PillPoppin

BY :

Ghaidaa azzam

Yomna zoabi

1. Introduction

1.1 Purpose

The purpose of **PillPopin** is to provide an efficient medication management system designed for both doctors and patients. This app ensures patients adhere to their prescribed medication schedules by offering reminders, tracking medicine intake, and managing prescriptions. For doctors, the system simplifies the process of managing patients' prescriptions and tracking patient compliance with their medications.

1.2 Scope

The scope of **PillPopin** encompasses a medication management system aimed at both doctors and patients. The system allows doctors to add and manage patient prescriptions, while patients can view their prescriptions, track their medicine intake, receive reminders for medication times, and adjust intake schedules.

The system automates key tasks like sending medication reminders and tracking medicine intake. By providing a user-friendly interface for both patients and doctors, **PillPopin** improves the efficiency of prescription management and medication adherence. The app helps patients manage their treatments independently by providing easy access to prescription information, reminders, and medicine details such as warnings and side effects.

1.3 Overview*

This Software Requirements Specification (SRS) document defines the functional and non-functional requirements for the **PillPopin** application, a mobile app designed for both doctors and patients to manage prescriptions and medication intake efficiently. The document covers the app's purpose, its architecture, key functionalities, and technical details.

Section 1 introduces the purpose and scope of **PillPopin**, explaining its role in helping doctors manage prescriptions and allowing patients to track their medications.

Section 2 provides an overview of the system's structure, describing how doctors, patients, and the system interact.

Section 3 outlines the specific functionalities available to doctors, patients, and the system, including prescription management, medicine intake tracking, profile updates, and notifications.

Section 4 details the non-functional requirements, including performance, security, usability, reliability, scalability, and portability.

Section 5 describes the system architecture, including the client-side (frontend) using Flutter, backend services using Firebase, and real-time data synchronization via Firestore.

Section 6 includes diagrams and references to support the technical understanding of the system, along with a glossary of key terms.

This document is aimed at providing a comprehensive understanding of the PillPopin application, its intended use, and the features that support effective medication management.

2. System Overview

The **PillPopin** system is designed to streamline the process of medication management for both doctors and patients. The system is composed of three main actors: the Doctor, the Patient, and the System itself, which automates notifications and tracks medication intake.

- Doctors are responsible for adding and managing patients, prescribing medications, and assigning intake schedules. The system provides doctors with tools to view prescription details.

- Patients use the system to track their medication intake and receive reminders about upcoming doses. The system allows patients to view details of their prescriptions, including dosage instructions and warnings. Patients can also adjust their intake times as needed and update their personal profiles through the app.

- System Automation: The system sends automated notifications to remind patients to take their medication at the correct times. It tracks the intake of medications based on the patient's daily schedule and allows patients to mark medications as taken or missed.

PillPopin eliminates the need for patients to contact doctors directly for routine updates on their medication schedules, ensuring a more efficient communication flow. The system helps patients stay on track with their prescriptions by automating reminders for medication intake, reducing the risk of missed doses.

3. Functional Requirements

3.1 User Authentication

- Description: Both doctors and patients must log in using valid credentials to access their respective functionalities within the app.

- Inputs: ID and password.

- Outputs: Access granted to either the doctor or patient dashboard.

- Actors: Doctor, Patient.

3.2 Doctor Functionalities

3.2.1 Add Patient

- Description: The doctor adds new patients to the system by entering their details.

- Inputs: Patient name, ID, birthdate, and other relevant information such as gender.

- Outputs: The system stores the new patient's details in the Firestore database.

- Actors: Doctor.

3.2.2 Manage Patients

- Description: Doctors can view patients' details and prescriptions, including medication information and intake history, but cannot update the patient's personal information like contact details or medical history.

- Inputs: Patient ID.

- Outputs: Displays the patient's profile and prescription history, including medication details.

- Actors: Doctor.

3.2.3 Add Prescription

- Description: The doctor assigns medications to a patient by entering prescription details such as the medication name, dosage, start and end date, dosage times, and any additional doctor's notes. The system validates the patient ID and adds the prescription to the patient's profile.

- Inputs: Patient ID, medicine name, daily dosage, pills per dose, start date, end date, dosage times, and any special doctor's notice.

- Outputs: A new prescription added to the patient's profile.

- Actors: Doctor.

3.2.4 Manage Prescriptions

- Description: Doctors can view and delete prescriptions from a patient's profile.

- Inputs: Patient ID, prescription selection for deletion.

- Outputs: Deleted prescription from the patient's profile.

- Actors: Doctor.

3.2.5 Track Medicine Intake

- Description: The doctor monitors whether the patient has followed the prescribed medication schedule by viewing detailed intake logs. The intake history is displayed, highlighting missed or completed doses for each day.

- Inputs: Patient ID .

- Outputs: Displays the patient's medicine intake history, including the number of taken and missed doses. This includes a visual representation of the data (e.g., bar chart or color-coded display).

- Actors: Doctor.

3.2.6 View Prescription Details

- Description: The doctor can view detailed information about a patient's prescriptions, including medicine name, dosage, start and end dates, and special notes provided by the doctor.

- Inputs: Patient ID, prescription selection.

- Outputs: Displays prescription details, including medication name, daily dose, pills per dose, and the prescription's validity period.

- Actors: Doctor, System.

3.3 Patient Functionalities

3.3.1 View Medications

- Description: Patients can view all the medications prescribed to them, including instructions and warnings.

- Inputs: Patient login credentials (ID, password).

- Outputs: Medication details, including dosage, warnings, and instructions.

- Actors: Patient.

3.3.2 Track Medicine Intake

- Description: The patient logs whether they have taken or missed a scheduled dose of their medication.

- Inputs: Patient ID, intake confirmation for each medication (taken or missed).

- Outputs: Updated medicine intake record.

- Actors: Patient, System.

3.3.3 Update Profile

- Description: The patient can update their personal details, such as name and contact information.

- Inputs: Updated full name, phone number, emergency contact number, email, and birthdate.

- Outputs: Updated patient profile.

- Actors: Patient.

3.3.4 Receive Medicine Reminders (Notifications)

- Description: The system sends notifications to the patient reminding them to take their medication at scheduled times.

- Inputs: Medicine schedule (from prescription data).

- Outputs: Notifications triggered at the specified times.

- Actors: System, Patient.

3.3.5 View Medicine Stock

- Description: Patients can view the remaining stock of their prescribed medications

- Inputs: Patient ID, prescription details.

- Outputs: Medicine stock.

- Actors: Patient, System.

3.4 System Functionalities

3.4.1 Send Notifications

- Description: The system sends reminders to patients to take their medication at the scheduled times. These notifications are based on the patient's prescribed dosage schedule.

- Inputs: Medication schedule (from prescription data).

- Outputs: Reminder notifications sent to the patient at specified times.

- Actors: System, Patient.

3.4.2 Track Medicine Stock

- Description: The system tracks the remaining stock of medicines based on patient intake logs. The patient can view the current stock of their medications.

- Inputs: Patient's medication stock details and daily intake

- Outputs: Updated stock levels displayed to the patient.

- Actors: System, Patient.

3.4.3 Track Prescription End Date

- Description: The system records the start and end dates of each prescription.

- Inputs: Prescription end date.

- Outputs: Displayed prescription details including start and end dates for patient viewing.

- Actors: System, Patient.

4. Non-Functional Requirements

4.1 Performance Requirements

- **PillPopin** ensures that critical interactions, such as user login, prescription retrieval, and medication reminders, are completed within 2 seconds. Reminder notifications are sent within 30 seconds of the trigger event.

4.2 Security Requirements

- All sensitive data, including patient and doctor information, is securely protected both during transmission and when stored. Firebase Authentication is implemented to guarantee that only authorized doctors and patients can access the system. Role-based access control ensures that users can only view or modify the data they are permitted to access.

4.3 Usability Requirements

- PillPopin provides a simple and intuitive interface for both doctors and patients. Doctors can easily manage patients and prescriptions, while patients can navigate medication reminders and prescription details without difficulty. The app is designed to minimize confusion and provide a user-friendly experience.

4.4 Reliability Requirements

- PillPopin operates with a 99% uptime, ensuring consistent access for both doctors and patients. All user actions, including adding prescriptions and tracking medication intake, are saved in real-time to prevent any data loss or corruption. Regular automatic data backups are conducted to ensure data reliability.

4.5 Scalability Requirements

- The system is designed to scale efficiently, handling increasing numbers of users, prescriptions, and medication tracking data as more doctors and patients use the app. The Firestore database scales dynamically to accommodate growth without compromising performance.

4.6 Portability Requirements

- PillPopin works seamlessly across Android platforms. The Firebase services integrated into the app ensure smooth operation and easy deployment across different Android devices without requiring significant modifications.

5. System Architecture

- **PillPopin** is developed using Flutter for cross-platform mobile development and Firebase for backend services. Firestore is used to manage real-time data, providing a scalable and efficient database solution. The system architecture includes the following key components:

5.1 Client-Side (Frontend)

- Flutter Framework: The app's user interface is developed using Flutter, allowing it to run on Android devices. Flutter ensures a consistent UI across platforms with reusable code.

- User Interface: The app has two main roles with distinct UI flows:

- Doctor Interface: Allows doctors to log in, add and manage patients, prescribe medications, view prescription details, and track patient intake logs.

- Patient Interface: Enables patients to log in, view medication details, log their medication intake, receive reminders for taking medication, manage their profile information, and view the current status of their medication stock .

5.2 Backend Services

- Firebase Authentication: Provides secure login and registration for both doctors and patients. It ensures that users' data is protected, and access is restricted to authorized individuals.

- Firestore Database: Firestore is used to store and retrieve patient, doctor, prescription, and medicine intake data in real-time. Data is organized into collections such as patients, prescriptions, and intake history, ensuring efficient data management and access control.

5.3 Cloud Functions (Automation)

- Notification System: Automated reminders for patients to take their medicine are handled using local notifications, triggered on the device based on preset times. These reminders are managed locally .

- Inventory Management: The system tracks medicine stock levels based on the patient's intake data.

- Prescription Expiry Management: The system tracks when a prescription is about to expire.

5.4 Communication Flow

1. User Interaction: Patients and doctors interact with the Flutter app, which processes requests and communicates with Firebase to fetch or update data.

2. Database Operations: The app interacts with Firestore to retrieve and update patient, prescription, and medicine data in real time. Changes made by users are immediately reflected in the Firestore database.

3. Notifications: for medication intake are triggered locally on the device using the flutter_local_notifications package.

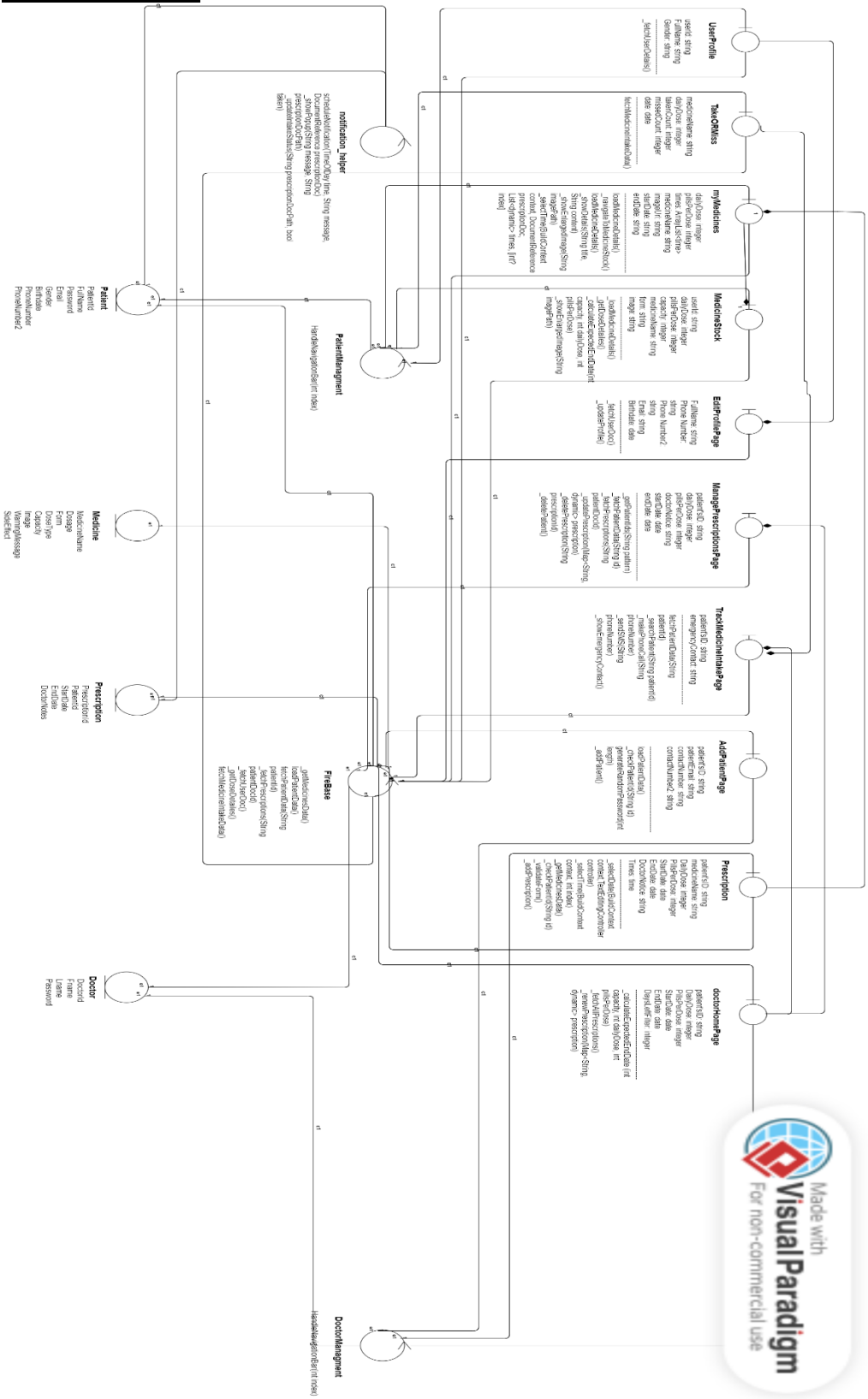
4. Security and Access Control: Firebase Authentication ensures that only registered and authenticated users (doctors and patients) can access the system. Firestore access rules are enforced to protect sensitive patient and prescription data, ensuring only authorized users can view or modify it.

5.5 Deployment Architecture

- Cross-Platform Support: The app is developed using the Flutter framework, allowing it to run on Android devices. Firebase services are used for backend management, providing real-time database operations through Firestore. This ensures that data is updated and synchronized for users without any delays.

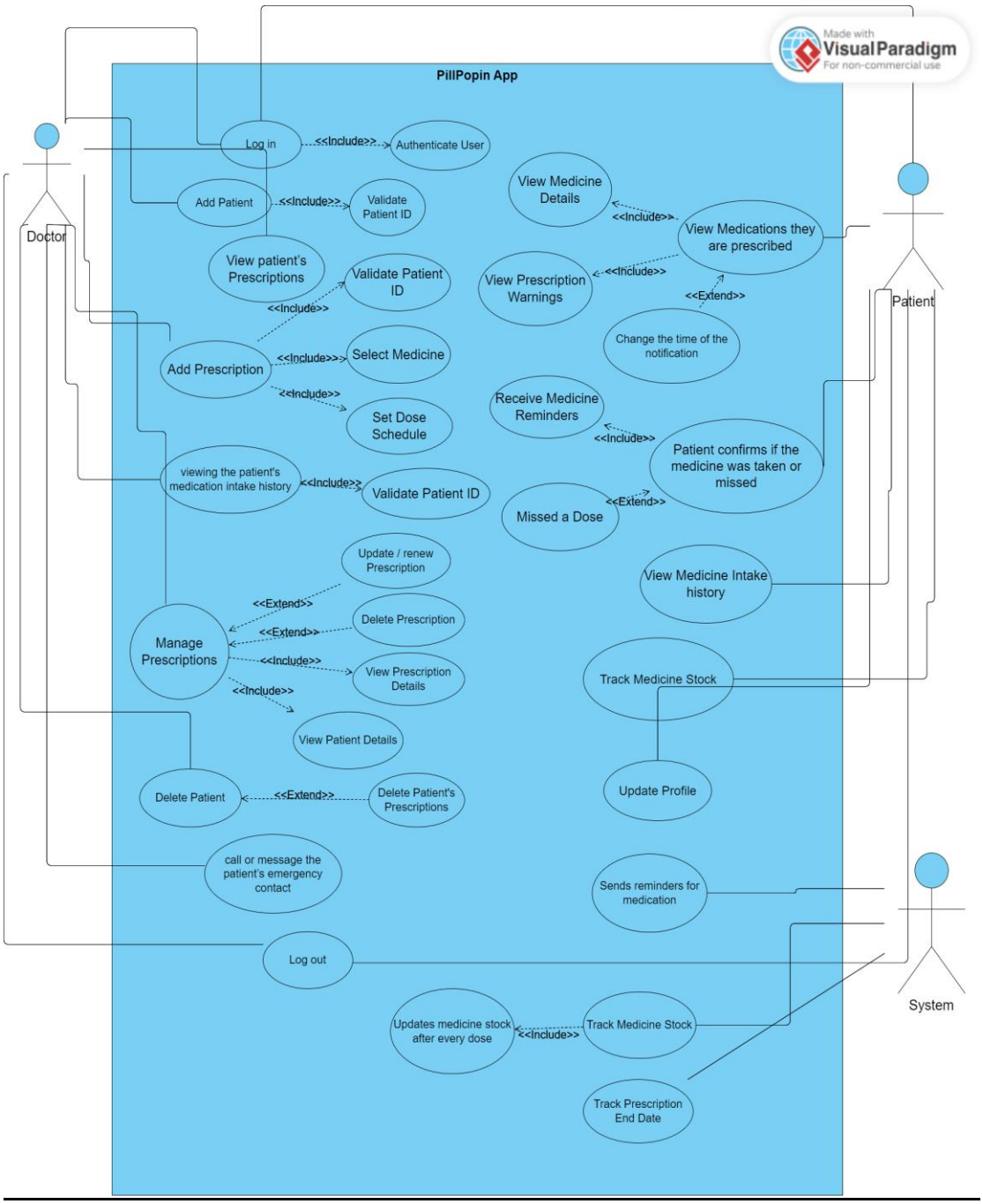
This system architecture ensures that **PillPopin** provides a seamless experience for both doctors and patients, offering real-time interactions, notifications, and secure data management.

6. Class Diagram



m

7. Use Case Diagram



8. Interface Requirements

This section outlines the various interfaces used in the **PillPopin** system, including system interfaces and external interfaces. The user interface screens have already been described in detail in previous sections.

8.1 User Interfaces

The detailed descriptions of user interfaces for both the doctor and patient, including the login page, dashboard, prescription management, medication tracking, and notification center… have been covered in earlier sections.

8.2 System Interfaces

- Firebase Authentication: Interface for managing secure user authentication for both doctors and patients.

- Firebase Firestore: Interface for storing and retrieving patient data, prescriptions, and medication intake records.

- Notification Service: Provides automated medication reminders, low stock alerts, and prescription expiry notifications.

8.3 External Interfaces

- Firebase Cloud Services: Cloud-based services for authentication, real-time database, and notifications.

- Device Notification Services: Interface with mobile devices to send push notifications (via Firebase Cloud Messaging or native platform notifications).

8.4 Hardware Interfaces

- Mobile Devices: The application runs on Android devices and integrates with mobile hardware for sending notifications.

8.5 Software Interfaces

- Operating Systems: Android OS , supporting Flutter applications and Firebase services.

9. System Constraints

This section outlines the limitations and constraints that the **PillPopin** system must adhere to during its development and operation. These constraints include hardware, software, legal, and other limitations that may affect the design or functionality of the system.

9.1 Hardware Constraints

- Device Compatibility: The application is designed to run primarily on Android devices. There is no support for iOS, desktop, or web versions at this time.

- Storage: The app relies on Firebase Firestore for data storage, and there are storage limits associated with Firebase's free tier. Expansion to a paid plan might be required if usage scales significantly.

9.2 Software Constraints

- Operating System: The application requires Android OS to run, which means it cannot function on devices running other operating systems like iOS or Windows without further development.

- Framework: The app is built using Flutter, which requires the Flutter SDK to be installed and updated to maintain compatibility with new OS versions and features.

9.3 Performance Constraints

- Real-Time Sync: The system depends on real-time data synchronization with Firebase Firestore. Any network issues or downtime in Firebase services may result in delays in updating patient data, prescription information, or syncing medicine intake logs.

- Notification Delivery: The app's notification system relies on device connectivity to deliver reminders and updates. If a device is offline, notifications may be delayed until the device is back online.

10. Future Enhancements

The **PillPopin** system has been designed to address the current needs of doctors and patients for managing prescriptions and medication intake. However, there are several potential future enhancements that could improve the app's functionality, scalability, and user experience. These enhancements include the following:

10.1 iOS Compatibility

- Support for iOS Devices*: Expanding the app's compatibility to include iOS devices would make it accessible to a wider user base. This would require adaptation of the app to meet Apple's guidelines and the use of Firebase services on iOS.

10.2 Multi-Language Support

- Additional Languages: Introducing multiple language options, especially for Arabic and other regional languages, would cater to a broader audience. This could include localization of the user interface and app content.

10.3 Expanded Notification System

- Customizable Notifications: Allowing patients to customize when and how they receive notifications (e.g., SMS, email, push notifications) would improve the user experience. Patients could also set up reminders for related activities like follow-up appointments.

- Multiple Reminder Types: Adding more diverse types of notifications for doctors, such as inventory shortage alerts, prescription refill reminders, and patient health updates, could enhance the system's efficiency.

10.4 Offline Functionality

- Offline Mode: Enabling the app to function offline with automatic data synchronization when the device reconnects to the internet would improve accessibility for patients in regions with unreliable connectivity.

11. Appendices

This section provides additional information and resources to support the main content of the SRS document.

11.1 Glossary

A list of key terms, abbreviations, and acronyms used throughout the document:

- SRS: Software Requirements Specification.

- PillPopin: The name of the application.

- Firestore: A real-time NoSQL database from Firebase used to store and sync data.

- Firebase: A backend platform for building mobile and web applications.

- Flutter: A cross-platform development framework used for building the app.

- API: Application Programming Interface.

- UI: User Interface.

- ID: Identifier (used for patient or doctor identification).

11.2 References

The references to all external documents, frameworks, or tools used in the development of the system:

- Flutter Framework Documentation:
https://flutter.dev/docs

- Firebase Documentation:
https://firebase.google.com/docs

- Firestore Documentation:
https://firebase.google.com/docs/firestore

11.3 Technological Stack

A summary of the tools and technologies used in the development of **PillPopin** :

- Programming Language: Dart (for Flutter).

- Framework: Flutter (for cross-platform mobile development).

- Database: Firebase Firestore (for real-time data storage and synchronization).

- Authentication: Firebase Authentication (to secure user login and access control).

- Backend Services: Firebase (for managing the backend operations and storage).