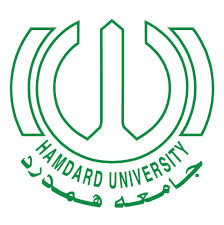
Hamdard University

Department of Computing

Final Year Project



**E-Health**

**(FYP-012/FL24)**

**Software Design Specifications**

Submitted by

Muaaz Ahmed Baig (1726-2021)

Syed Fazal Ullah (2173-2021)

S.M Hamza Sohail (2271-2021)

Supervisor

Sir Mohsin Raza Khan

**Fall 2021**

**Document Sign off Sheet**

### Document Information

|  |  |
| --- | --- |
| **Project Title** | E-Health |
| **Project Code** | FYP-012/FL24 |
| **Document Name** | Software Requirements Specifications |
| **Document Version** | 1.0 |
| **Document Identifier** | FYP-012/FL24 |
| **Document Status** | Final Draft |
| **Author(s)** | Muaaz Ahmed Baig |
| **Approver(s)** | Sir Mohsin Raza Khan |
| **Issue Date** | 14-01-2025 |

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Role** | **Signature** | **Date** |
| Muaaz Ahmed Baig | Team Lead |  |  |
| Syed Fazal Ullah | Team Member 2 |  |  |
| S.M Hamza Sohail | Team Member 3 |  |  |
| Sir Mohsin Raza Khan | Supervisor |  |  |
| ----- | Co-Supervisor |  |  |
| Sir Suleman Naz | Project Coordinator |  |  |

**Revision History**

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Version** | **Description** | **Author** |
| 14-01-2021 | 1.0 | First Draft | Muaaz Ahmed Baig |
| 16-01-2021 | 1.1 | Revised Version | Sir Mohsin Khan |
| 17-01-2021 | 1.2 | Final report | Muaaz, Fazal, Hamza |
|  |  |  |  |

**Definition of Terms, Acronyms, and Abbreviations**

|  |  |
| --- | --- |
| **Term** | **Description** |
| **Scrum** | It is used to agile methodology in this project |
| **Firebase** | Firebase can help the hosting the mobile and web application |
| **Frontend** | It is part of the User Interface which shows in the screen using the HTML, CSS, JavaScript and the modern language that is React Native. |
| **Backend** | The back side of the application that can not shows the user. It can store the user data and help the functionality. |
| **MongoDB** | It is flexible and scalable data storage in the database system. |
|  |  |
|  |  |
|  |  |

# Table of Contents

[1. Introduction Error! Bookmark not defined.](#_Toc4677804)

[1.1 Purpose of Document **Error! Bookmark not defined.**](#_Toc4677805)

[1.2 Intended Audience **Error! Bookmark not defined.**](#_Toc4677806)

[1.3 Abbreviations **Error! Bookmark not defined.**](#_Toc4677807)

[2. Overall System Description Error! Bookmark not defined.](#_Toc4677808)

[2.1 Project Background **Error! Bookmark not defined.**](#_Toc4677809)

[2.2 Problem Statement **Error! Bookmark not defined.**](#_Toc4677810)

[2.3 Project Scope **Error! Bookmark not defined.**](#_Toc4677811)

[2.4 Not In Scope **Error! Bookmark not defined.**](#_Toc4677812)

[2.5 Project Objectives **Error! Bookmark not defined.**](#_Toc4677813)

[2.6 Stakeholders & Affected Groups **Error! Bookmark not defined.**](#_Toc4677814)

[2.7 Operating Environment **Error! Bookmark not defined.**](#_Toc4677815)

[2.8 System Constraints **Error! Bookmark not defined.**](#_Toc4677816)

[2.9 Assumptions & Dependencies **Error! Bookmark not defined.**](#_Toc4677817)

[3. External Interface Requirements Error! Bookmark not defined.](#_Toc4677818)

[3.1 Hardware Interfaces **Error! Bookmark not defined.**](#_Toc4677819)

[3.2 Software Interfaces **Error! Bookmark not defined.**](#_Toc4677820)

[3.3 Communications Interfaces **Error! Bookmark not defined.**](#_Toc4677821)

[4. System Functions / Functional Requirements Error! Bookmark not defined.](#_Toc4677822)

[4.1 System Functions **Error! Bookmark not defined.**](#_Toc4677823)

[4.2 Use Cases **Error! Bookmark not defined.**](#_Toc4677824)

[4.2.1 List of Actors **Error! Bookmark not defined.**](#_Toc4677825)

[4.2.2 List of Use Cases **Error! Bookmark not defined.**](#_Toc4677826)

[4.2.3 Use Case Diagram **Error! Bookmark not defined.**](#_Toc4677827)

[4.2.4 Description of Use Cases **Error! Bookmark not defined.**](#_Toc4677828)

[5. Non - Functional Requirements Error! Bookmark not defined.](#_Toc4677829)

[5.1 Performance Requirements **Error! Bookmark not defined.**](#_Toc4677830)

[5.2 Safety Requirements **Error! Bookmark not defined.**](#_Toc4677831)

[5.3 Security Requirements **Error! Bookmark not defined.**](#_Toc4677832)

[5.4 Reliability Requirements **Error! Bookmark not defined.**](#_Toc4677833)

[5.5 Usability Requirements **Error! Bookmark not defined.**](#_Toc4677834)

[5.6 Supportability Requirements **Error! Bookmark not defined.**](#_Toc4677835)

[5.7 User Documentation **Error! Bookmark not defined.**](#_Toc4677836)

[6. References Error! Bookmark not defined.](#_Toc4677837)

# Introduction

Through this E–Health System, Patients can search clinic or doctors based on specialization, name, or availability. Patients can receive a token number online for their appointment. Displays the current token number being served and an estimated wait time. A laboratory management system will work in the same way as a clinic management system. When scheduling a test, you can see a list of laboratories, their prices, and any other relevant details. This application will provide online medical services and patients can buy their prescribed medicines.

## Purpose of Document

Software Design Specifications (SDS) outline the architecture, components, interfaces, and data for a software system. They serve as a blueprint to guide developers and stakeholders, ensuring alignment on functionality and design. SDS documents help maintain clarity, reduce risks, and streamline the development process.

## Intended Audience

The intended audience for Software Design Specifications (SDS) includes:

1. **Developers**: To understand the architecture and implementation details.
2. **Project Managers**: For planning and tracking project progress.
3. **Testers**: To create test cases based on system design.
4. **Stakeholders/Clients**: For review and approval of the system's design and functionality.
5. **Maintenance Teams**: To ensure efficient updates and troubleshooting.

## Document Convention

* **Formatting**: Headings in bold, body text in standard font.
* **Diagrams**: UML for designs.
* **References**: Cited in [APA/IEEE format].
* **Glossary**: Defines acronyms and technical terms.

## Project Overview

The "E-Health" project is a comprehensive healthcare management system connecting doctors, patients, laboratories, and medical stores. It includes appointment scheduling, laboratory management, and an online pharmacy for prescribed medicines. The system aims to reduce wait times, enhance patient engagement, and improve healthcare outcomes through secure and user-friendly features.

## Scope

1. Patients can search for doctors/clinics by specialization, name, or availability.
2. Online token system for appointments with current token tracking and estimated wait time.
3. Laboratory management for scheduling tests, price comparison, and details.
4. Online medical store for purchasing prescribed medicines.
5. Comprehensive healthcare solution integrating clinics, labs, and pharmacies.

# Design Considerations

The design considerations for the "E-Health" system focus on creating a secure, user-friendly interface that ensures data privacy and scalability. It leverages Agile methodology for adaptability and collaboration while integrating healthcare systems using modern tools like React Native and Node JS.

## Assumptions and Dependencies

**Assumptions**:

1. The system will be used by users with basic internet access and devices.
2. All third-party services (e.g., Firebase, payment gateways) will be available and functional.
3. The platform's user interface will be responsive across different screen sizes.
4. Users will have prior knowledge of using the application for basic tasks.

**Dependencies**:

1. Firebase for authentication and database management.
2. Bootstrap for responsive layout design.
3. External libraries (e.g., Sweet Alert) for enhanced UI features.
4. A stable internet connection for all real-time interactions.

## Risks and Volatile Areas

1. Dependency on third-party services (e.g., Firebase) may lead to downtime or disruptions.
2. Security vulnerabilities in user authentication or payment processing could compromise data.
3. Compatibility issues across browsers or devices might affect the user experience.
4. Handling large volumes of users or data could strain system performance.

## System Level Architecture

1. **System Decomposition**:  
   The system is divided into several key elements or subsystems, including:
   * **User Interface (UI)**: Handles user interaction, including login, signup, and navigation.
   * **Backend Server**: Manages business logic, user authentication, and communication with databases.
   * **Database**: Stores user data, transactions, and other persistent information.
   * **Third-party Integrations**: Includes external services like Firebase for authentication and payment gateways.
2. **Relationships Between Elements**:
   * The UI communicates with the Backend Server for data fetching and sending user actions.
   * The Backend Server interacts with the Database for data storage and retrieval.
   * Third-party Integrations work with the Backend Server to authenticate users and process payments.
3. **Interfaces to External Systems**:
   * Integration with Firebase for user authentication and database management.
   * Integration with Payment Gateways for processing transactions.
   * External services for email notifications and SMS alerts (if applicable).
4. **Major Physical Design Issues**:
   * The UI will be executed on client devices (browsers or mobile apps).
   * The Backend Server will be hosted on cloud platforms or dedicated servers for scalability and reliability.
   * The Database will be managed via cloud services like Firebase or other distributed database systems for fault tolerance.
5. **Global Design Strategies**:
   * **Error Handling**: Use centralized logging and error handling in the Backend **Server** to monitor and track failures.
   * **Scalability**: The system should be designed to scale horizontally with increased user load.
   * **Security**: Apply strong encryption for data storage and secure authentication protocols for user login.

## Software Architecture

The software architecture will follow a three-tier architecture model, which separates the system into three layers:

1. **User Interface Layer (UI Layer)**:
   * The UI Layer is responsible for interacting with the end-user, providing a visually engaging interface for login, navigation, and displaying content.
   * It sends requests to the Middle Tier for business logic processing and data retrieval.
2. **Middle Tier (Application Layer)**:
   * The Middle Tier contains the business logic of the application.
   * It handles requests from the User Interface Layer, processes them, and interacts with the Data Access Layer to fetch or store data.
   * This layer performs user authentication, processes transactions, and applies business rules.
3. **Data Access Layer:**
   * The Data Access Layer manages interactions with the Database.
   * It abstracts the database operations, such as querying and updating data, ensuring the Middle Tier does not need to know the details of data storage.
   * This layer can utilize APIs or direct database queries to interact with the database.

**Interaction Between Layers:**

* User Interface Layer communicates with the Middle Tier via HTTP or API calls, passing data or requests (e.g., user login, order processing).
* The Middle Tier processes the request and interacts with the Data AccessLayer to retrieve or manipulate data.
* Data Access Layer executes the appropriate database queries and returns the result to the Middle Tier, which then sends it back to the UI Layer.

**Architecture Illustration:**

User Interface

(UI Layer)

Middle Tier

(Application Layer)

Data Access Layer

Interacts via API Processes Logic Executes

DB

Queries

Client Device (Browser, App)

Business Logic (Authentication, transaction)

Database (e.g. MySQL, Firebase)

# Design Strategy

1. **Future System Extension or Enhancement**:  
   The system is designed with scalability in mind to accommodate future extensions or enhancements, such as adding new features (e.g., advanced user analytics, additional payment methods). The architecture employs modular design principles, with well-defined boundaries between layers (UI, application logic, data access), enabling easy integration of new modules. Future enhancements can be added with minimal disruption to existing functionality, ensuring a flexible and extensible system.
2. **System Reuse**:  
   Reusability is a key design goal to minimize redundant development efforts. By adopting a layered architecture (UI, middle tier, data access), components such as the authentication module, payment processing, and data access layer can be reused across multiple parts of the system or even in other projects. For example, integrating third-party authentication services or payment gateways can be done without affecting the core logic.
3. **User Interface Paradigms**:  
   The system follows responsive design principles to ensure a consistent and user-friendly interface across different devices (desktops, tablets, mobile). A clean, intuitive interface is prioritized to simplify user interactions, with emphasis on accessibility. The UI layer is designed to be highly modular, with reusable components (e.g., forms, buttons, and navigation menus), allowing easy updates and customizations without altering the entire system.
4. **Data Management**:  
   The system uses a centralized, cloud-based database for data storage and persistence. The data access layer abstracts the database operations, ensuring consistency and separation of concerns. For distribution, data is fetched in real-time via API calls to ensure up-to-date information. For persistence, the system ensures that user data is securely stored and backed up, with robust error handling in place for data integrity. As part of future enhancements, the system can scale by utilizing distributed database solutions for performance and availability.
5. **Concurrency and Synchronization**:  
   Concurrency and synchronization are handled at the Middle Tier to ensure that multiple requests from users are processed efficiently without data conflicts. The system uses asynchronous processing for non-blocking tasks, such as user authentication and database queries. Additionally, locks and transactions are employed at the data access layer to prevent race conditions during concurrent data updates. By using these mechanisms, the system ensures smooth user interaction even under high load conditions, while maintaining data consistency.

# Detailed System Design

A detailed design should include the following:

## Design Class Diagram

1. **User**
   * **Attributes**: userID, name, email, password, role (admin/user)
   * **Methods**: login(), register(), logout(), updateProfile()
2. **Order**
   * **Attributes**: orderID, userID, items, totalPrice, status
   * **Methods**: placeOrder(), updateStatus(), calculateTotal()
3. **MenuItem**
   * **Attributes**: itemID, name, description, price, category
   * **Methods**: getDetails(), updateDetails()
4. **Cart**
   * **Attributes**: cartID, userID, items, totalAmount
   * **Methods**: addItem(), removeItem(), updateCart(), checkout()
5. **Payment**
   * **Attributes**: paymentID, orderID, amount, paymentMethod, status
   * **Methods**: initiatePayment(), confirmPayment(), processRefund()

**Interaction**:  
The User class interacts with the Cart and Order classes to manage orders. The Cart class stores items added by the user, and the Order class processes those items into an order. Payment processes the transaction, while MenuItem provides item details.

## Database Design

**Logical Data Model (E/R Model)**:

* **Entities**:
  1. **User**: userID (PK), name, email, password, role
  2. **Order**: orderID (PK), userID (FK), totalPrice, status
  3. **MenuItem**: itemID (PK), name, description, price, category
  4. **Cart**: cartID (PK), userID (FK), totalAmount
  5. **Payment**: paymentID (PK), orderID (FK), amount, paymentMethod, status
* **Relationships**:
  1. A User can have multiple Orders.
  2. A Cart belongs to a User.
  3. An Order is related to a Payment.
  4. A Cart contains multiple MenuItems (many-to-many).

### ER Diagram

**Payment**

Payment ID (PK)

Order ID (FK)

amount

payment method

status

**Order**

Order ID (PK)

User ID (FK)

totalPrice

status

**USER**

User ID (PK)

name

email

password

**MenuItem**

Item ID (PK)

name

description

price

category

**Cart**

cart ID (PK)

User ID (FK)

totalAmount

### Data Dictionary

**6.1.2.1 Data 1: User**

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | | |  | | --- | | **Field Name** |  |  | | --- | |  | |  |  | | --- | |  | | |  | | --- | | **Description** |  |  | | --- | |  | | **Type** | **Required** | |  | | --- | | **Example** |  |  | | --- | |  | |
| |  | | --- | | **userID** |  |  | | --- | |  | | |  | | --- | |  | | Unique identifier for the user. | |  |  | | --- | |  | | |  | | --- | | ObjectId |  |  | | --- | |  | | |  | | --- | | Yes |  |  | | --- | |  | | |  | | --- | | "64a76f29b92b14d8f5c9876e" |  |  | | --- | |  | |
| |  | | --- | | **name** |  |  | | --- | |  | | |  | | --- | | Full name of the user. |  |  | | --- | |  | | String | |  | | --- | | Yes |  |  | | --- | |  | | |  | | --- | | "Muaaz Ahmed" |  |  | | --- | |  | |
| **email** | |  | | --- | | User’s email address for communication and login. |  |  | | --- | |  | | String(unique) | |  | | --- | | Yes |  |  | | --- | |  | | “muaazahmed0111@gmail.com” |
| **password** | |  | | --- | | Encrypted password for user authentication. |  |  | | --- | |  | | String | |  | | --- | | Yes |  |  | | --- | |  | | |  | | --- | | "$2b$10$5dEw6p..." |  |  | | --- | |  | |
| **role** | |  | | --- | | Role of the user (e.g., Patient, Doctor, Admin). |  |  | | --- | |  | | |  | | --- | | String (enum) |  |  | | --- | |  | | |  | | --- | | Yes |  |  | | --- | |  | | |  | | --- | | "Patient" |  |  | | --- | |  | |
| **createdAt** | |  | | --- | | Timestamp of account creation. |  |  | | --- | |  | | Date | |  | | --- | | Yes |  |  | | --- | |  | | |  | | --- | | "2025-01-22T12:00:00Z" |  |  | | --- | |  | |
| |  | | --- | | **updatedAt** |  |  | | --- | |  | | |  | | --- | | Timestamp of last account update. |  |  | | --- | |  | | Date | No | "2025-01-22T12:30:00Z" |

#### Data 2

.

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | | |  | | --- | | **Field Name** |  |  | | --- | |  | |  |  | | --- | |  | | |  | | --- | | **Description** |  |  | | --- | |  | | **Type** | **Required** | |  | | --- | | **Example** |  |  | | --- | |  | |
| |  | | --- | | **userID** |  |  | | --- | |  | | |  | | --- | |  | | |  | | --- | | Unique identifier for the appointment. |  |  | | --- | |  | | |  |  | | --- | |  | | |  | | --- | | ObjectId |  |  | | --- | |  | | |  | | --- | | Yes |  |  | | --- | |  | | |  | | --- | | "64a76f29b92b14d8f5c9876e" |  |  | | --- | |  | |
| |  |  |  | | --- | --- | --- | | |  | | --- | | **patientId** |  |  | | --- | |  | |  |  | | --- | |  | | |  |  |  | | --- | --- | --- | | |  | | --- | | Reference to the patient's unique identifier (Users collection). |  |  | | --- | |  | |  |  | | --- | |  | | ObjectId | |  | | --- | | Yes |  |  | | --- | |  | | |  | | --- | | "64a76f29b92b14d8f5c1234a" |  |  | | --- | |  | |
| |  | | --- | | **doctorId** |  |  | | --- | |  | | |  |  |  | | --- | --- | --- | | |  | | --- | | Reference to the doctor's unique identifier (Users collection). |  |  | | --- | |  | |  |  | | --- | |  | | ObjectId | |  | | --- | | Yes |  |  | | --- | |  | | |  |  | | --- | --- | | "64a76f29b92b14d8f5c5678b" |  |  |  | | --- | |  | |
| **appointmentDate** | |  |  |  | | --- | --- | --- | | |  | | --- | | Date and time of the appointment. |  |  | | --- | |  | |  |  | | --- | |  | | Date | |  | | --- | | Yes |  |  | | --- | |  | | |  | | --- | |  | | "2025-01-25T10:00:00Z" | |  |  | | --- | |  | |
| |  | | --- | | **status** |  |  | | --- | |  | | |  |  |  | | --- | --- | --- | | |  | | --- | | Current status of the appointment (e.g., "Scheduled", "Completed", "Cancelled"). |  |  | | --- | |  | |  |  | | --- | |  | | |  | | --- | | String |  |  | | --- | |  | | |  | | --- | | Yes |  |  | | --- | |  | | |  |  |  | | --- | --- | --- | | |  | | --- | | "Scheduled" |  |  | | --- | |  | |  |  | | --- | |  | |
| **createdAt** | |  |  |  | | --- | --- | --- | | |  | | --- | | Timestamp of when the appointment was created. |  |  | | --- | |  | |  |  | | --- | |  | | Date | |  | | --- | | Yes |  |  | | --- | |  | | |  | | --- | | "2025-01-22T12:00:00Z" |  |  | | --- | |  | |
| |  | | --- | | **updatedAt** |  |  | | --- | |  | | |  |  |  | | --- | --- | --- | | |  | | --- | | Timestamp of the last update to the appointment |  |  | | --- | |  | |  |  | | --- | |  | | Date | No | "2025-01-22T12:30:00Z" |

#### Data 3

#### 

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  | | --- | | **Field Name** |  |  | | --- | |  | | |  | | --- | | **Description** |  |  | | --- | |  | | |  | | --- | | **Type** |  |  | | --- | |  | | |  | | --- | | **Required** |  |  | | --- | |  | | |  | | --- | | **Default** |  |  | | --- | |  | | | **Reference** | | --- |  |  | | --- | |  | |
| |  | | --- | | **\_id** |  |  | | --- | |  | | |  | | --- | | Unique identifier for the payment (auto-generated by MongoDB) |  |  | | --- | |  | | |  | | --- | | ObjectId |  |  | | --- | |  | | Yes | |  | | --- | | Auto-generated |  |  | | --- | |  | | ---- |
| |  | | --- | | **applicationID** |  |  | | --- | |  | | |  | | --- | | Reference to the linked application for payment. |  |  | | --- | |  | | |  | | --- | | ObjectId |  |  | | --- | |  | | Yes | ---- | |  | | --- | | References Application collection |  |  | | --- | |  | |
| |  | | --- | | **amount** |  |  | | --- | |  | | |  | | --- | | Payment amount. |  |  | | --- | |  | | |  | | --- | | Decimal128 |  |  | | --- | |  | | Yes | ----- | ---- |
| |  | | --- | | **status** |  |  | | --- | |  | | |  | | --- | | Payment status (e.g., Paid, Pending, Failed). |  |  | | --- | |  | | |  | | --- | | String |  |  | | --- | |  | | Yes | ----- | ---- |
| |  | | --- | | **timestamp** |  |  | | --- | |  | | |  | | --- | | Timestamp of the payment transaction. |  |  | | --- | |  | | |  | | --- | | Date |  |  | | --- | |  | | Yes | Date.now | ---- |

**6.2.2.3 Data 4**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  | | --- | | **Field Name** |  |  | | --- | |  | | |  | | --- | | **Description** |  |  | | --- | |  | | |  | | --- | | **Type** |  |  | | --- | |  | | |  | | --- | | **Required** |  |  | | --- | |  | | |  | | --- | | **Default** |  |  | | --- | |  | | | **Reference** | | --- |  |  | | --- | |  | |
| |  | | --- | | **\_id** |  |  | | --- | |  | | |  |  |  | | --- | --- | --- | | |  | | --- | | Unique identifier for the notification (auto-generated by MongoDB) |  |  | | --- | |  | |  |  | | --- | |  | | |  | | --- | | ObjectId |  |  | | --- | |  | | Yes | |  | | --- | | Auto-generated |  |  | | --- | |  | | ---- |
| |  |  |  | | --- | --- | --- | | |  | | --- | | **recipientID** |  |  | | --- | |  | |  |  | | --- | |  | | |  |  |  | | --- | --- | --- | | |  | | --- | | Reference to the user receiving the notification. |  |  | | --- | |  | |  |  | | --- | |  | | |  | | --- | | ObjectId |  |  | | --- | |  | | Yes | ---- | |  |  |  | | --- | --- | --- | | |  | | --- | | References User collection |  |  | | --- | |  | |  |  | | --- | |  | |
| |  |  |  | | --- | --- | --- | | |  | | --- | | **message** |  |  | | --- | |  | |  |  | | --- | |  | | |  |  |  | | --- | --- | --- | | |  | | --- | | Content of the notification (email/SMS content). |  |  | | --- | |  | |  |  | | --- | |  | | |  |  |  | | --- | --- | --- | | |  | | --- | | String |  |  | | --- | |  | |  |  | | --- | |  | | Yes | ----- | ---- |
| |  |  |  | | --- | --- | --- | | |  | | --- | | **sentDate** |  |  | | --- | |  | |  |  | | --- | |  | | |  |  |  | | --- | --- | --- | | |  | | --- | | Date and time the notification was sent. |  |  | | --- | |  | |  |  | | --- | |  | | |  | | --- | | String |  |  | | --- | |  | | Yes | Date.now | ---- |

## GUI Design

**Introduction & Login/Register:**

A screenshot of a phone

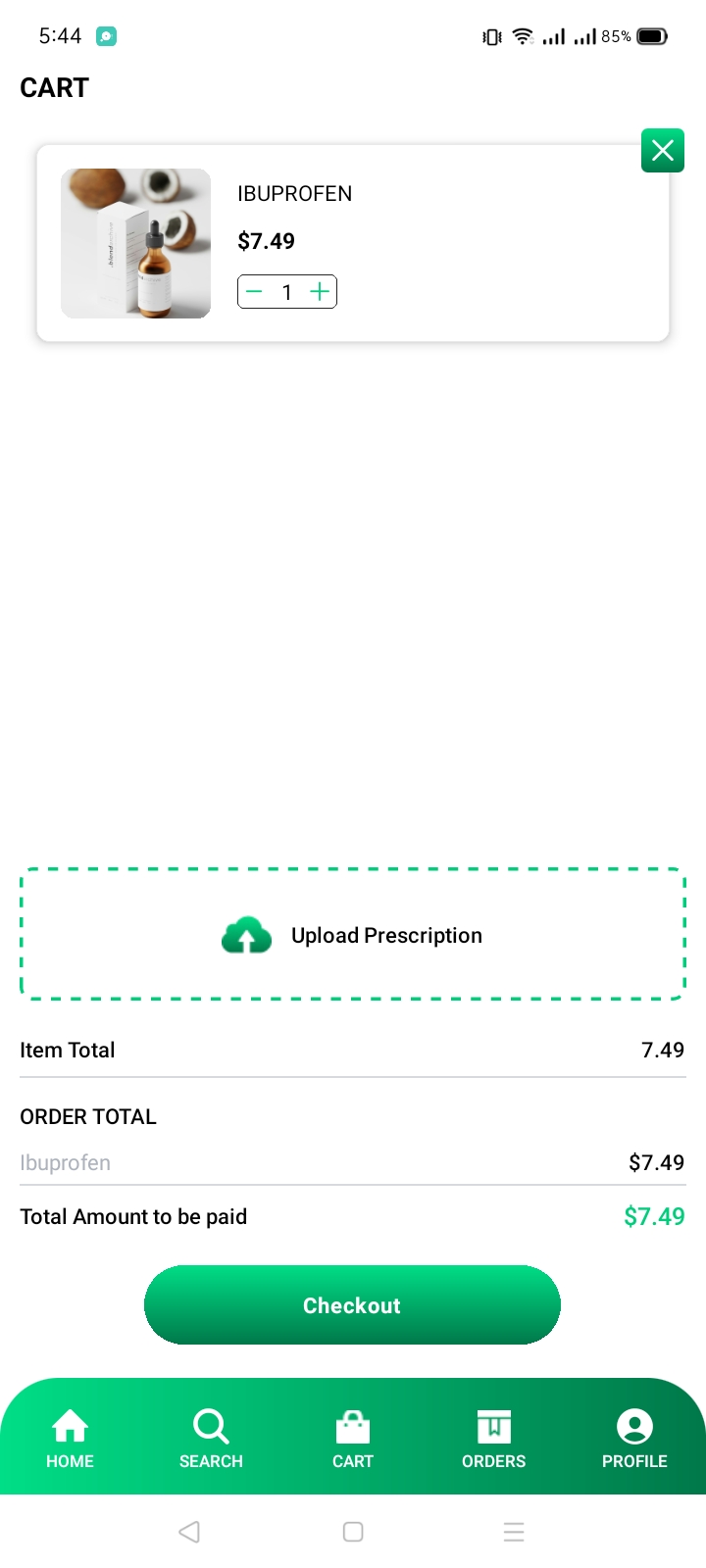
Description automatically generated A person standing next to a tablet

Description automatically generated

# 

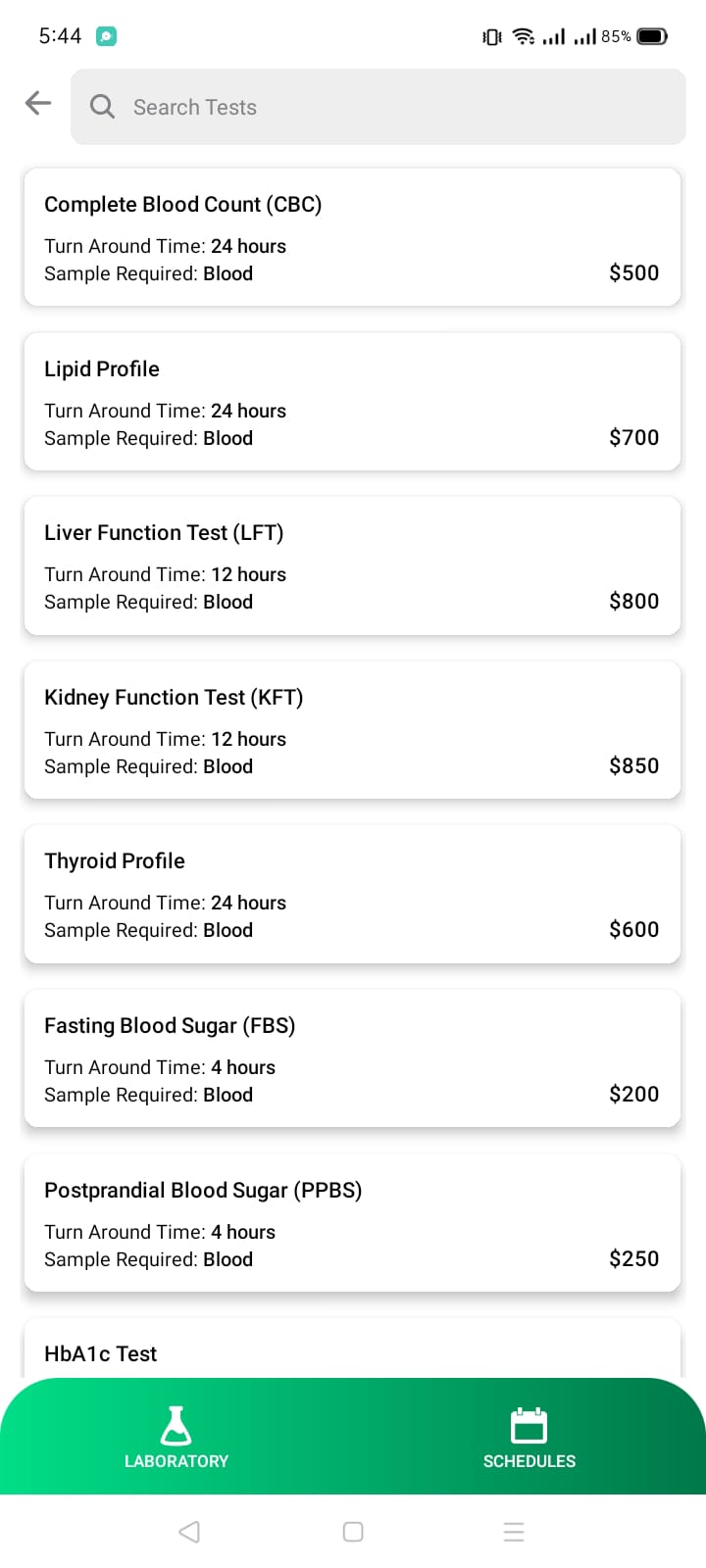
# 

**Medical Store UI:**

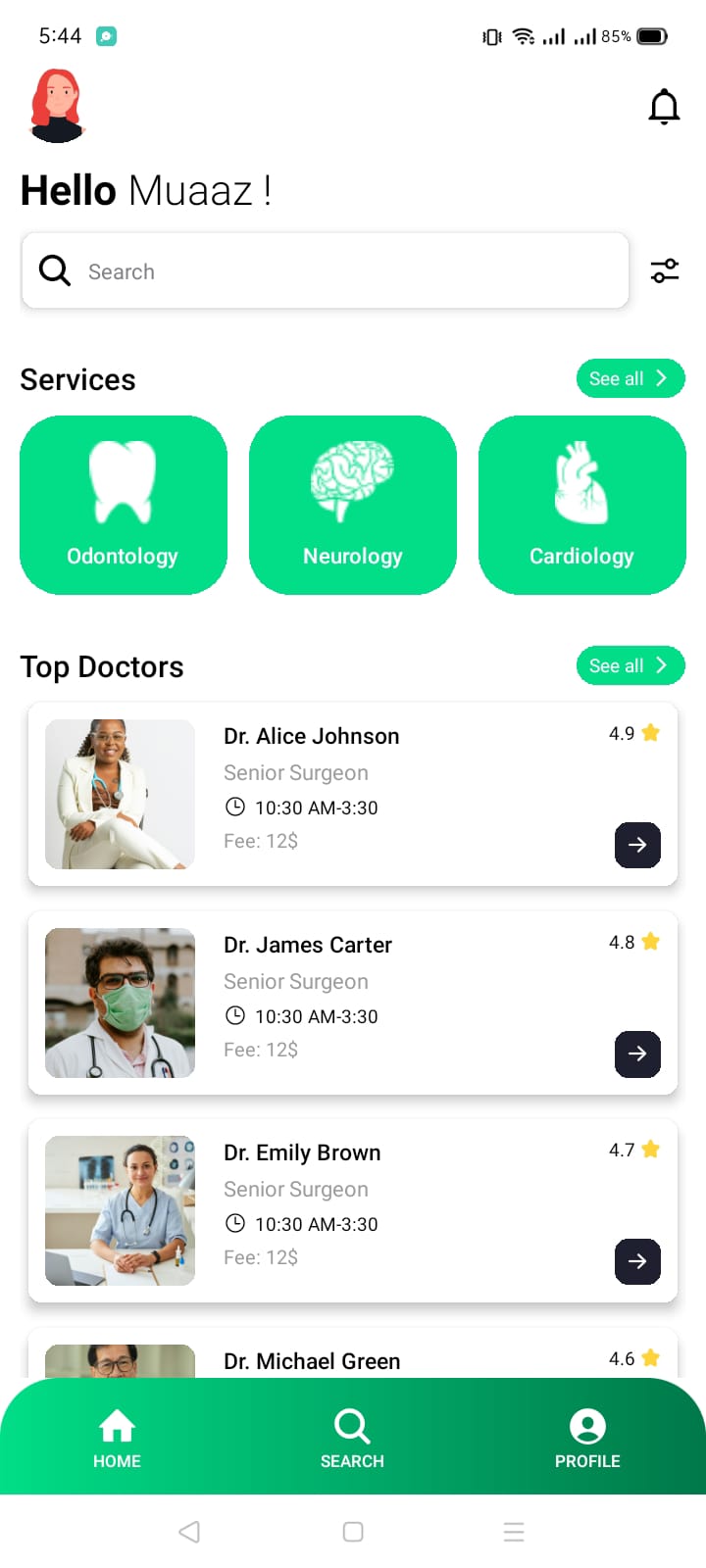
A screenshot of a phone

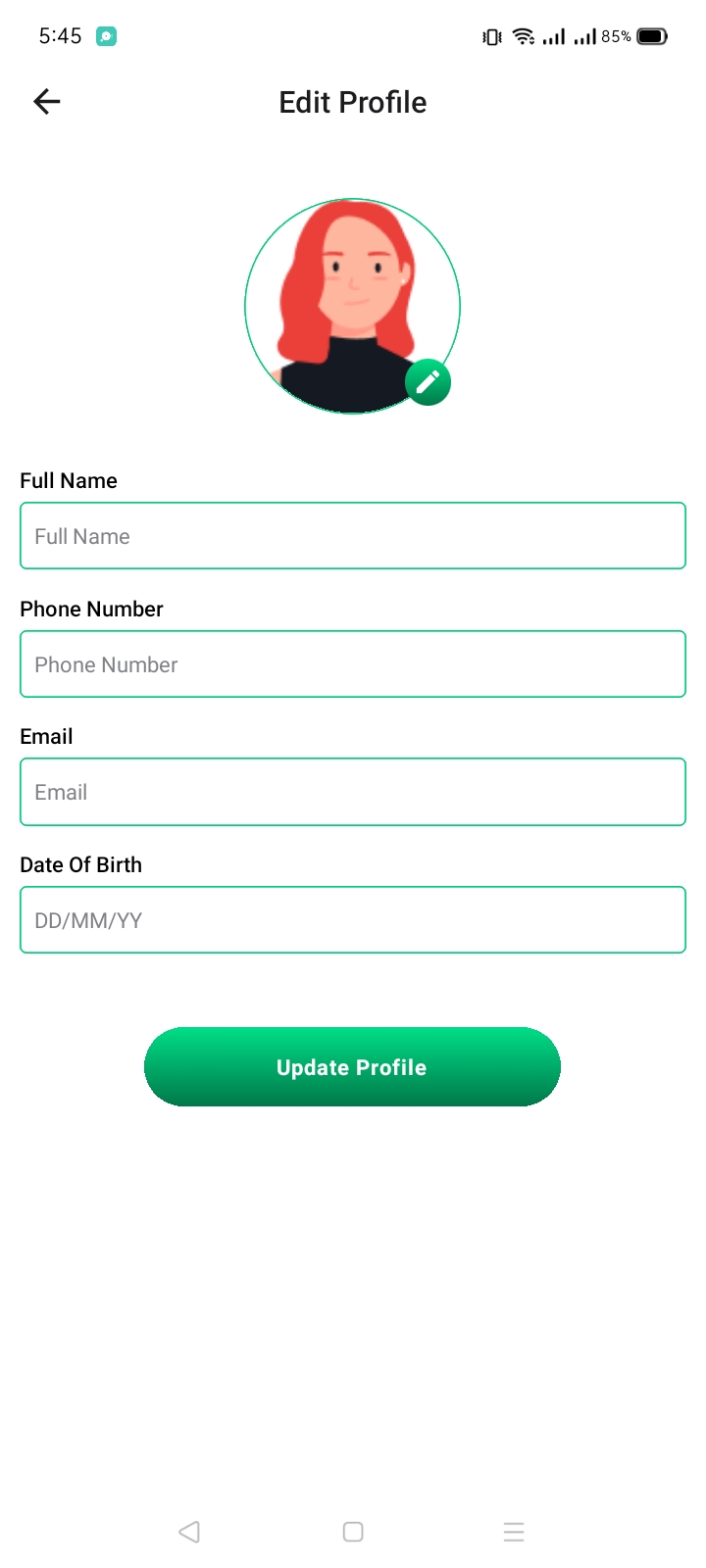
Description automatically generated

**Laboratory UI:**



**Clinic/Doctor Appointment UI:**





# References

* https://www.dvago.pk/
* https://www.cbc.ca/news/health/hospital-patients-may-worry-less-after- seeing-their -medical-records
* https://www.cbc.ca/news/health/patients-gain-when-doctors-notes- are%20online
* https://doi.org/10.1145/2379776.2379779
* https://doi.org/10.1258/1357633001934113
* Online Health Records (2011) https://www.cbc.ca/news/canada/toronto/online-health-records-popular-with

# 