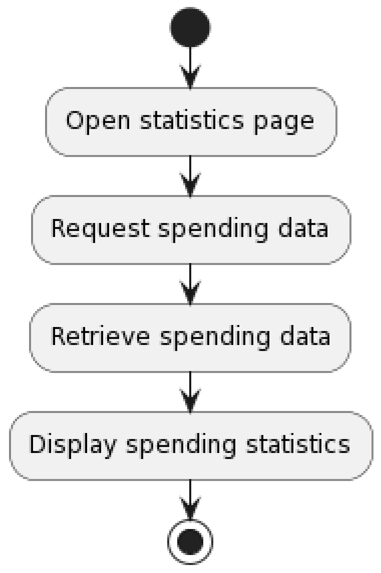

View Past Transactions Activity Diagram
Open transactions page
Request past transaction data
Retrieve past transaction data

7. View Statistics about Spending

View Statistics about Spending Sequence Diagram

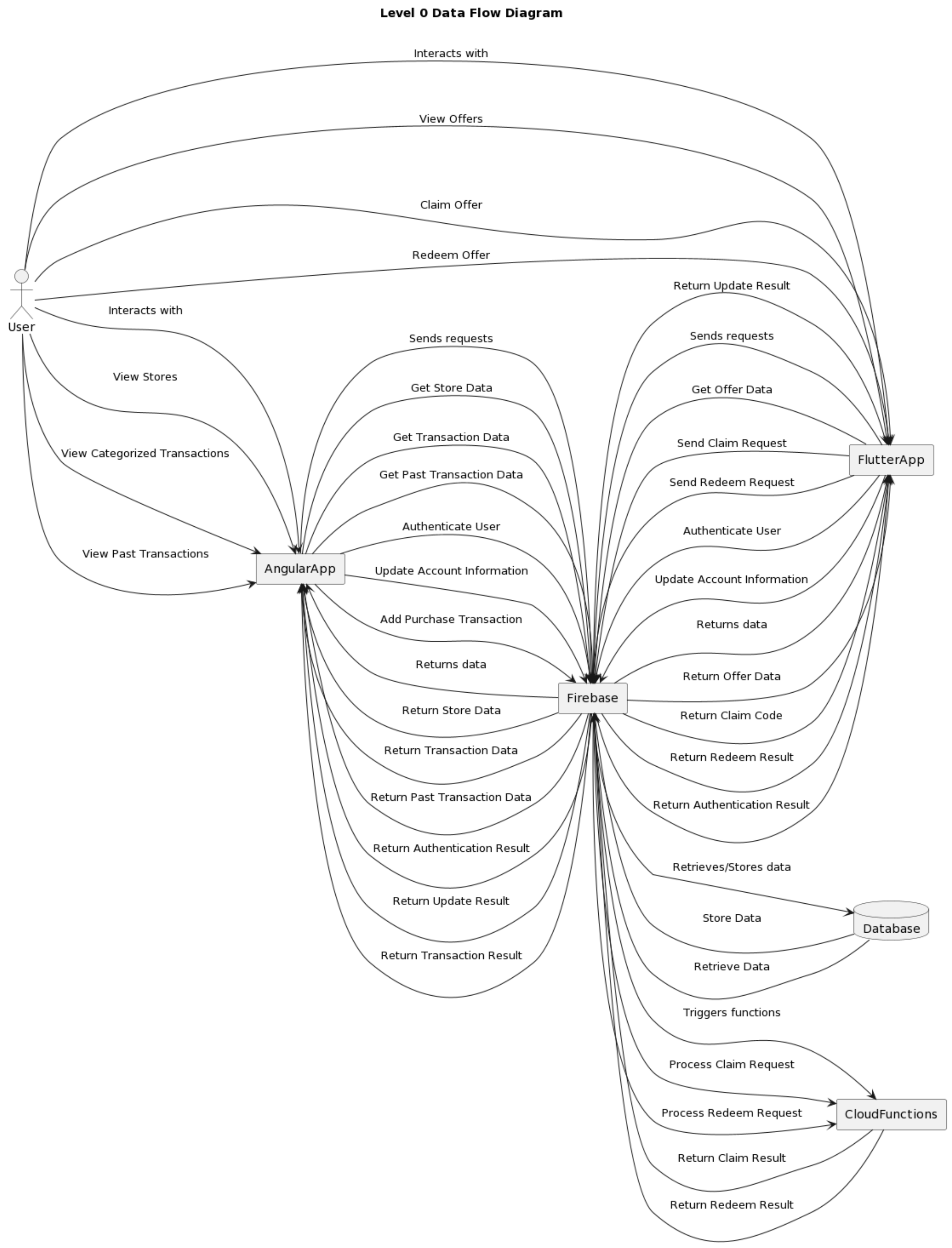


View Statistics about Spending Activity Diagram



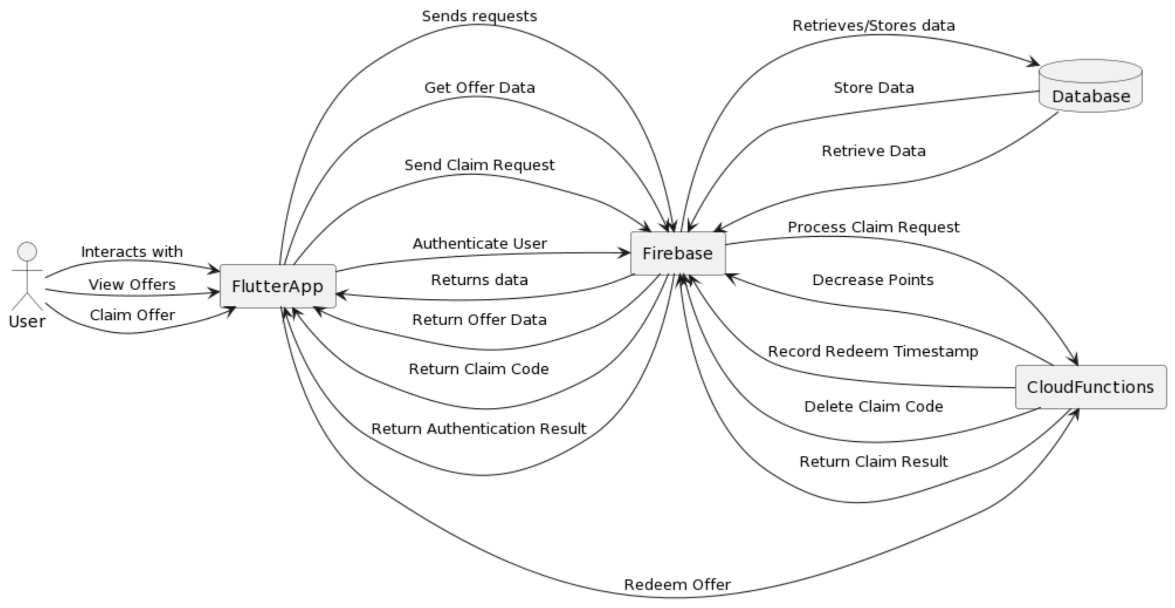
Interfaces – Data Flow Diagrams

Level 0 Data flow

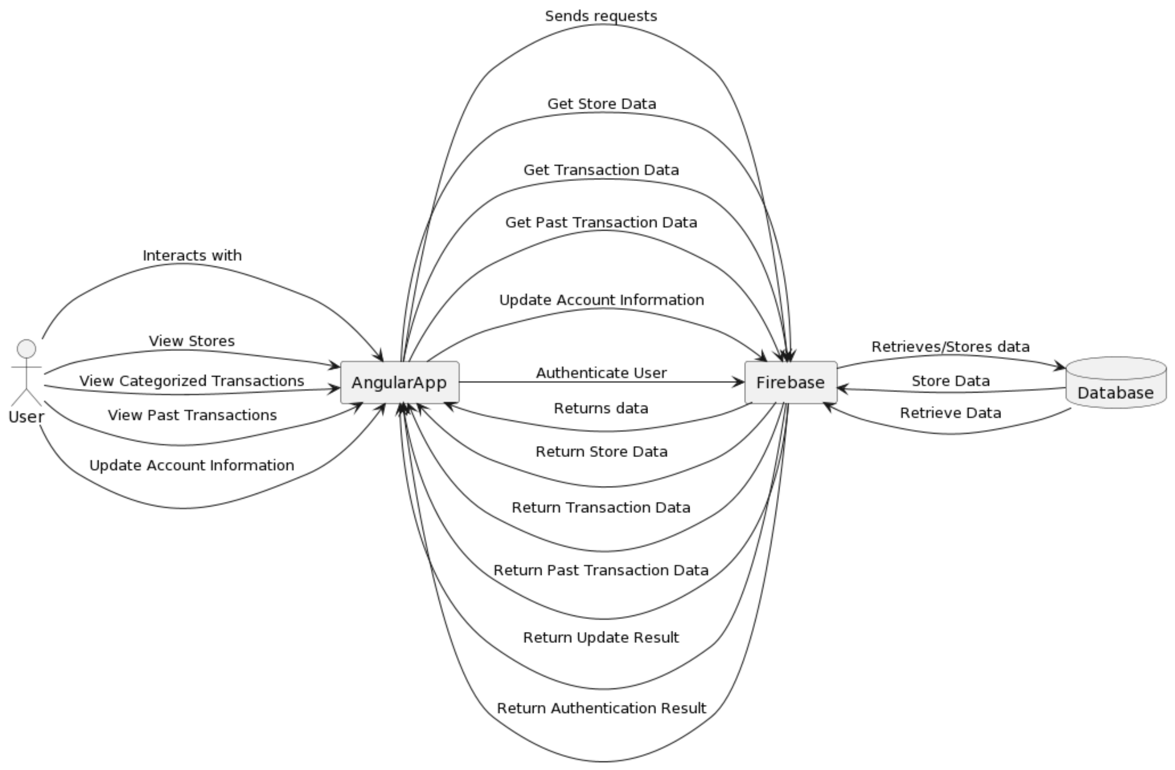


Level 1 Data flow

Level 1 Data Flow Diagram - FlutterApp



Level 1 Data Flow Diagram - AngularApp



Pseudo Code for non-Trivial Functions

```
Procedure: Extracting Amount, Date, Store from SMS message
INPUT: Message, Format -> Object that shows which format is applied to this message
extractFromMessage(message, format):
    list Data => []
    for object in format:
        while( there is no more line in message):
            if(next line in message contains object):
                Data.append(the extracted data from the line)
            End If
        End While
    End For
```

```
Add Offer:
plaintextCopy code
function addOffer(offerDetails):
    offer = createOfferObject(offerDetails)

    if offer is not null:
        firebase.createOffer(offer)
        showSuccessMessage("Offer added successfully!")
    else:
        showErrorMessage("Invalid offer details. Please check the inputs.")
```

```
Delete Offer:
plaintextCopy code
function deleteOffer(offerId):
    firebase.deleteOffer(offerId)
    showSuccessMessage("Offer deleted successfully!")
```

```
Edit Offer:
plaintextCopy code
function editOffer(offerId, updatedDetails):
    offer = firebase.getOffer(offerId)

    if offer is not null:
        updatedOffer = updateOfferObject(offer, updatedDetails)

        if updatedOffer is not null:
            firebase.updateOffer(updatedOffer)
            showSuccessMessage("Offer updated successfully!")
        else:
            showErrorMessage("Invalid offer details. Please check the inputs.")
    else:
        showErrorMessage("Offer not found. Please check the offer ID.")
```

View Statistics Dashboard:
plaintextCopy code
```
function viewStatisticsDashboard():
    statisticsData = firebase.getStatisticsData()

    if statisticsData is not null:
        displayStatistics(statisticsData)
    else:
        showErrorMessage("Failed to fetch statistics data. Please try again later.")
```

Redeem Offer:
plaintextCopy code
```
function redeemOffer(offerCode):
    claimResult = callCloudFunction("processRedeemRequest", offerCode)

    if claimResult == "success":
        showSuccessMessage("Offer redeemed successfully!")
    else if claimResult == "expired":
        showErrorMessage("Offer has expired.")
    else if claimResult == "invalid":
        showErrorMessage("Invalid offer code.")
    else:
        showErrorMessage("Failed to redeem offer. Please try again later.")
```

Change Password:
plaintextCopy code
```
function changePassword(newPassword):
    currentUser = getCurrentUser()

    if currentUser is not null:
        updatedUser = updatePassword(currentUser, newPassword)

        if updatedUser is not null:
            firebase.updateUser(updatedUser)
            showSuccessMessage("Password updated successfully!")
        else:
            showErrorMessage("Invalid password. Please check the new password.")
    else:
        showErrorMessage("User not authenticated. Please log in.")
```

Claim Offer:
plaintextCopy code
```
function claimOffer(offerId):
    offer = firebase.getOffer(offerId)

    if offer is not null:
        claimCode = callCloudFunction("processClaimRequest", offerId)

        if claimCode is not null:
            showClaimCode(claimCode)
        else:
            showErrorMessage("Failed to claim offer. Please try again later.")
    else:
        showErrorMessage("Offer not found. Please check the offer ID.")
```

View Stores:
plaintextCopy code

```
function viewStores():
    storeList = firebase.getAllStores()

    if storeList is not null:
        displayStores(storeList)
    else:
        showErrorMessage("Failed to fetch store data. Please try again later.")
```

View Categorized Transactions:
plaintextCopy code

```
function viewCategorizedTransactions(category):
    transactions = firebase.getTransactionsByCategory(category)

    if transactions is not null:
        displayTransactions(transactions)
    else:
        showErrorMessage("Failed to fetch transactions. Please try again later.")
```

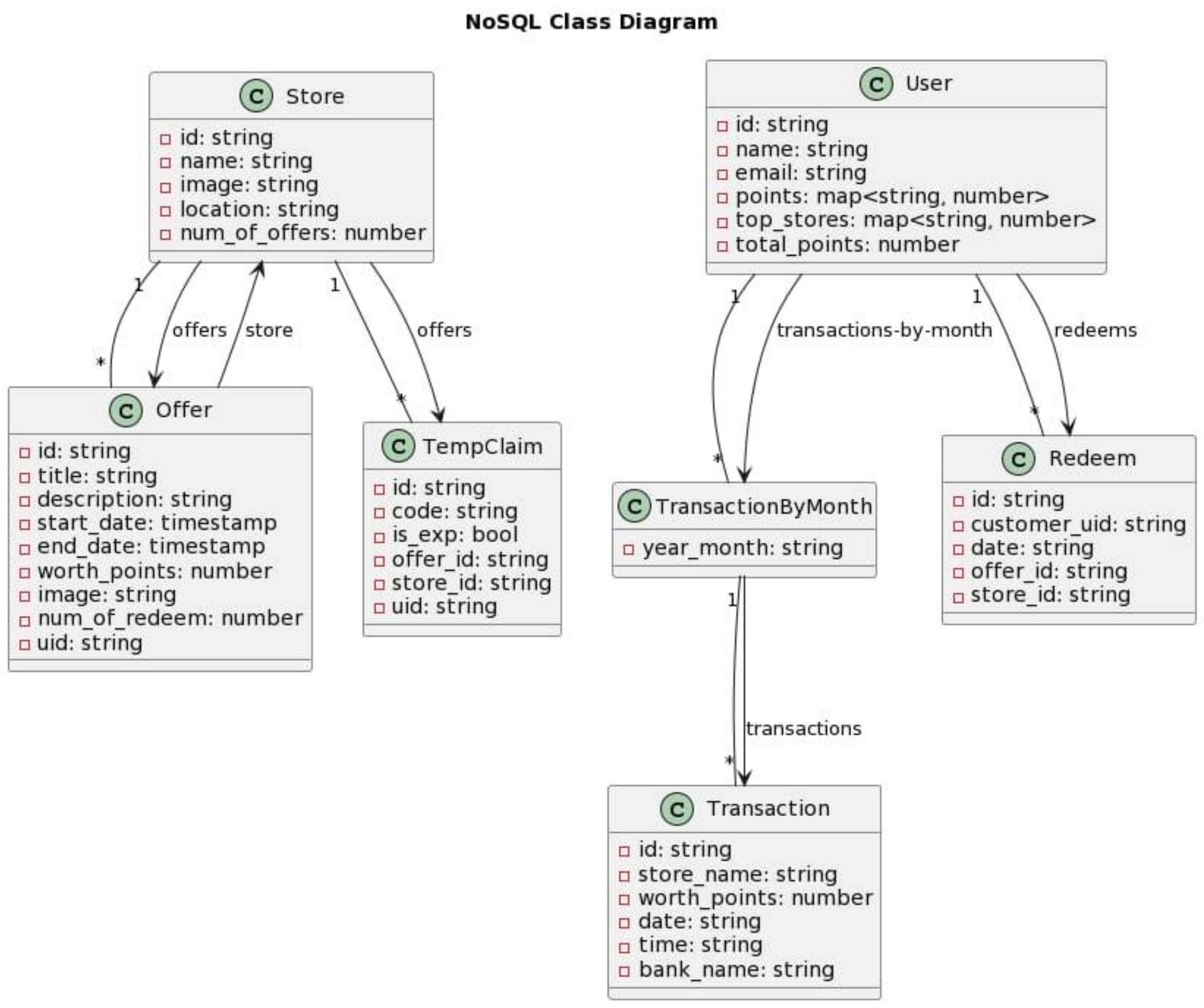
View Past Transactions:
plaintextCopy code

```
function viewPastTransactions():
    transactions = firebase.getPastTransactions()

    if transactions is not null:
        displayTransactions(transactions)
    else:
        showErrorMessage("Failed to fetch past transactions. Please try again later.")
```

Data Design

Database Description



Data Dictionary

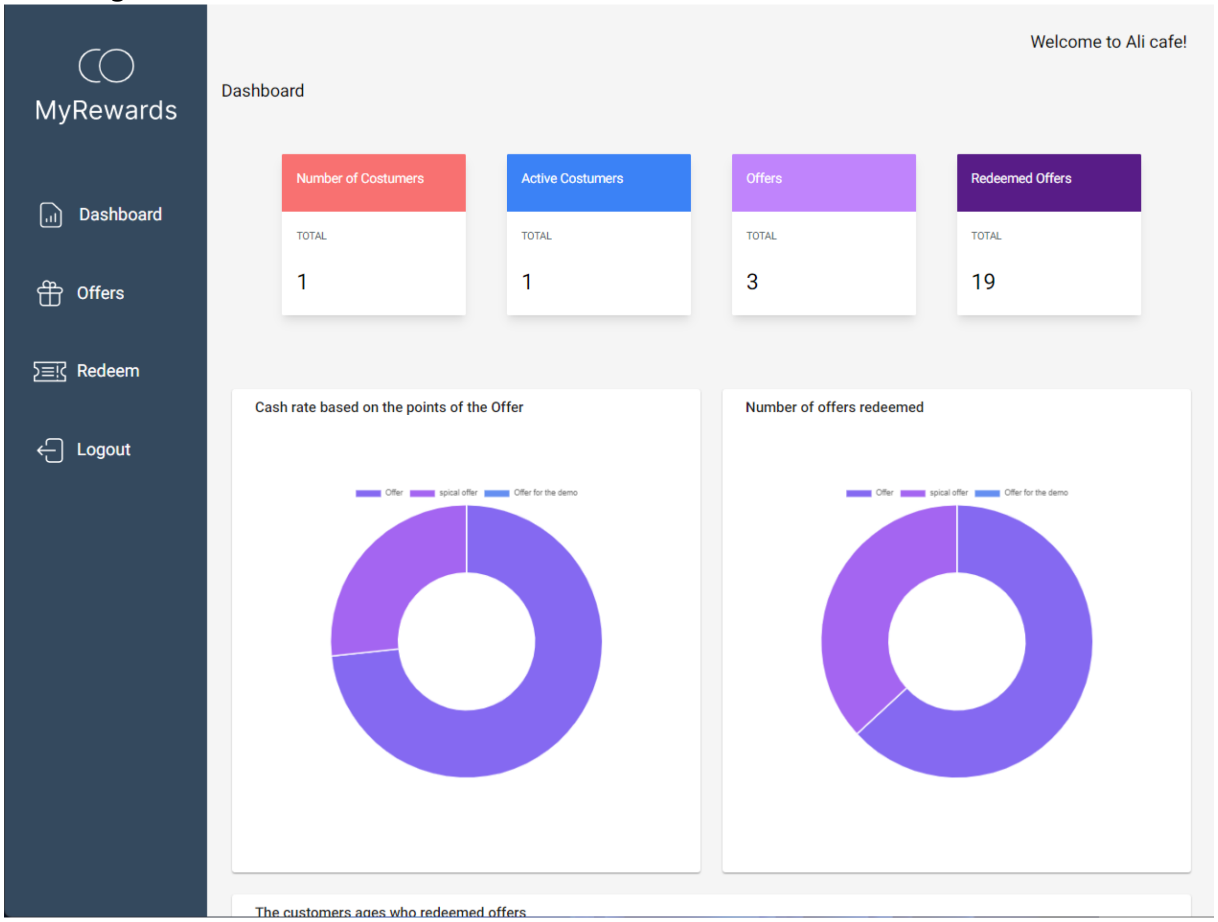
Entity	Description	Attribute	Data type	Description
User	A collection that holds user information. The user can be both mobile or web application user	id	string	
		name	string	
		email	string	
		Points	<string, number>	
		Total_points	<string, number>	
Offer	A collection that hold offer information	id	String	
		title	String	
		Description	String	
		Start_date	timestamp	
		End_date	timestamp	
		Worth_points	number	
		image	String	
		Num_of_redeem	number	
		Uid	String	
Temp-Claim	A collection that is used to store the temporarily generated offer code information	id	string	
		code	string	
		Is_exp	boolean	
		Offer_id	string	
		Store_id	String	
		Uid	string	
Transaction	A collection that stores the transactions infromation	id	string	
		Store_name	string	
		Worth_points	number	
		date	string	
		time	string	
		Bank_name	string	
Store	A collection that holds store information	id	string	
		name	string	
		image	string	
		Location	string	
		Num_of_offers	Number	
Redeem	A collection that stores information about redeemed offers	id	string	
		Customer_uid	String	
		date	String	
		Offer_id	string	
		Store_id	String	
TransactionsByMonth		Year_month		

Human Interface Design

Screen Images

Business owner web portal

Main Page



Welcome to Ali cafe!

Dashboard

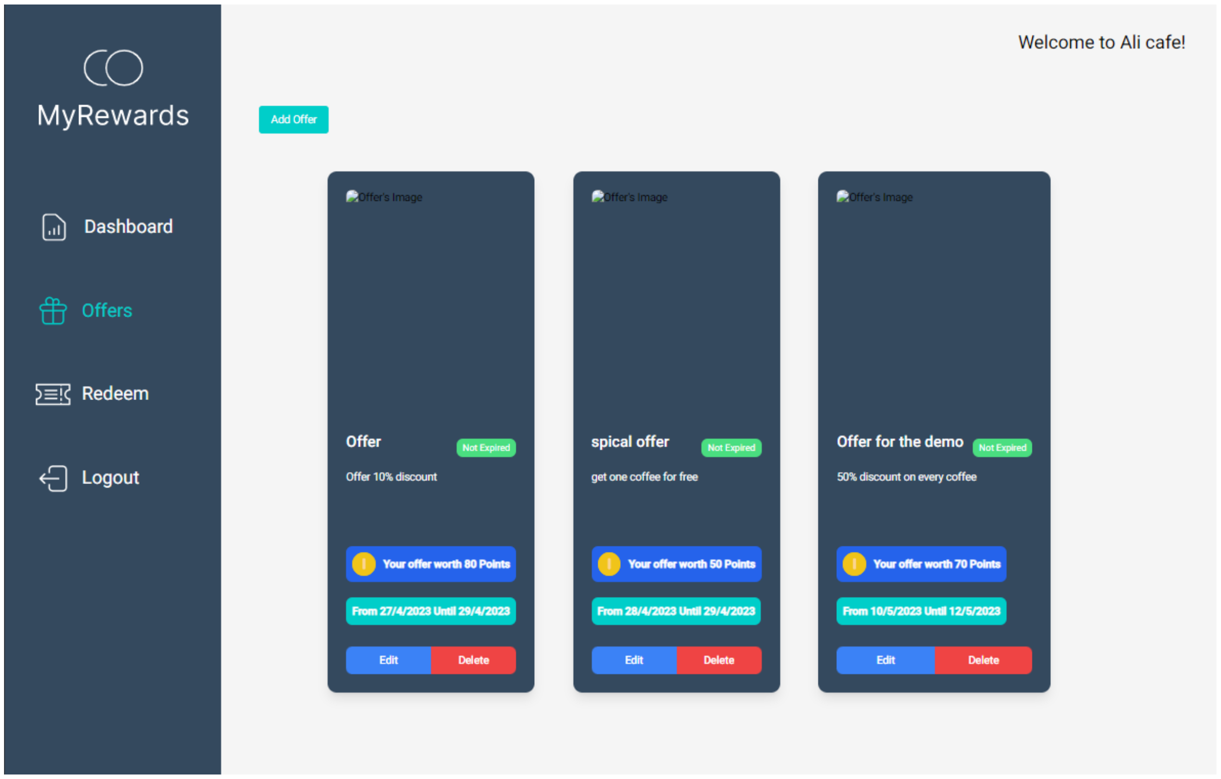
Number of Costumers	Active Costumers	Offers	Redeemed Offers
TOTAL	TOTAL	TOTAL	TOTAL
1	1	3	19

Cash rate based on the points of the Offer

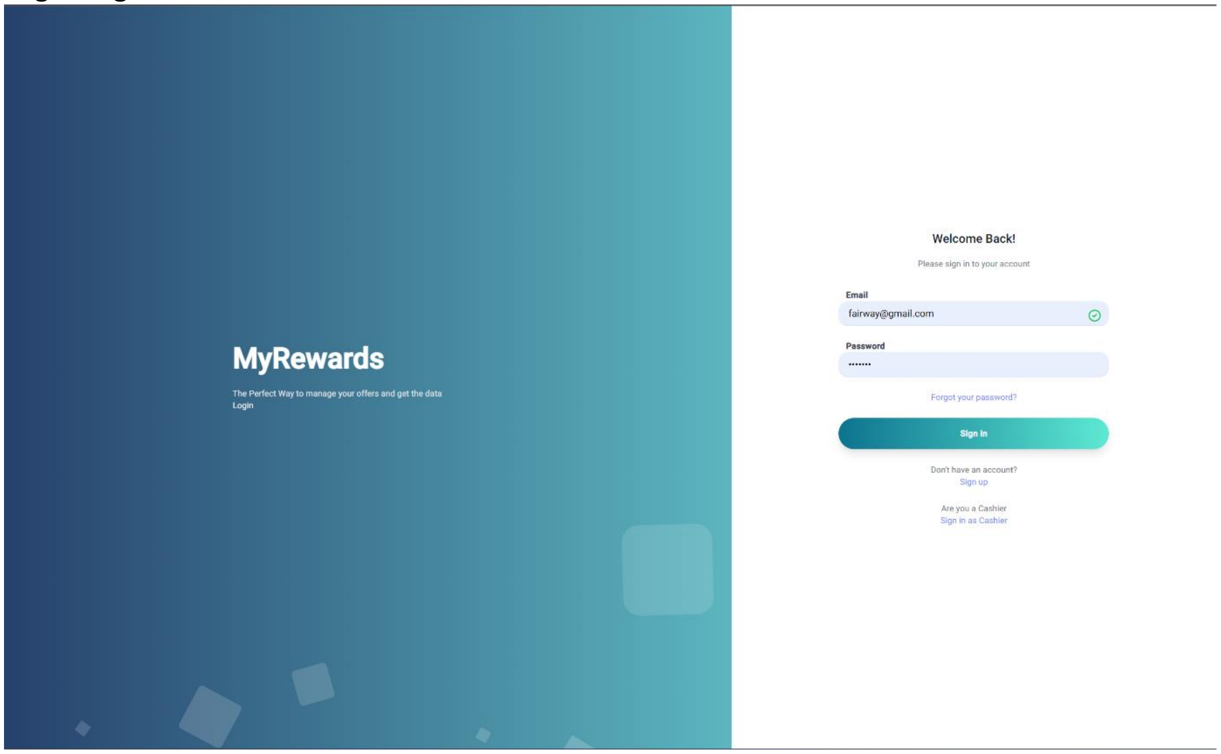
Number of offers redeemed

The customers ages who redeemed offers

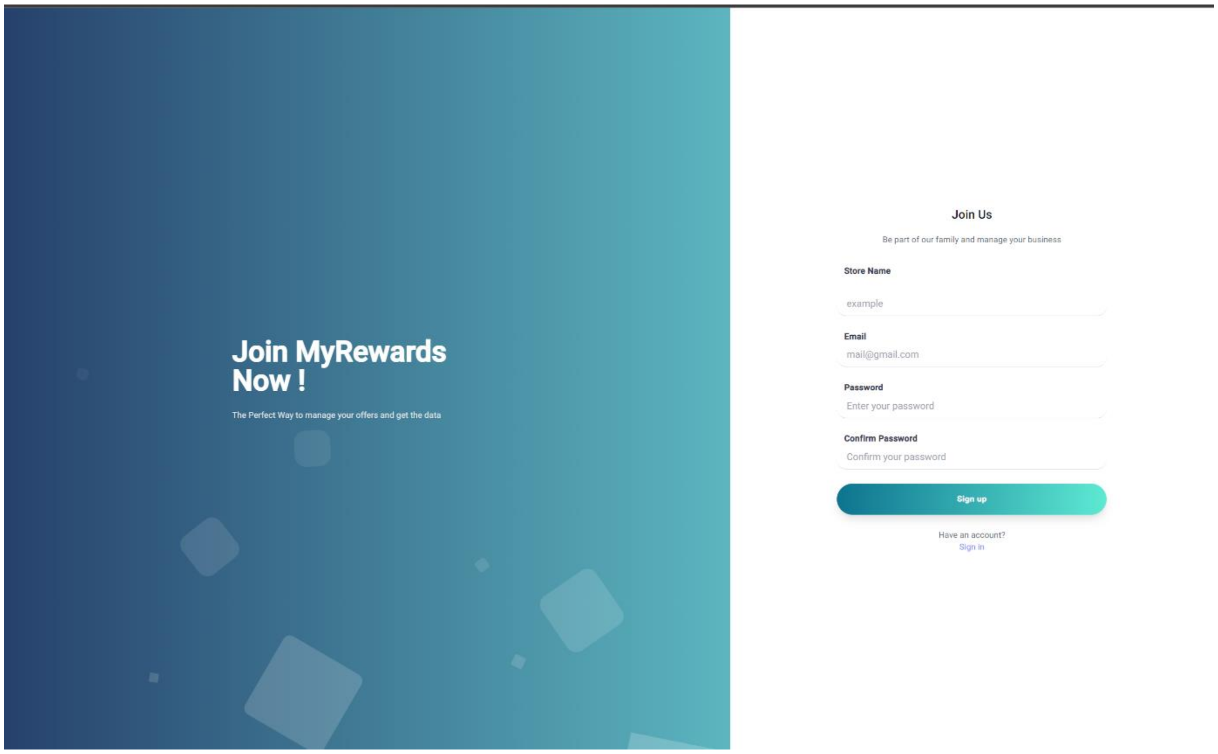
Offers Page



Login Page



Registration Page


Join MyRewards Now !

The Perfect Way to manage your offers and get the data

Join Us

Be part of our family and manage your business

Store Name: example

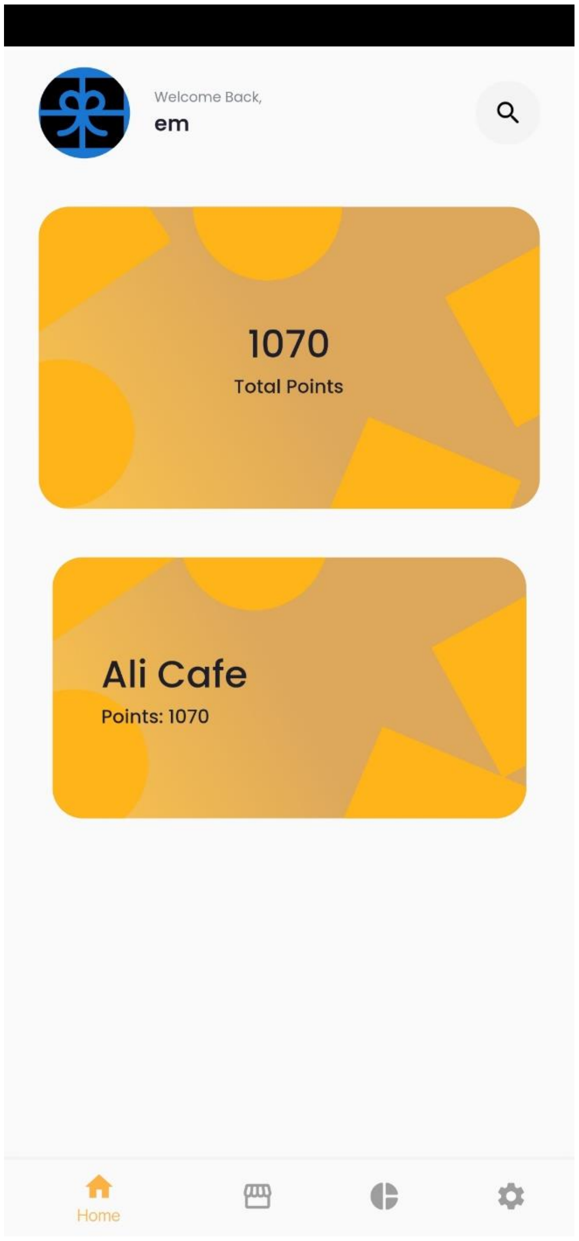
Email: mail@gmail.com

Password: Enter your password

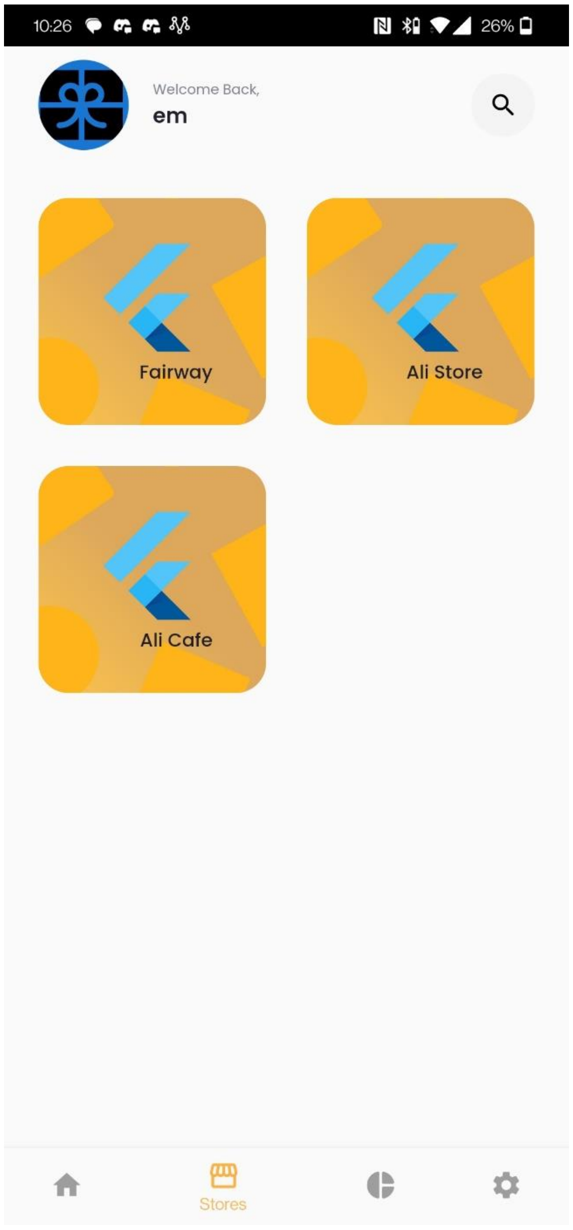
Confirm Password: Confirm your password

Customer mobile application

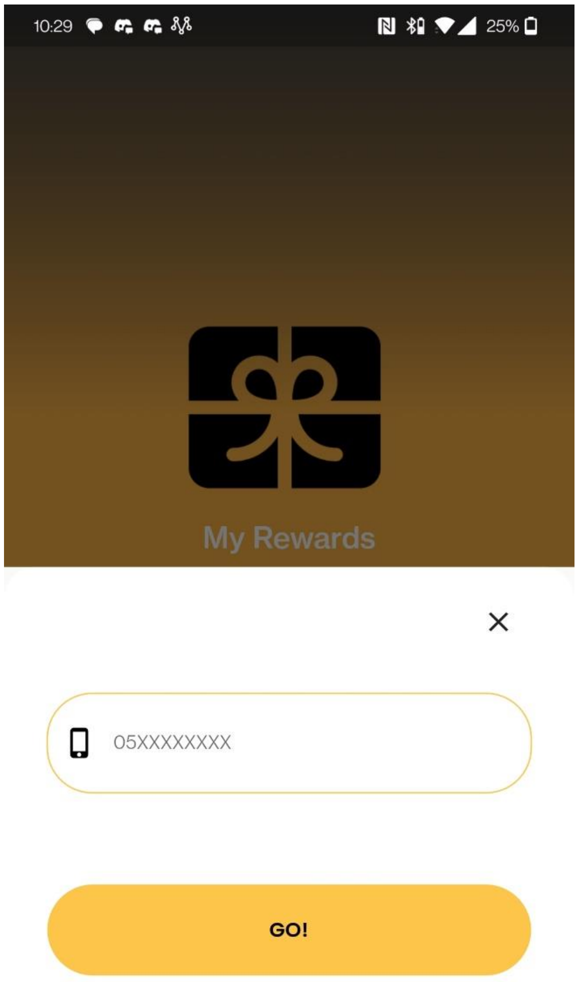
Main screen

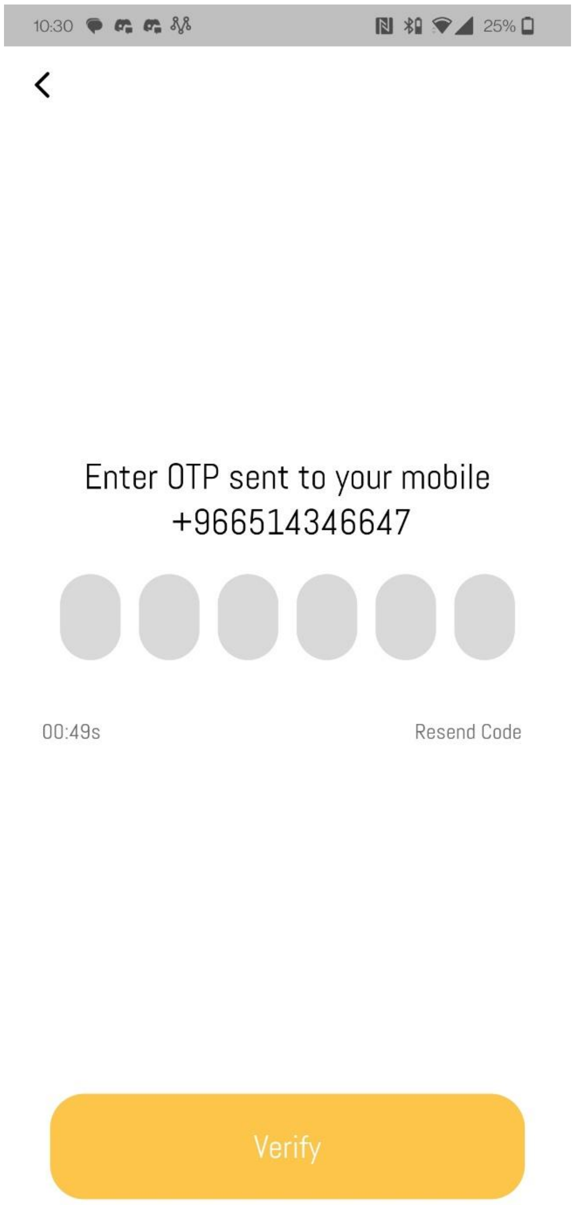


Stores screen



Login/Signup screen





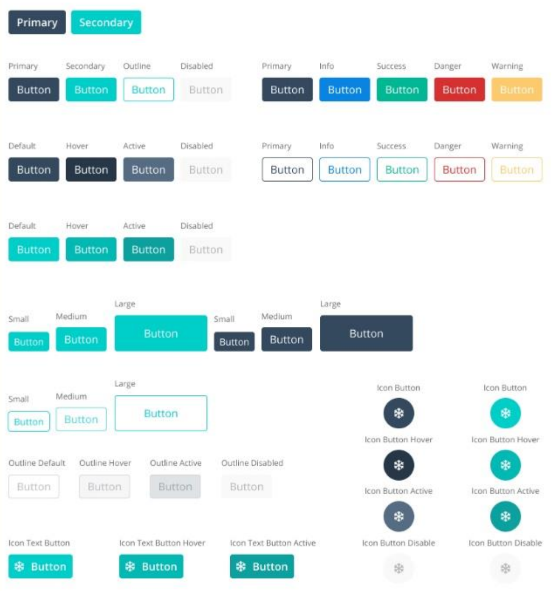
Report Format

We have decided to follow this format for the identity for our application

Buttons

Colors

Buttons



Theme Colors

primary	Secondary	Tertiary
#34495E	#00cec9	#00cec9

State Colors

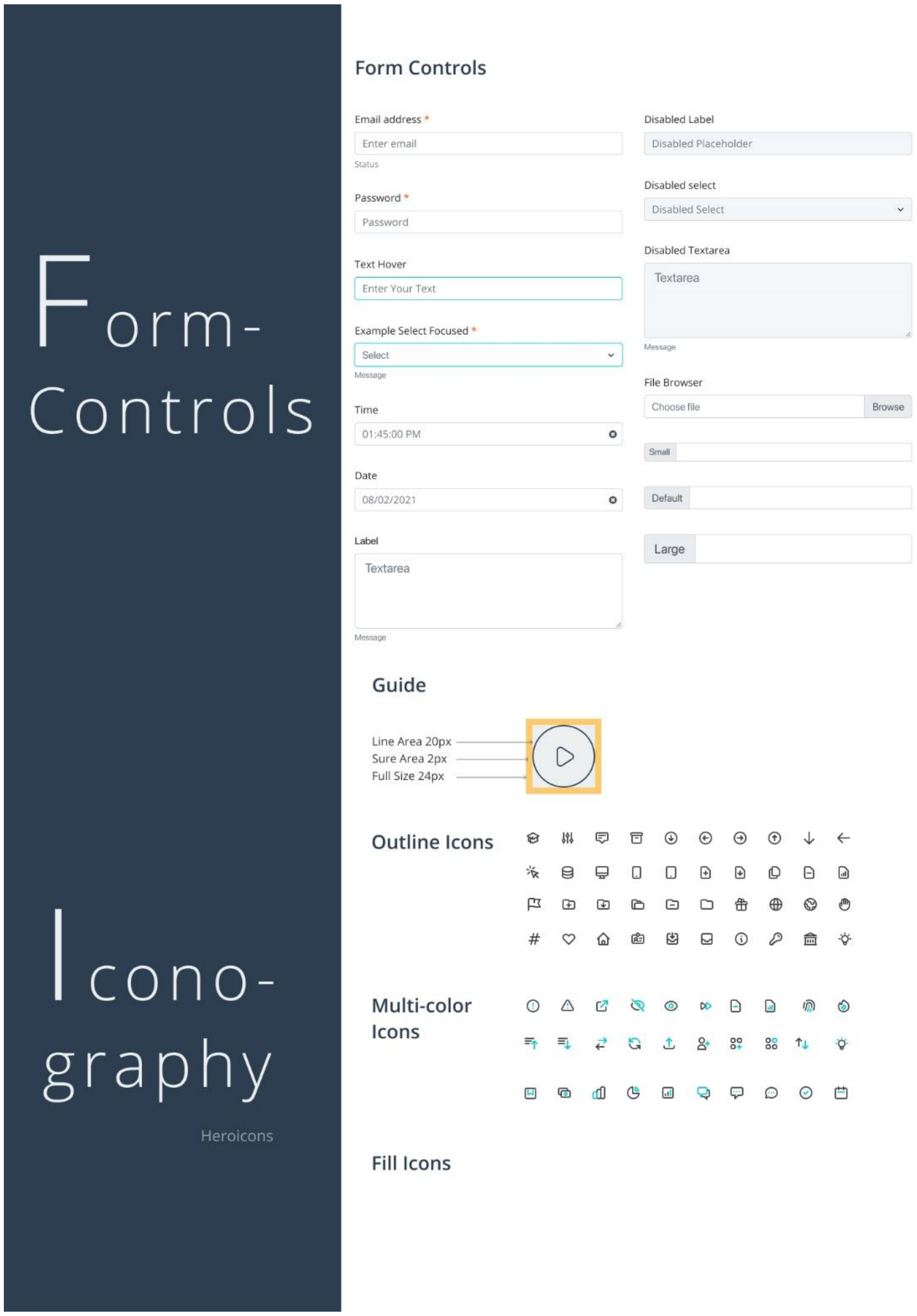
Info	Success	Danger	Warning
#0984E3	#00B894	#D63031	#FDCB6E

Black Colors

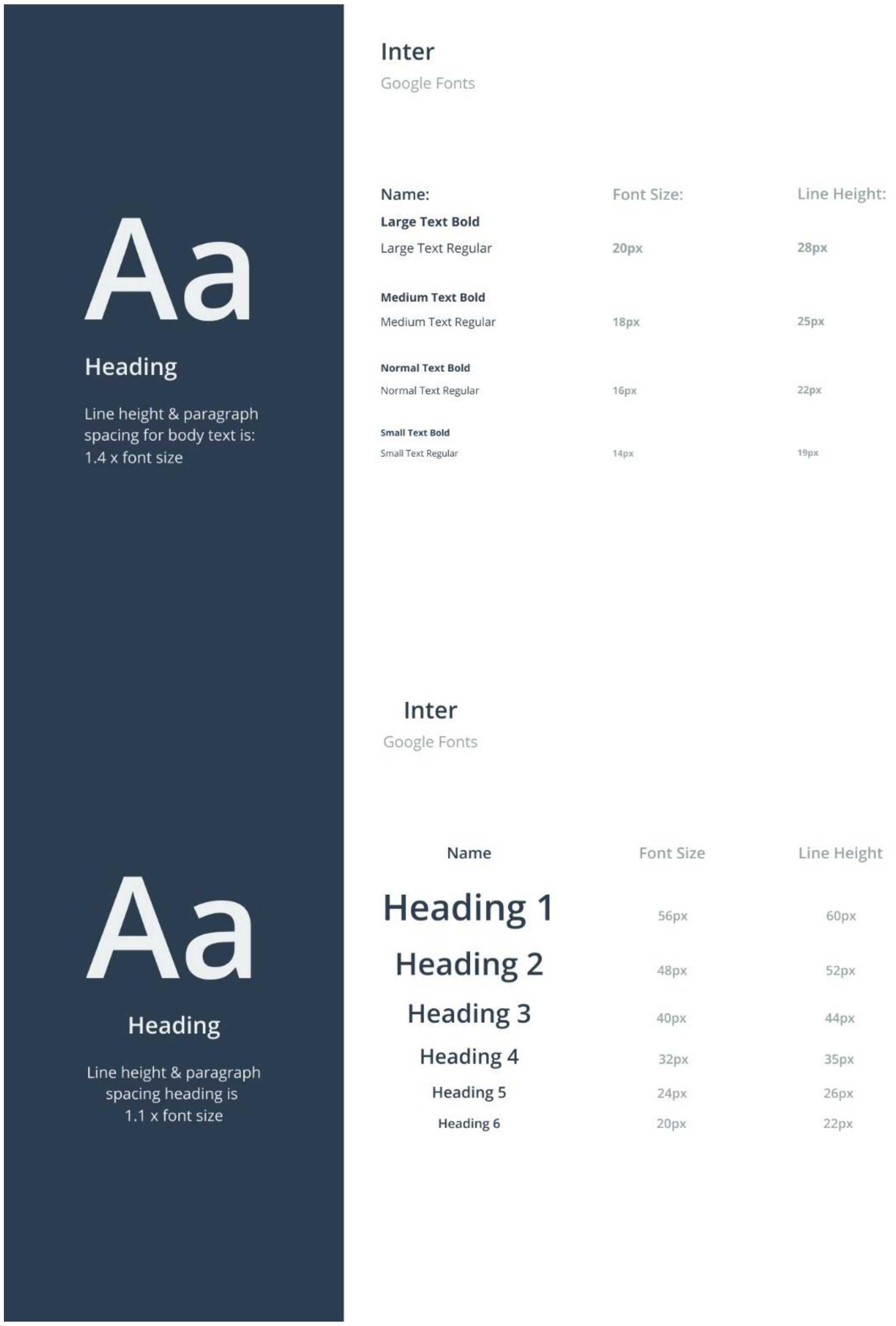
Dracula Orchid-1	Dracula Orchid-2	Dracula Orchid-3	Dracula Orchid-4
#000000	#171919	#25292A	#2D3436

Gray Colors

American River-1	American River-2	American River-3	Soothing Breeze
#636E72	#737D81	#9CA7AC	#B2BEC3



This is our font and its style:


Aa

Heading

Line height & paragraph spacing for body text is: 1.4 x font size

Inter
Google Fonts

Name:	Font Size:	Line Height:
Large Text Bold		
Large Text Regular	20px	28px
Medium Text Bold		
Medium Text Regular	18px	25px
Normal Text Bold		
Normal Text Regular	16px	22px
Small Text Bold		
Small Text Regular	14px	19px

Aa

Heading

Line height & paragraph spacing heading is 1.1 x font size

Inter
Google Fonts

Name	Font Size	Line Height
Heading 1	56px	60px
Heading 2	48px	52px
Heading 3	40px	44px
Heading 4	32px	35px
Heading 5	24px	26px
Heading 6	20px	22px

Requirements Matrix

Jira Requirement ID	Jira Associated Requir	Requirement Name	User Story	Assigned To	Test Case	Current Status

	ement ID					
26	26.1	Add new offer	As a business owner I want to add a new offer so that my customers can benefit from.	Mohammed Almohammedsaleh	None	Development
26	26.2	Validate offer claim	As a Business owner I want to validate the offer that is claimed by the customer so that I can give him the offer		None	Not Started
26	26.24	Delete existing offer	As a Business owner I want to validate offer claim so that I can give the customer the claimed offer		None	Not Started
26	26.3	Modify existing offer	As a Business owner I want to modify an existing offer		None	Not Started

26	26.4	Subscribe to offer	As a user I want to subscribe to an offer so that benefit from the offer	Mohammed Almohammedsaleh	None	Not Started
26	26.5	View progress offer	As I user I want to view my progress to gain an offer so that I can redeem it at the store		None	Not Started
26	26.6	Claim offer	As a user I want to get the offer code so that I can use it in the store		None	Not Started
28	28.8	Review a Store	As a user I want to review the stores that I have visited so that I can get points		None	Not Started
29	29.1	Use friend invitation	As a user I want to use an existing invite so that my inviter and I can gain more points		None	Not Started
29	29.22	BO Login	As a System User I	Ali khalil	None	Development

			want to login to my account			
29	29.34	Register	As a System User I want to register my account so that I can benefit from the system features	ALI ALBANNAY	None	Development
29	29.68	BO, Sub-Accounts	As a business owner I want to create sub-accounts for my store, so that my employees can validated offers and track the store statistics	Ali khalil	None	Elaboration
29	29.9	Recommend store to friend	As a user I want to invite other users to buy from a store so that I can gain more loyalty points		None	Not Started

30	30.11	Post Inquiry	As a user I want to post an inquiry about a store/service so that I get a feedback from other customers		None	Not Started
30	30.12	Help resolving inquiry	As a user, I want to give feedback, so I can get points and help others with my experiences.		None	Not Started
30	30.18	Vote for the best inquiry help	As a User I want to vote for the most helpful answer about my inquiry so that he and I can gain more points		None	Not Started
31	31.14	Add manual purchase information (cash payment)	As a BO I want to add a manual purchase information so that my customer can gain points if they prefer to		None	Not Started

			pay me with cash			
31	31.7	Detect & Add purchase transactio n	As a user I want my purchase to be added automatic ally so that I get points		None	Not Started
32	32.13	View spending history	As a user, I want to see my spending based on the day, month, or a year.		None	Not Started
32	32.15	Add manual transactio n	As a User I want to add a purchase details manully so that I can track my spending		None	Not Started
32	32.16	View Statistical Informatio n	As a Business Owner I want to view statistical represent ations about each of my offers, so that I can make business decisions		None	Not Started

			regarding that offer			
32	32.19	View Wallet	As a User I want to view my wallet credit		None	Not Started
32	32.2	Redeem Coins	As a user I want to exchange my gained coins with a merchandise		None	Not Started

Resource Estimates

For BO:

A computer is required at least with these specifications

- 4 GB RAM.
- 1.8 Hz CPU.

For the customer:

A phone is required at least with these specifications:

- 2 GB RAM.
- Android 10 or more.

Definitions

#	Word	Definition
1	BO	Business Owner
2	IOS	Iphone Operating System