SUBSCRIPTION MANAGEMENT AND TRACKING APP (MILESTONE 1)

Team: Abdulwahab Alnemer, Yosef Alhedr

Faculty Advisor: Khaled Slhoub

Software Requirements Specification

1. Introduction

The Subscription Management and Tracking App is designed to simplify and streamline the management of user subscriptions. This document outlines the requirements for the app, including expected features, sample inputs and outputs for correct and incorrect behavior, and technical challenges.

2. General Requirements

- The app must support multiple platforms, including Web, iOS, and Android.
- The app should be developed using relevant technologies such as React.js.
- The app must ensure data consistency and integrity by designing an efficient database schema.
- The app must provide a user-friendly interface for ease of use.

3. User Interface Requirements

- The app's user interface should be intuitive and easy to navigate.
- The user interface must be responsive and compatible with various devices and screen sizes.

4. Subscription Dashboard

4.1 Comprehensive Subscription Dashboard

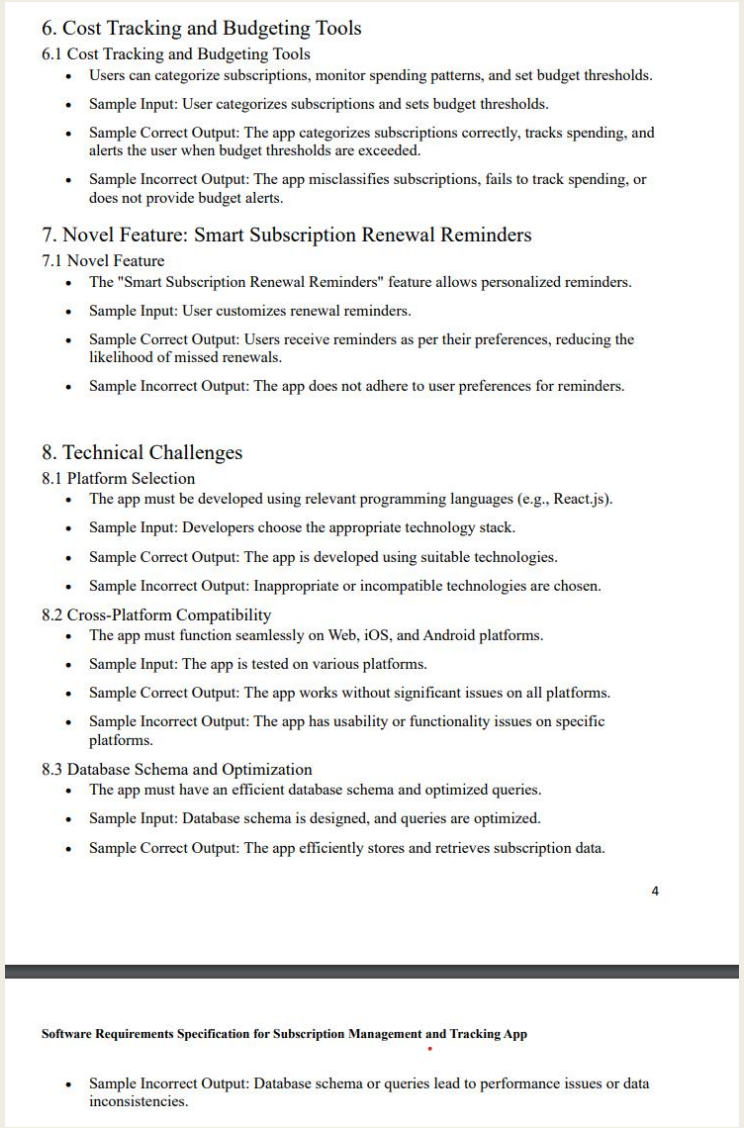
- The app must provide users with a centralized dashboard displaying all subscriptions.
- Sample Input: User logs in and views the dashboard.
- Sample Correct Output: The dashboard displays a list of active subscriptions, upcoming billing dates, and associated costs.
- Sample Incorrect Output: The dashboard is empty or displays inaccurate subscription information.

5. Smart Subscription Renewal Reminders

5.1 Smart Subscription Renewal Reminders

- Users can customize how they receive subscription renewal reminders.
- Sample Input: User sets preferences for renewal reminders.
- Sample Correct Output: User receives reminders through preferred channels (push notifications, emails, in-app alerts).
- Sample Incorrect Output: User does not receive reminders or receives reminders through the wrong channels.

Software Requirements Specification Cont.



6. Cost Tracking and Budgeting Tools

6.1 Cost Tracking and Budgeting Tools

- Users can categorize subscriptions, monitor spending patterns, and set budget thresholds.
- Sample Input: User categorizes subscriptions and sets budget thresholds.
- Sample Correct Output: The app categorizes subscriptions correctly, tracks spending, and alerts the user when budget thresholds are exceeded.
- Sample Incorrect Output: The app misclassifies subscriptions, fails to track spending, or does not provide budget alerts.

7. Novel Feature: Smart Subscription Renewal Reminders

7.1 Novel Feature

- The "Smart Subscription Renewal Reminders" feature allows personalized reminders.
- Sample Input: User customizes renewal reminders.
- Sample Correct Output: Users receive reminders as per their preferences, reducing the likelihood of missed renewals.
- Sample Incorrect Output: The app does not adhere to user preferences for reminders.

8. Technical Challenges

8.1 Platform Selection

- The app must be developed using relevant programming languages (e.g., React.js).
- Sample Input: Developers choose the appropriate technology stack.
- Sample Correct Output: The app is developed using suitable technologies.
- Sample Incorrect Output: Inappropriate or incompatible technologies are chosen.

8.2 Cross-Platform Compatibility

- The app must function seamlessly on Web, iOS, and Android platforms.
- Sample Input: The app is tested on various platforms.
- Sample Correct Output: The app works without significant issues on all platforms.
- Sample Incorrect Output: The app has usability or functionality issues on specific platforms.

8.3 Database Schema and Optimization

- The app must have an efficient database schema and optimized queries.
- Sample Input: Database schema is designed, and queries are optimized.
- Sample Correct Output: The app efficiently stores and retrieves subscription data.

- Sample Incorrect Output: Database schema or queries lead to performance issues or data inconsistencies.

Software Requirements Specification Cont.

9. Security Requirements

9.1 User Authentication

- The app must implement secure user authentication mechanisms, including password hashing and protection against common security vulnerabilities like SQL injection and cross-site scripting (XSS).
- Sample Input: User creates an account or logs in.
- Sample Correct Output: User authentication is secure and resistant to common attacks.
- Sample Incorrect Output: Authentication is vulnerable to attacks, leading to unauthorized access.

9.2 Data Encryption

- Sensitive user data, such as login credentials and financial information, must be encrypted during transmission and storage.
- Sample Input: User submits sensitive information.
- Sample Correct Output: Sensitive data is encrypted and securely stored.
- Sample Incorrect Output: Data is transmitted or stored without encryption, posing a security risk.

10. User Account Management

10.1 Profile Management

- Users should be able to update their profiles, including personal information and contact details.
- Sample Input: User edits their profile information.
- Sample Correct Output: User profile is updated as per the input.
- Sample Incorrect Output: Profile updates are not reflected, or errors occur during the update process.

10.2 Subscription History

- The app should maintain a detailed history of subscription changes, including cancellations and modifications.
- Sample Input: User cancels or modifies a subscription.
- Sample Correct Output: A record of the change is stored in the user's history.
- Sample Incorrect Output: Subscription history is not accurately recorded or is missing information.

Software Requirements Specification Cont.

11. Localization Requirements

11.1 Multilingual Support

- The app should support multiple languages to accommodate a diverse user base.
- Sample Input: User selects a preferred language.
- Sample Correct Output: The app displays content in the selected language.
- Sample Incorrect Output: Language preferences are not honored, or translations are inaccurate.

12. Performance Requirements

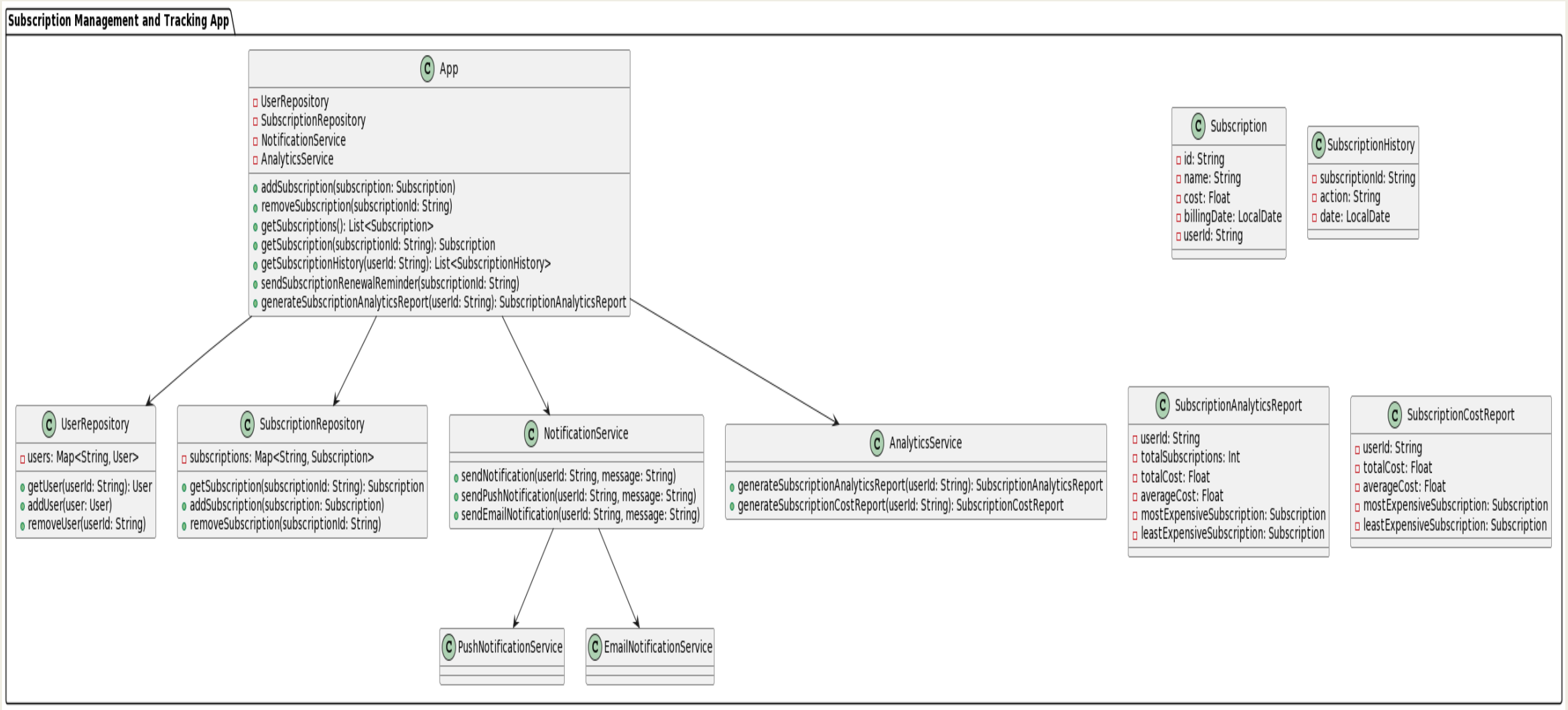
12.1 Response Time

- The app must provide a responsive user experience with minimal loading times and quick responses to user actions.
- Sample Input: User interacts with the app.
- Sample Correct Output: The app responds promptly to user actions.
- Sample Incorrect Output: Slow response times or delays hinder the user experience.

12.2 Scalability

- The app must be designed to handle a growing user base and increasing data without significant performance degradation.
- Sample Input: The app experiences increased user activity.
- Sample Correct Output: The app scales to accommodate the increased load without performance issues.
- Sample Incorrect Output: Increased load leads to slowdowns or crashes.

Software Design – Architecture



Software Design – Some of The Classes

2.1 Classes and Methods

App class: This is the main entry point for the application. It interacts with the UserRepository, SubscriptionRepository, NotificationService, and AnalyticsService classes to perform tasks such as adding and removing subscriptions, sending renewal reminders, and generating analytics reports.

- **Methods:**
- addSubscription(subscription: Subscription): This method adds a new subscription to the system.
- removeSubscription(subscriptionId: String): This method removes an existing subscription from the system.
- getSubscriptions(): List<Subscription>: This method returns a list of all subscriptions for the current user.
- getSubscription(subscriptionId: String): Subscription: This method returns the subscription with the specified ID.
- getSubscriptionHistory(userId: String): List<SubscriptionHistory>: This method returns a list of all subscription history records for the current user.
- sendSubscriptionRenewalReminder(subscriptionId: String): This method sends a renewal reminder notification to the user for the subscription with the specified ID.
- generateSubscriptionAnalyticsReport(userId: String): SubscriptionAnalyticsReport: This method generates a subscription analytics report for the current user.

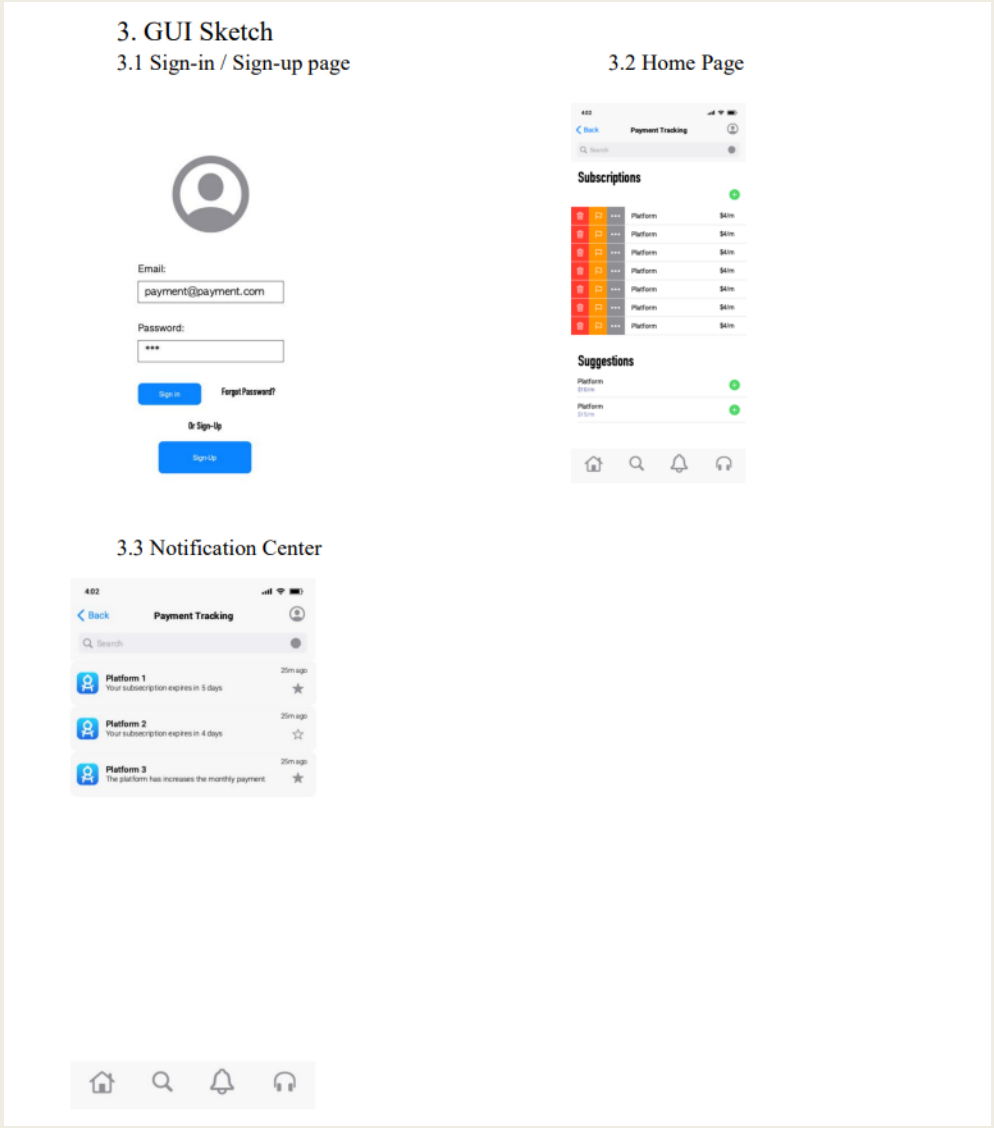
UserRepository class: This class stores information about users, such as their name, email address, and subscription history.

- **Methods:**
- getUser(userId: String): User: This method returns the user with the specified ID.
- addUser(user: User): This method adds a new user to the system.
- removeUser(userId: String): This method removes an existing user from the system.

SubscriptionRepository class: This class stores information about subscriptions, such as their name, cost, and billing date.

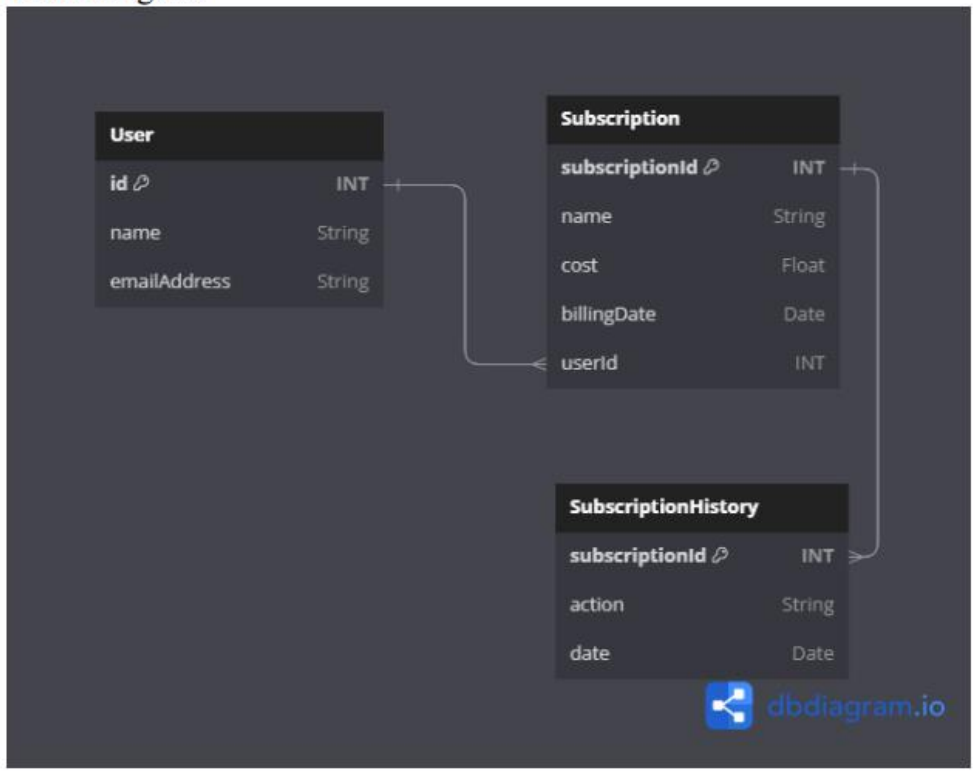
- **Methods:**
- getSubscription(subscriptionId: String): Subscription: This method returns the subscription with the specified ID.
- addSubscription(subscription: Subscription): This method adds a new subscription to the system.
- removeSubscription(subscriptionId: String): This method removes an existing subscription from the system.

Software Design – GUI Sketch



Software Design – ER Diagram

4. ER Diagram



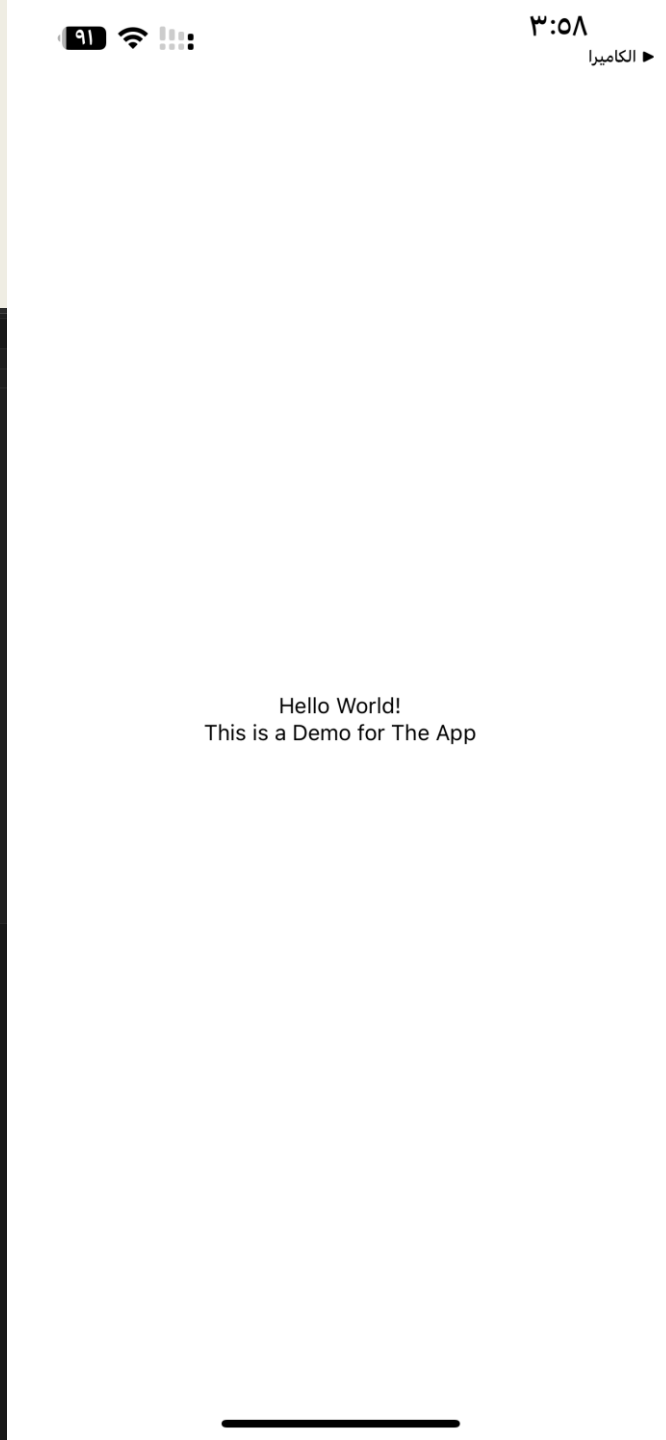
4.1 Tables

User Table: This table stores information about users, such as their ID, name, and email address.

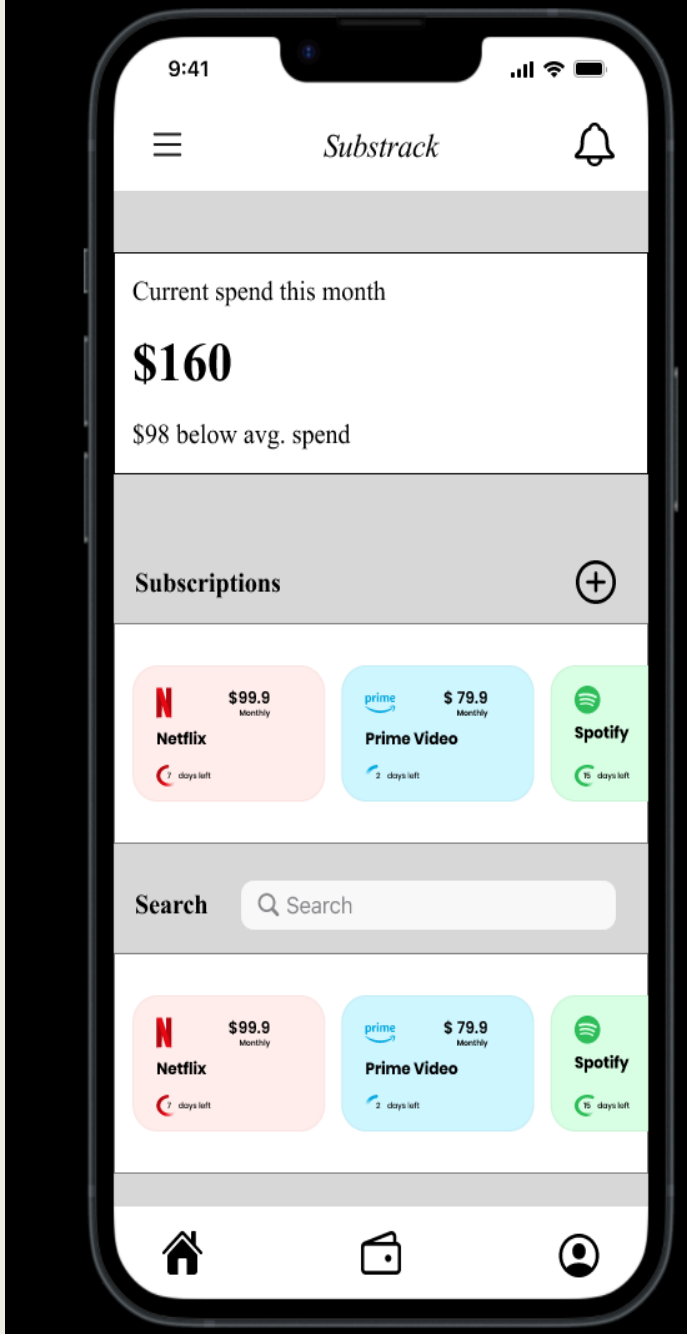
Subscription Table: This table stores information about subscriptions, such as their ID, name, cost, billing date, and user ID.

SubscriptionHistory Table: This table stores information about subscription history records, such as the ID of the subscription, the action that was performed, and the date of the action.

Hello World!
Demo



Basic UI


9:41
Substrack
Current spend this month
$160
$98 below avg. spend
Subscriptions
$99.9
Monthly
Netflix
7 days left
prime
$ 79.9
Monthly
Prime Video
2 days left
Spotify
15 days left
Search
Search
$99.9
Monthly
Netflix
7 days left
prime
$ 79.9
Monthly
Prime Video
2 days left
Spotify
15 days left

Next Milestone 2 (Oct 30)

- Finalize the app's user interface design.
- Create a clickable prototype of the app's key screens for user testing and validation.
- Implement user registration and login functionality.
- Develop the central dashboard to display a list of subscriptions (static data).
- Set up a basic database structure to store subscription data.
- Begin working on the notification system for reminders (basic functionality).