#### Capstone Project (CPRO306)

**Assessment 5: Systems Design and Implementation Plan**

#### **Project Title:** **Austin Hospital Management System**

|  |  |  |
| --- | --- | --- |
| **Team Members** | | |
| **No.** | **Name** | **SID** |
| **1** | **Bianca Carvalho** | **K221552** |
| **2** | **Lunshwa Shakya** | **K231451** |
| **3** | **Santiago Ortiz** | **K200370** |
| **4** | **Sulav Homagai** | **K220028** |
| **5** | **Willow Postlethwaite** | **K191297** |

**Submission Date:** Week 9

[Capstone Project (CPRO306) 1](#_Toc177322002)

[**Project Title:** **Austin Hospital Management System** 1](#_Toc177322003)

[**1. Introduction** 3](#_Toc177322004)

[**Project Overview** 3](#_Toc177322005)

[**Objectives of the Report** 3](#_Toc177322006)

[**2. System Development Planning and Management** 3](#_Toc177322007)

[**Project Planning** 3](#_Toc177322008)

[**Resource Management** 4](#_Toc177322009)

[**Timeline and Milestones** 4](#_Toc177322010)

[**3. Teamwork Management** 4](#_Toc177322011)

[**Team Roles and Responsibilities** 4](#_Toc177322012)

[**Communication Strategies** 4](#_Toc177322013)

[**Collaboration Tools Used** 5](#_Toc177322014)

[**4. Business Requirements Elicitation** 5](#_Toc177322015)

[**Requirement Gathering Process** 5](#_Toc177322016)

[**Business Requirements Document** 6](#_Toc177322017)

[**Stakeholder Analysis** 7](#_Toc177322018)

[**5. System Architecture** 9](#_Toc177322019)

[**Architecture Diagram** 10](#_Toc177322020)

[**6. Database Design** 10](#_Toc177322021)

[**Database Schema Overview** 10](#_Toc177322022)

[**Physical Layout and Structure** 11](#_Toc177322023)

[**Tables, Keys, and Relationships** 11](#_Toc177322024)

[**Entity-Relationship Diagram (ERD)** 16](#_Toc177322025)

[**Database Design Tools Used** 17](#_Toc177322026)

[**7. User Interface Design** 18](#_Toc177322027)

[**7.1** **UI Design Principles** 18](#_Toc177322028)

[**8. Programming and Implementation** 52](#_Toc177322029)

[**Code Overview** 52](#_Toc177322030)

[**Frontend Code** 52](#_Toc177322031)

[**Technologies Used:** 52](#_Toc177322032)

[**Backend Code** 54](#_Toc177322033)

[**9. Screenshots and File Attachments** 56](#_Toc177322034)

[**10. Conclusion** 60](#_Toc177322035)

[**11. References** 60](#_Toc177322036)

### **1. Introduction**

#### **Project Overview**

This report has been created to present the overview of the system built for Austin Hospital, it will provide a comprehensive insight into the development, architecture or structure of the database and the physical layout of the system for the case study. The objective is to outline the detailed methodologies and tools that we used throughout the project to create the front-end interface, how the database schema was designed and how the Entity Relationship Structure is laid out.

The report has been structured and to include evidence of the system in its current state of the project, evidence can include the files and folders for the system, as well as screenshots of elements. The visual evidence intends to give a clear and concise view of the project and system structure, to ensure there is understanding of the integration and development approach of the AHMS.

#### **Objectives of the Report**

The report's primary objectives are to explain the processes and schema. The report will outline the and document the following:

1. Document and outline the front-end development process
2. Describe and outline the database layout
3. Showcase the Database Schema
4. Showcase the Entity Relationships via an ERD
5. Outline the tools used and design decisions
6. Ensure project transparency
7. Outline the interface functions and requirements

By achieving the objectives highlighted above the report will help with transparency on the planning, development and implementation phases to ensure all individuals are working towards the same end-goal.

### **2. System Development Planning and Management**

#### **Project Planning**

A thorough project planning process was necessary for the creation of the Austin Hospital Management System (AHMS). The group first engaged with hospital stakeholders, including physicians, patients, and the administration, to perform a complete need analysis. This made It easier to comprehend the critical features the system required, like appointment scheduling, patient registration, and safe patient data processing. After then, the project was divided into smaller, more doable tasks, such as database setup, back-end API integration, and front-end design. Each work was assigned a specific completion date through the establishment of clear timeframes, and risks, such as possible delays or technological difficulties were recognized early on and properly mitigated. Iterative development and the ability to modify plans in response to new requirements were made possible by the selection of the agile methodology.

#### **Resource Management**

Effective resource management was essential to the AHMS’s effective deployment. The team member’s areas of expertise were taken into consideration while allocating the human resources: for instance, front-end developers took care of the user interface, while Back-end developers concentrated on server-side tasks like data administration and security. Technological resources included version control tools like Git and database management systems like MySQL for handling big medical datasets. These instruments guaranteed that the system was safe, scalable, and tailored to the hospital’s requirements. Effective budget management was also necessary, involving the prudent distribution of funds among hardware and software necessities and the establishment of a reserve account for unforeseen costs or expansion needs.

#### **Timeline and Milestones**

The AHMS project followed a planned schedule, guaranteeing that every stage was finished on schedule. Early on, important benchmarks were determined, such as the start of the project, front-end and back-end development, database integration, testing, and ultimate deployment. Weekly sprints gave the team the opportunity to concentrate on immediate objectives while maintaining an eye on the project’s overall schedule. The goal of each sprint was to produce a functioning version of the system, which was examined and tested prior to going on to the following milestone. By using an iterative process, the team was able to address issues and incorporate feedback from stakeholders while maintaining constant progress before proceeding to the next step.

### **3. Teamwork Management**

#### **Team Roles and Responsibilities**

Successful management of teamwork was essential to the AHMS project’s success. The team duties were well-defined from the outset, guaranteeing that each member had a distinct duty that aligned with their abilities. The project manager managed all aspects of the development process, arranged for stakeholder engagement with the hospital, and reduced risks. Front-end developers worked on creating the user interface so that patients could make appointments and create accounts. Testers made that the system was error-free and operated as intended, while back-end developers handled server-side operations like database interactions and security. After the system was deployed, the IT staff continued to maintain it by offering technical assistance and managing any updates or enhancements.

#### **Communication Strategies**

To make sure that everyone in the team was on the same page and working toward the same objectives, clear communication was crucial. The team used a variety of communication techniques, such as daily stand-ups when each member reported on their accomplishments and any difficulties they were having. The team was able to discuss the project’s overall progress in greater detail at weekly review sessions, which made it easier to modify timetables or deal with unforeseen problems. Maintaining regular updates and feedback loops with stakeholders ensured that the system adapted to the Hospital’s changing demands. Stakeholder communication was also crucial. By using these tactics, the team was able to keep the project moving forward by promptly identifying and resolving any obstacles.

#### **Collaboration Tools Used**

The utilisation of many collaboration technologies improved project management and teamwork. Team members were able to communicate in real time by using Slack, which allowed for rapid queries and updates. Trello was used to manage tasks, giving everyone on the team a clear picture of what they were working on and guaranteeing accountability. Version control was made easier by GitHub, allowing several engineers to work on the system at once without sacrificing the quality of the code. To make sure that everyone had access to the most recent project material, Google Drive was used for document storage and sharing, including architecture diagrams, database designs, and meeting minutes. The team was able to work together effectively even when working asynchronously or distantly thanks to these technologies.

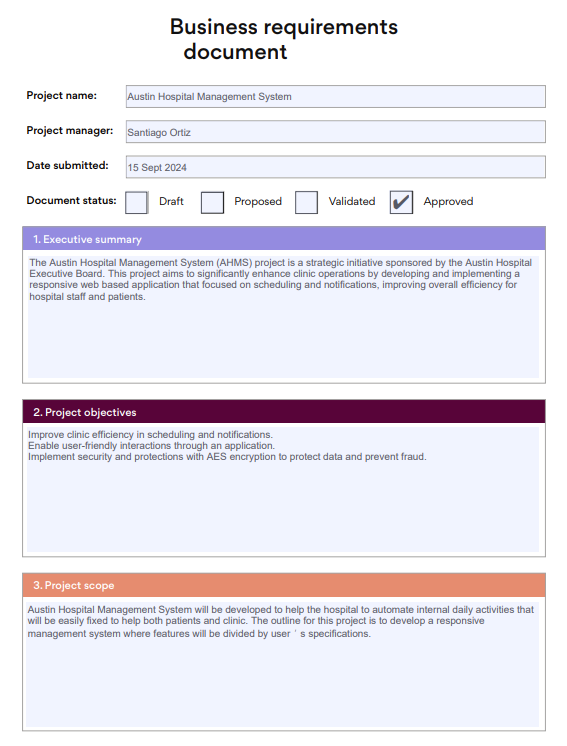
### **4. Business Requirements Elicitation**

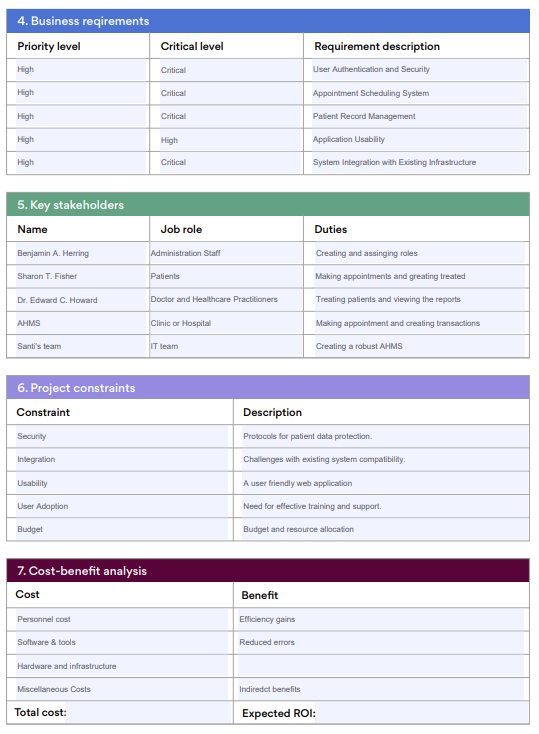
#### **Requirement Gathering Process**

The requirement gathering process includes the following steps:

|  |  |
| --- | --- |
| **Assign Roles** | The stakeholders are identified and assigned roles to determine how they interact with each AHMS component. The scope of the project can be defined when the roles are assigned. The project manager, teams with roles such as developer and tester are also defined. This helps identify the requirements and the roles which can help identify project risk. |
| **Identifying stakeholders** | The stakeholders are identified based on the requirements document provided by the hospital. Each stakeholder is categorised based on internal and external stakeholder. The patients are the external stakeholders whereas all other actors as cashier, doctor, clinic itself are an external entity. Similar projects as hospital management helps identify any loopholes in the system and provide insights. |
| **Gather & Documentation** | As the stakeholders are classified and identified the roles and responsibilities of the stakeholders are documented. |
| **Assumption & Requirements** | As per the requirement documentation the requirements goals are created as scheduling of the project, creating a team based on their skills. The documentation further helps us identify the project risks. |
| **Getting approval** | The gathered requirements are approved by the stakeholders. The documentation helps the stakeholders identify the scope of the project focusing on limitations of the project. The limitation gives stakeholders a sense of direction on what they can expect from the project. The stakeholders, especially the team, know the implementation of the project. |
| **Monitor progress** | This step helps us monitor the progress of AHMS. This means the revision in the project budget throughout project execution. |

#### **Business Requirements Document**

****

****

#### **Stakeholder Analysis**

|  |  |
| --- | --- |
|  |  |
| **Administration Staff** | * Admin staff can set the level of abstraction ensuring user accesses are accurate. Their primary role ensures the system complies with healthcare regulations. * The administration staff creates an account for cashiers, doctors, and other clinic staff. |
| **Patients** | * The users utilise and access their healthcare services, appointments, and medical records. * Patients require a system to find a doctor and get treatment. Also, the system to securely hold their treatment history for any future references. * The AHMS makes it easier for patients to access their report and make an appointment or post an AES key. |
| **Doctor & HealthCare Practitioners** | * Doctor or healthcare practitioners manage patient’s medical records and communicate with patients once a report is received. * Doctors require the system that helps them access and store the patient data in their treatment sessions. * Doctors can view the AES key posted by the patient and view the patient report once the hospital assigns it. |
| **Clinic staff or Hospital** | * The clinic staff allocates and monitors the resources in the hospital and makes or updates an appointment. * Clinic staff require a system to update appointments and allocate the hospital resources. * Clinic staff can update any appointment in-hand over the phone in real-time. |
| **IT Team** | * The IT team designs, develops, and maintains the AHMS, to meet scope and operate securely within the IoT. * The IT team requires requirements and gathers them. Also, the IT team provides online support and frequent updates if there are any changes in the future. |

### **5. System Architecture**

|  |  |
| --- | --- |
|  |  |
| **Architectural Design Overview** | The architectural design consists of the components, technology used to develop and the design patterns that are followed throughout the system. |
| **System Components** | The main component of the system is the front-end which allows patients to create an account, login, and make an appointment. The back end handles the application processing and manages the data. The database holds and manages the data. |
| **Technology** | * Front-End: HTML, CSS, JavaScript. * Back-End: PHP. * Database: MySQL. * Development Tools: Git (for version control), and Postman (for API testing). |
| **Functional Requirements** | * Admin managing user accounts * Doctor managing complaints, clinic updates, and viewing AES keys. * Patients searching for doctors, making booking, and posting AES keys. * Cashier processing payments. * Clinic generating reports. * User authentication (password-based). * Data backup & recovery. |
| **Non-Functional Requirements** | * Responsiveness across different dimensions. * Patient’s data protection from unauthorised access. * Secured from unauthorised access. * AHMS to be scalable to accommodate for future occurrence and changes. * System should be available and consistent. * The system should have proper documentation and user training guides and manuals. * The system should take minimum response time. * Able to maintain the code and reuse it. * Performance of the system in terms of speed, efficiency, and resources. |

#### **Architecture Diagram**

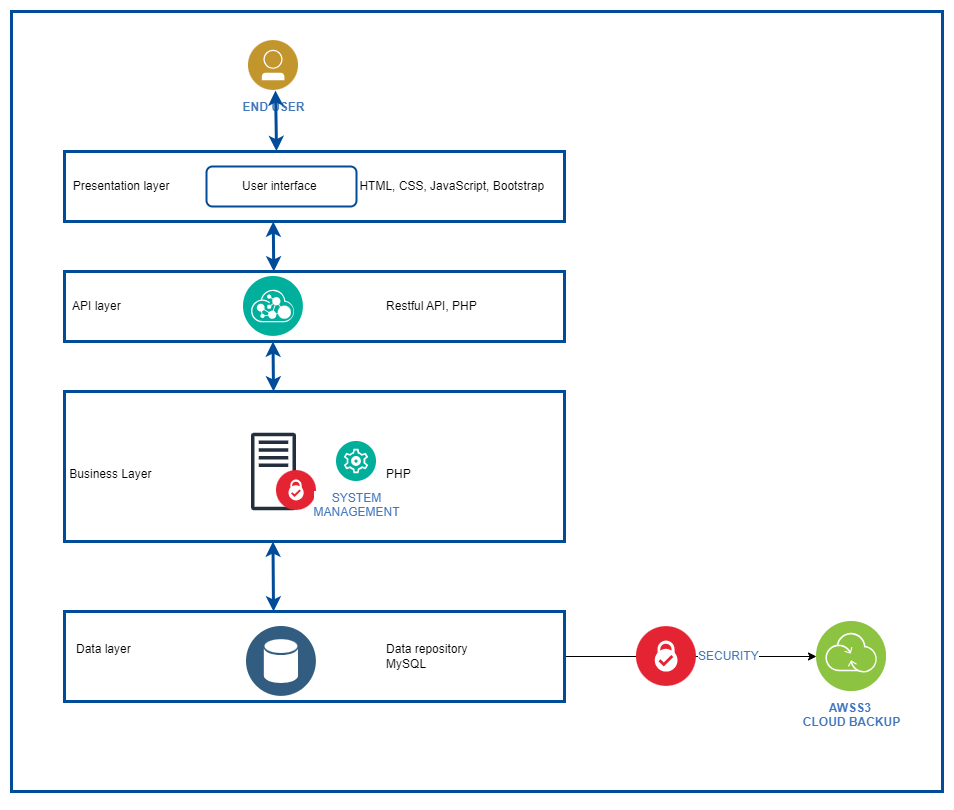
****

Fig. Architecture Diagram

### **6. Database Design**

#### **Database Schema Overview**

Everything is always planned down to the last detail, and this is reflected in the design of the database schema as it integrates a broad range of functions that may transpire in a hospital, including patient profile, doctor’s schedule, financial processes, and complaints handling among others. Apart from being the framework for hospital’s administrative activities, it also guarantees the synchronisation of multiple departmental functions for the sake of proper and smooth operation.

The most important aspects of the schema are, for example, the proper storage of confidential information of patients, personnel timetables, and correct record of all financial operations including billing and payments, etc. However, the schema presents sound mechanisms of handling and responding to patient concerns hence increasing the satisfaction of the patrons of hospital services.

In designing the structure of the database, the major concern is the integrity and consistency of the data. In doing so, the relational schema also sets up well-confirmed relations between the tables and primary and foreign keys guarantee that the information processing will be complete and effective. In addition, indexes help in query optimization so that it can quickly produce essential data for intended analysis, which is very important in a busy health care setting.

This schema also enables scalability in the system in that the hospital can extend its operations or add on new functionalities that it may decide on in the future and these will not compromise on the performance of the system. Altogether, the fact observed in the design is the insight of the hospital’s management to its operational requirements which are then fulfilled in the current state but also adjustable for future expansion.

#### **Physical Layout and Structure**

##### **Tables, Keys, and Relationships**

1. aes\_keys

|  |  |  |  |
| --- | --- | --- | --- |
| **Column Name** | **Data Type** | **KeyType** | **Description** |
| id | int(11) | Primary | Unique identifier for each AES key |
| patient\_id | int(11) | Foreign | References users.id (patient) |
| doctor\_id | int(11) | Foreign | References users.id (doctor) |
| aes\_key | varchar(255) |  | Encrypted key associated with patient and doctor |

**Relationships:**

* **aes\_keys.patient\_id → users.id**
* **aes\_keys.doctor\_id → users.id**

2. appointments

|  |  |  |  |
| --- | --- | --- | --- |
| **Column Name** | **Data Type** | **Key Type** | **Description** |
| id | int(11) | Primary | Unique identifier for each appointment |
| doctor\_id | int(11) | Foreign | References user\_details.user\_id (doctor) |
| patient\_id | int(11) | Foreign | References user\_details.user\_id (patient) |
| availability\_id | int(11) | Foreign | References availability.id |
| status | enum |  | Status of the appointment (Pending, Accepted, Rejected) |
| created\_at | timestamp |  | Timestamp when the appointment was created |

**Relationships:**

* **appointments.doctor\_id → user\_details.user\_id**
* **appointments.patient\_id → user\_details.user\_id**
* **appointments.availability\_id → availability.id**

3. availability

|  |  |  |  |
| --- | --- | --- | --- |
| **Column Name** | **Data Type** | **Key Type** | **Description** |
| id | int(11) | Primary | Unique identifier for availability |
| doctor\_id | int(11) | Foreign | References user\_details.user\_id |
| date | date |  | Date of availability |
| start\_time | time |  | Start time of availability |
| end\_time | time |  | End time of availability |
| is\_available | tinyint(1) |  | Availability status (1 for available) |

**Relationships:**

* **availability.doctor\_id → user\_details.user\_id**

4. availability\_patterns

|  |  |  |  |
| --- | --- | --- | --- |
| **Column Name** | **Data Type** | **Key Type** | **Description** |
| id | int(11) | Primary | Unique identifier for pattern |
| doctor\_id | int(11) | Foreign | References user\_details.user\_id |
| pattern\_name | varchar(255) |  | Name of the availability pattern |
| days\_of\_week | varchar(255) |  | Days of the week pattern applies |
| start\_time | time |  | Start time for the pattern |
| end\_time | time |  | End time for the pattern |
| start\_date | date |  | Start date for the pattern |
| end\_date | date |  | End date for the pattern (optional) |

**Relationships:**

* **availability\_patterns.doctor\_id → user\_details.user\_id**

5. availability\_templates

|  |  |  |  |
| --- | --- | --- | --- |
| **Column Name** | **Data Type** | **Key Type** | **Description** |
| id | int(11) | Primary | Unique identifier for template |
| doctor\_id | int(11) | Foreign | References user\_details.user\_id |
| template\_name | varchar(255) |  | Name of the availability template |
| days\_of\_week | varchar(255) |  | Days of the week template applies |
| start\_time | time |  | Start time for the template |
| end\_time | time |  | End time for the template |

**Relationships:**

* **availability\_templates.doctor\_id → user\_details.user\_i****d**

6. charges

|  |  |  |  |
| --- | --- | --- | --- |
| **Column Name** | **Data Type** | **Key Type** | **Description** |
| id | int(11) | Primary | Unique identifier for each charge |
| patient\_id | int(11) | Foreign | References users.id (patient) |
| doctor\_id | int(11) | Foreign | References users.id (doctor) |
| complaint\_id | int(11) | Foreign | References complaints.id |
| description | varchar(255) |  | Description of the charge |
| amount | decimal(10,2) |  | Amount charged |
| status | enum |  | Payment status (pending, paid) |
| payment\_method | varchar(50) |  | Method of payment |

**Relationships:**

* **charges.patient\_id → users.id**
* **charges.doctor\_id → users.id**
* **charges.complaint\_id → complaints.id**

7. clinics

|  |  |  |  |
| --- | --- | --- | --- |
| **Column Name** | **Data Type** | **Key Type** | **Description** |
| id | int(11) | Primary | Unique identifier for each clinic |
| clinic\_name | varchar(255) |  | Name of the clinic |
| clinic\_address | text |  | Address of the clinic |
| clinic\_contact | varchar(50) |  | Contact information for the clinic |
| doctor\_id | int(11) | Foreign | References users.id (doctor) |

**Relationships:**

* **clinics.doctor\_id → users.id**

8. complaints

|  |  |  |  |
| --- | --- | --- | --- |
| **Column Name** | **Data Type** | **Key Type** | **Description** |
| id | int(11) | Primary | Unique identifier for each complaint |
| user\_id | int(11) | Foreign | References users.id |
| complaint | text |  | Text of the complaint |
| date | timestamp |  | Date and time of complaint |
| doctor\_id | int(11) | Foreign | References users.id (optional) |
| response | text |  | Response to the complaint |
| status | varchar(50) |  | Status of the complaint (Open, Closed) |
| clinic\_details | text |  | Details of the clinic involved (optional) |
| prescribed\_items | text |  | Items prescribed during the consultation (optional) |

**Relationships:**

* **complaints.user\_id → users.id**
* **complaints.doctor\_id → users.id**

9. items

|  |  |  |  |
| --- | --- | --- | --- |
| **Column Name** | **Data Type** | **Key Type** | **Description** |
| id | int(11) | Primary | Unique identifier for each item |
| name | varchar(255) |  | Name of the item |
| cost | decimal(10,2) |  | Cost of the item |

**Relationships:**

* **No direct relationships (independent table).**

10. reports

|  |  |  |  |
| --- | --- | --- | --- |
| **Column Name** | **Data Type** | **Key Type** | **Description** |
| id | int(11) | Primary | Unique identifier for each report |
| patient\_id | int(11) | Foreign | References users.id |
| doctor\_id | int(11) | Foreign | References users.id |
| complaint\_id | int(11) | Foreign | References complaints.id |
| report | text |  | Text of the report |
| date | timestamp |  | Date and time of report creation |

**Relationships:**

* **reports.patient\_id → users.id**
* **reports.doctor\_id → users.id**
* **reports.complaint\_id → complaints.id**

11. reviews

|  |  |  |  |
| --- | --- | --- | --- |
| **Column Name** | **Data Type** | **Key Type** | **Description** |
| id | int(11) | Primary | Unique identifier for each review |
| patient\_id | int(11) | Foreign | References users.id (patient) |
| doctor\_id | int(11) | Foreign | References users.id (doctor) |
| rating | int(11) |  | Rating given by the patient (1-5) |
| comment | text |  | Review comment by the patient |
| created\_at | timestamp |  | Date and time when the review was created |

**Relationships:**

* **reviews.patient\_id → users.id**
* **reviews.doctor\_id → users.id**

12. roles

|  |  |  |  |
| --- | --- | --- | --- |
| **Column Name** | **Data Type** | **Key Type** | **Description** |
| id | int(11) | Primary | Unique identifier for each role |
| role\_name | varchar(50) |  | Name of the role (e.g., admin, doctor, patient) |

**Relationships:**

* **Referenced by the users table.**

13. users

|  |  |  |  |
| --- | --- | --- | --- |
| **Column Name** | **Data Type** | **Key Type** | **Description** |
| id | int(11) | Primary | Unique identifier for each user |
| userId | varchar(50) |  | User ID for login |
| password | varchar(255) |  | Password for login |
| role\_id | int(11) | Foreign | References roles.id |
| created\_at | timestamp |  | Date and time when the user was created |

**Relationships:**

* **users.role\_id → roles.id**

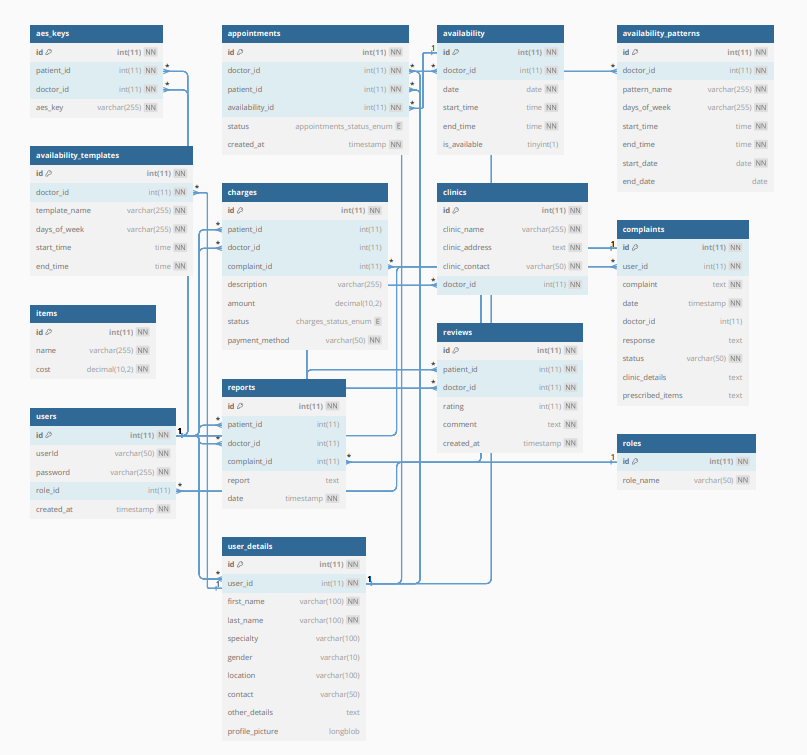
14. user\_details

|  |  |  |  |
| --- | --- | --- | --- |
| **Column Name** | **Data Type** | **Key Type** | **Description** |
| id | int(11) | Primary | Unique identifier for each user detail |
| user\_id | int(11) | Foreign | References users.id |
| first\_name | varchar(100) |  | First name of the user |
| last\_name | varchar(100) |  | Last name of the user |
| specialty | varchar(100) |  | Specialty of the doctor (optional) |
| gender | varchar(10) |  | Gender of the user (optional) |
| location | varchar(100) |  | Location of the user |
| contact | varchar(50) |  | Contact information of the user |
| other\_details | text |  | Other details about the user (optional) |
| profile\_picture | longblob |  | Profile picture of the user (optional) |

**Relationships:**

* **user\_details.user\_id → users.id**

#### **Entity-Relationship Diagram (ERD)**

****

#### **Database Design Tools Used**

**phpMyAdmin:**

* Database Management: For general administration of the database and in particular every day the default web application phpMyAdmin was used to import or export data, manage users’ access rights, and run general SQL queries on the MySQL server. It was web based with a user-friendly graphical front end thus could easily be modified or updated as required.
* Data Migration: phpMyAdmin was particularly very helpful in copying data from one environment to the other like the development, staging and production databases.

**DBDiagram.io:**

* **Visual Documentation:** DBDiagram.io was utilised in designing ERDs in basic and easy to share formats to the other stakeholders. This online tool enabled drawing the diagram based upon the textual description of the database schema. It lets the users present the database structure in the simplest form for better understanding of other team members or stakeholders who may want to be part of the process.
* **Collaborative Design:** The tool also facilitated collaboration where various users could work on the same ERD and review the changes in real time making the design more of an open and cyclical process.

### **7. User Interface Design**

User Interface Design is responsible for creating user-friendly system management interfaces. The main idea is to increase user interaction with the system and ensure that the experience is consistent.

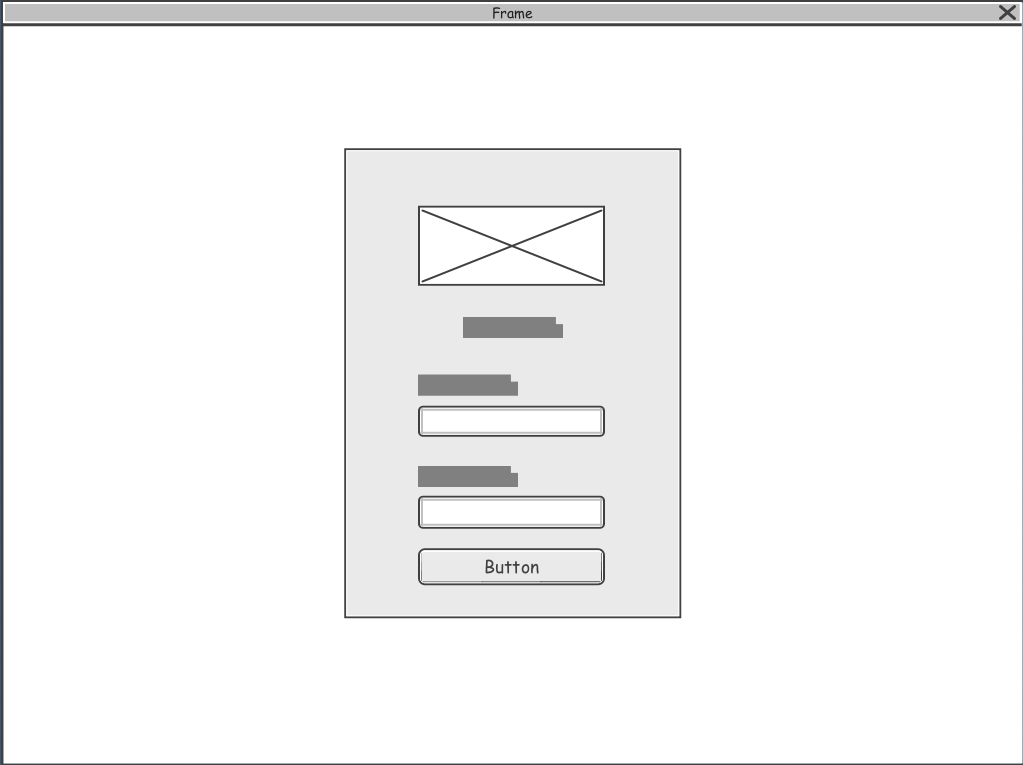
#### **7.1** **UI Design Principles**

UI Design principles are the guidelines to follow to ensure that your system will be effective and a high standard user experience. Here are the 6 main principles that you should keep an eye on:

* **Clarity:** The first principle manages if the interface is easy to understand by users. It means, users should be able to understand how the system works just by looking at it. The system may be filled by clear language, easy to read fonts in a good size, intuitive icons and the most important is to have a simple layout.
* **Consistency:** The second principle is in charge of managing the elements on the interface. It means, users should be able to predict what will happen while they are interacting with the whole system. The AHMS may have a consistent colour scheme, typography and style beyond all screens.
* **Feedback:** The third principle conducts users feedback to improve the system based on their expectations. It may consist in using different types of animations, such as, sounds, different colours or alerts to indicate the outcomes from their actions.
* **Affordance:** The fourth principle suggests that elements must meet their function, for example, a button needs to look like a button (clickable).
* **Hierarchy:** The fifth principle organises elements on the screen, making sure that the system is getting the user's attention to what is important. Using font size, weight, positioning on the right spot to guide user focus.
* **Accessibility:** The sixthprinciple runs to ensure the interface is easy to be used by everyone including people with disabilities, granting no one is left behind.
* **Simplicity:** The seventh principle looks after minimising the redundant elements in the system. It means that the developers should focus on what is essential to be in the system, having a minimalist design and whitespaces will ensure the system is efficient.
* **Responsiveness:** The eighth principle refers to the UI being responsive, it means, it should adapt to different screen sizes, such as, laptops, smartphones, tablets, and others.
* **Efficiency:** The ninth principle says the UI should be able to allow users to do their tasks without too much effort.
* **Aesthetic:** The tenth principle assures aesthetic design increases user satisfaction.
* **User Control:** The eleventh principle consists in granting users control about their actions, like, navigate easily through the system and undo actions.
* **Error Prevention and Recovery:** The twelfth principle manages to prevent and minimise errors and be ready to recover in case anything bad happens.

**7.2 Wireframes**

The main purpose of wireframes is to design the process and visualise the interface based on stakeholders’ expectations and needs. Here are the wireframes of Austin Hospital Management System:

Fig. Login Wireframe

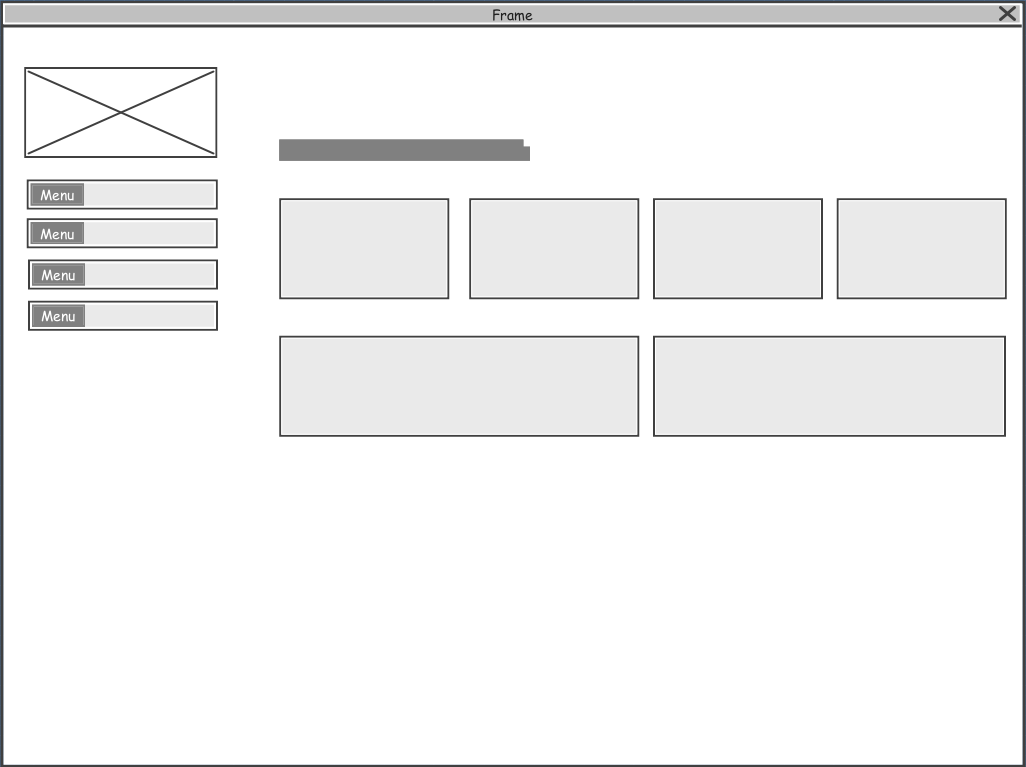


Fig. Admin Dashboard Wireframe

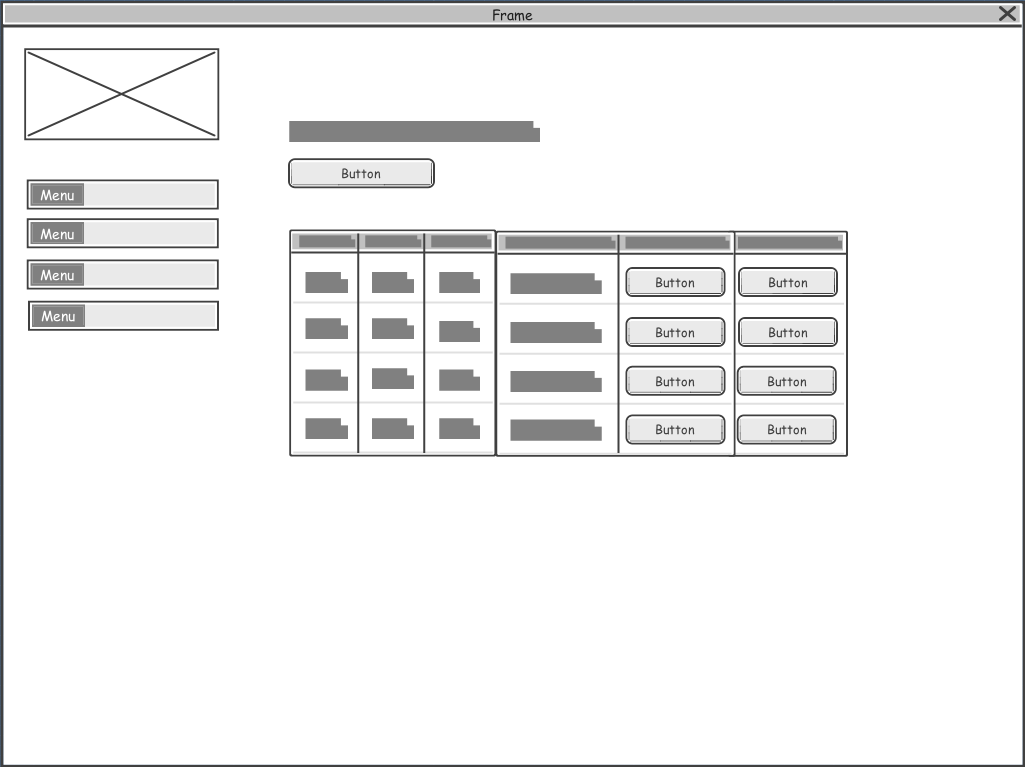


Fig. Admin Users Wireframe

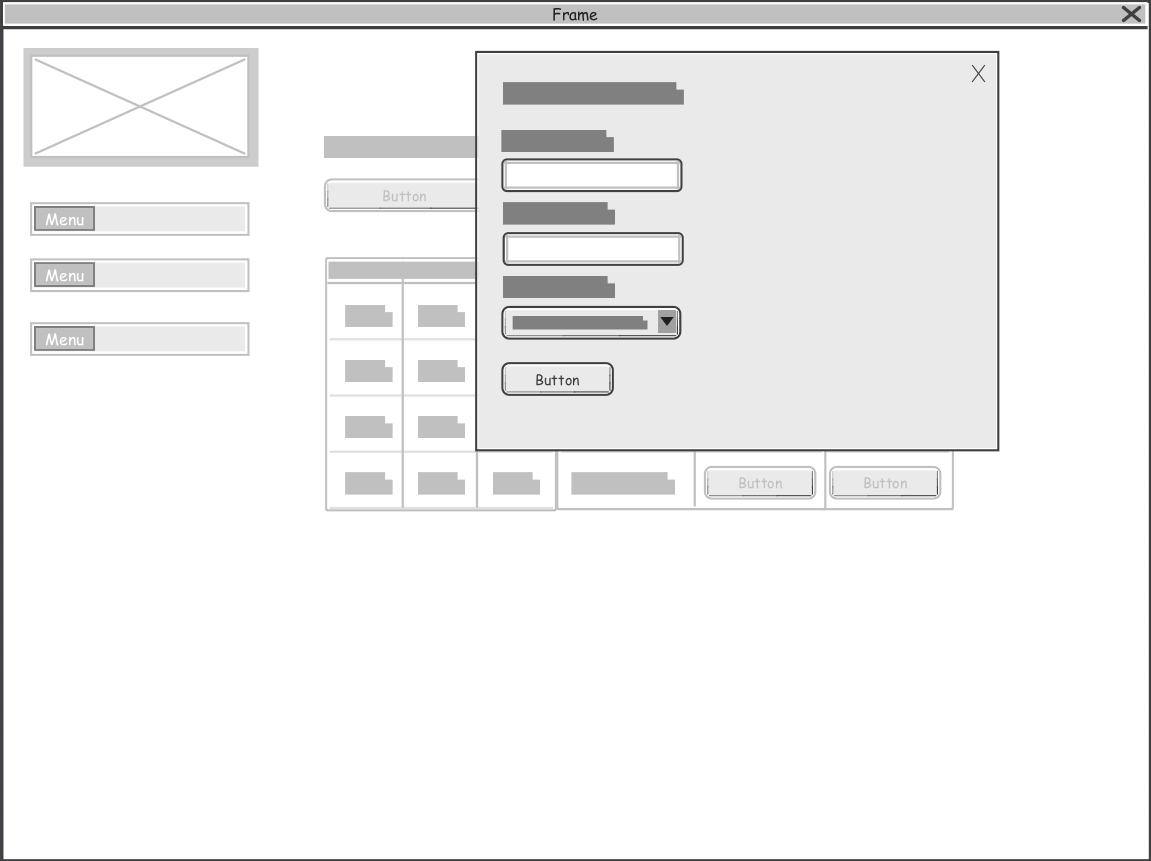


Fig. Admin Create New User Wireframe

Fig. Admin Update Details Wireframe

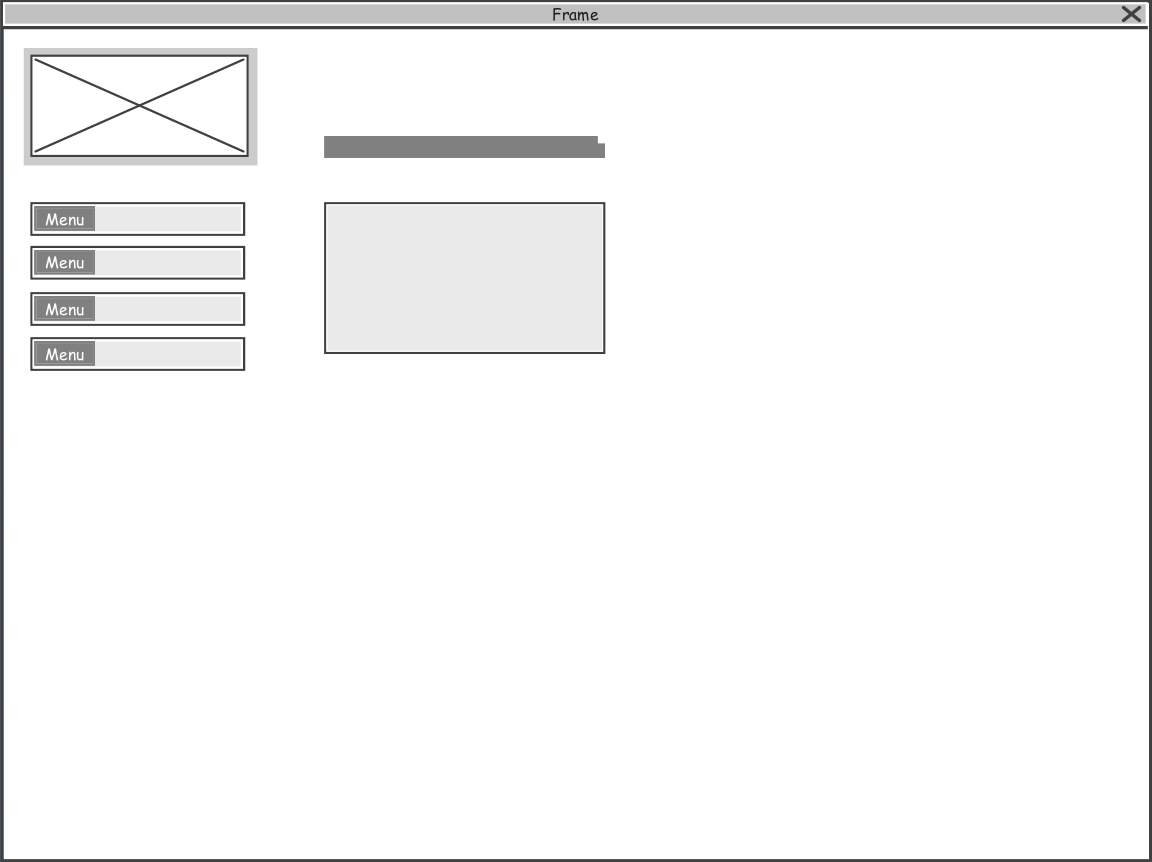


Fig. Clinic Dashboard Wireframe

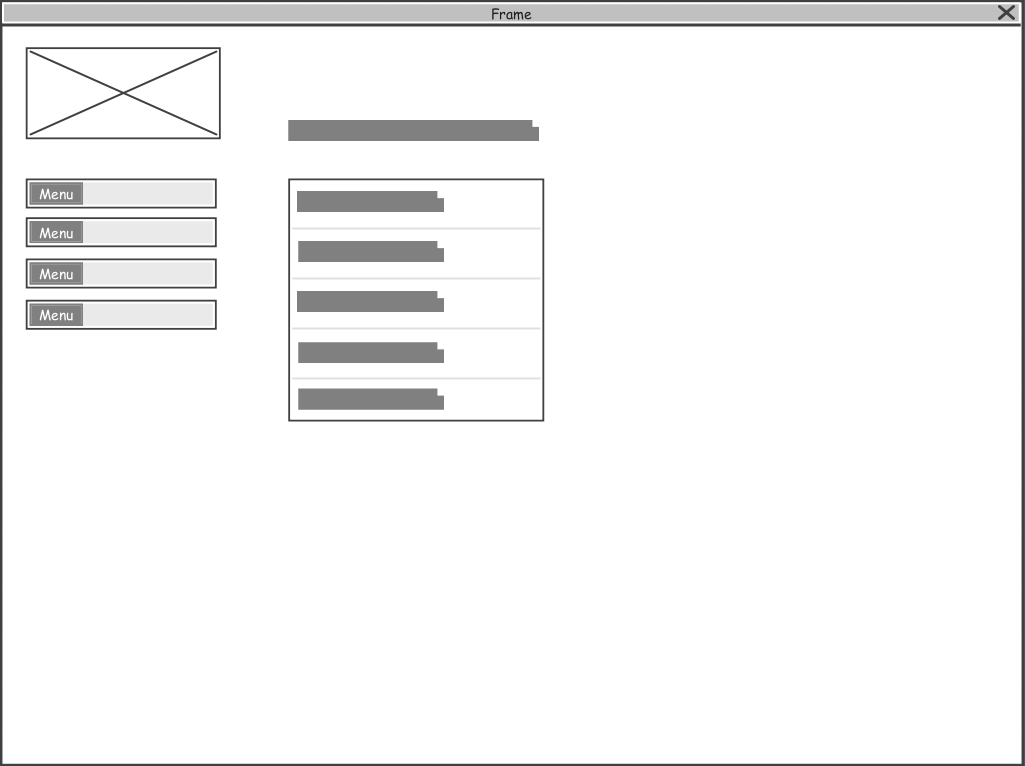


Fig. Clinic View Reports Wireframe

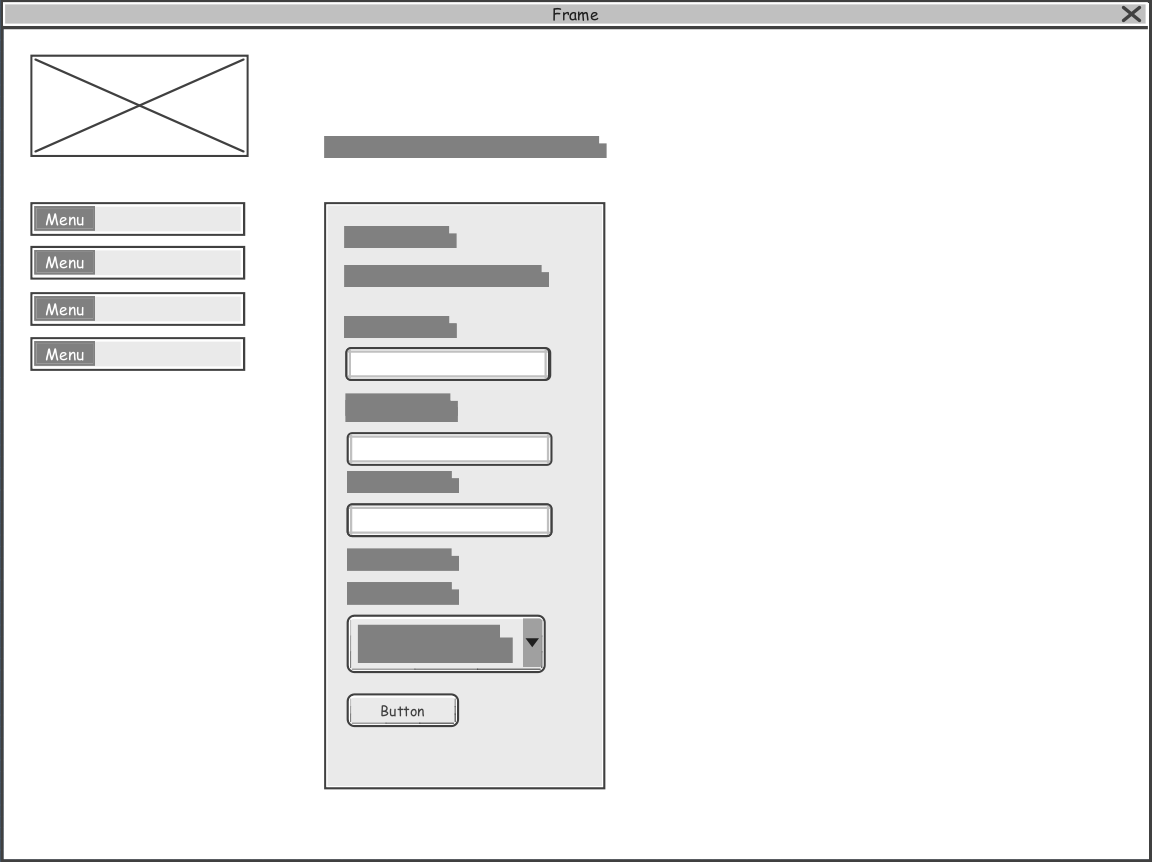


Fig. Clinic Update Report Details Wireframe

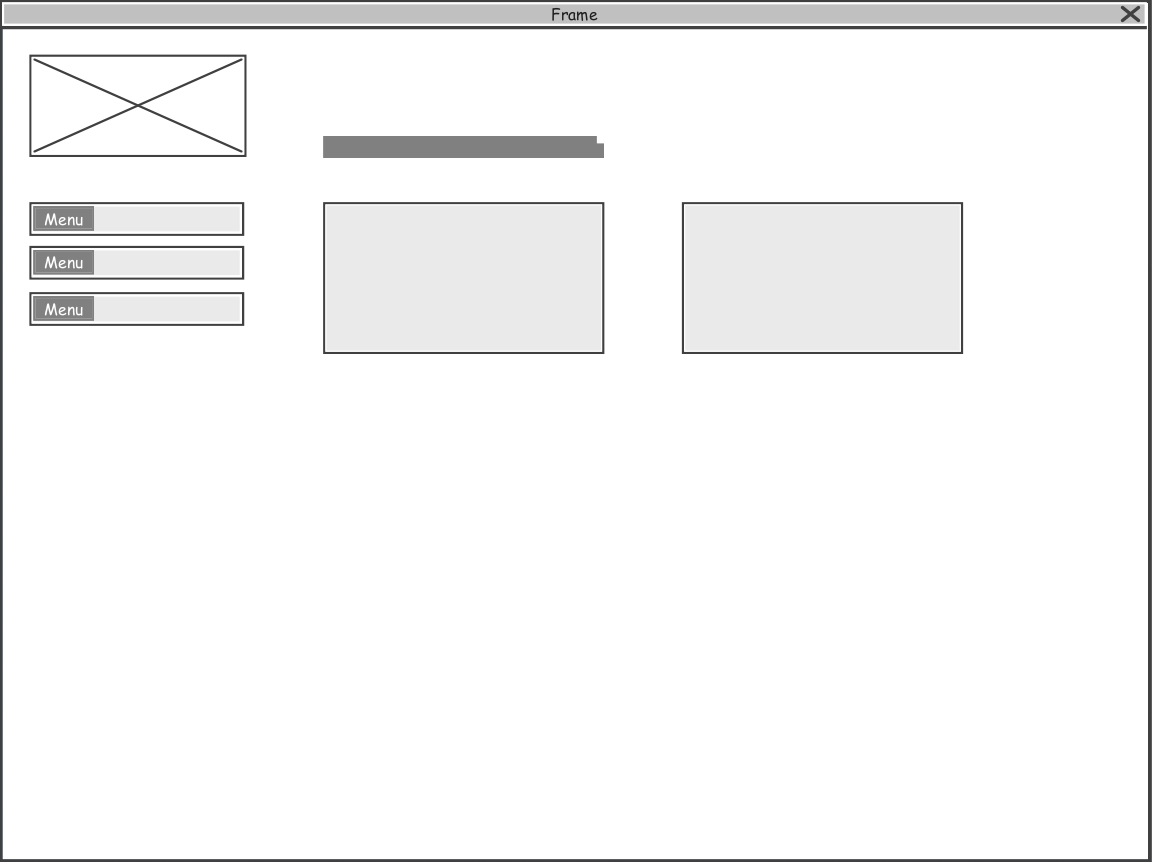


Fig. Cashier Dashboard Wireframe

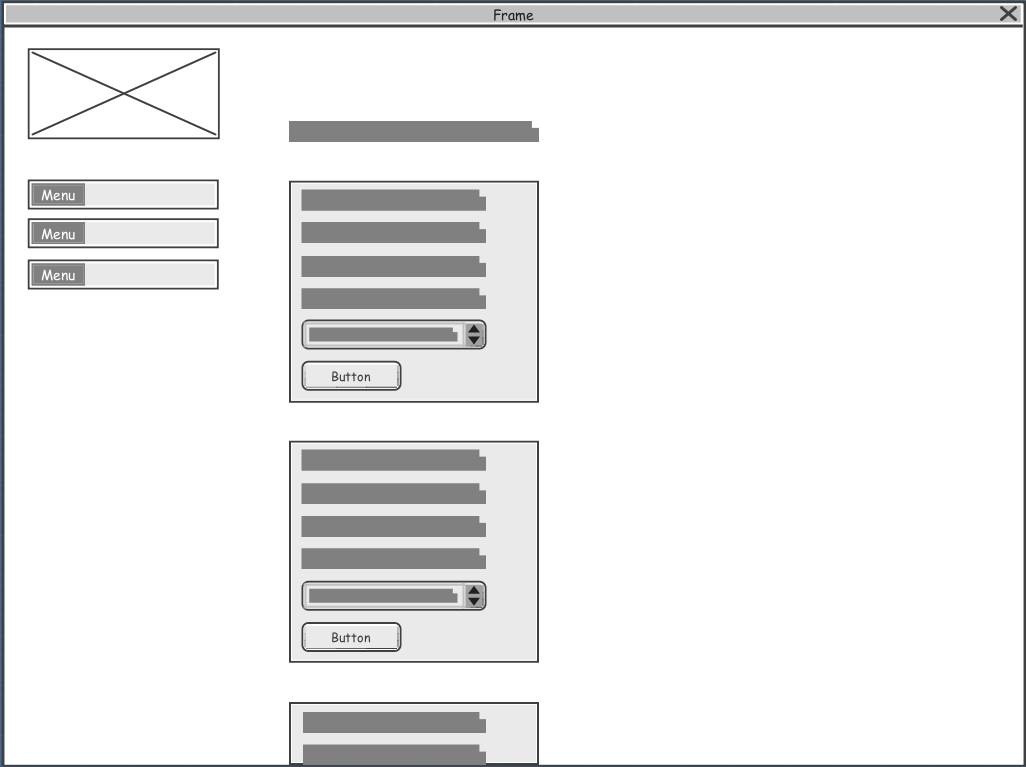


Fig. Cashier Charge Patients Wireframe

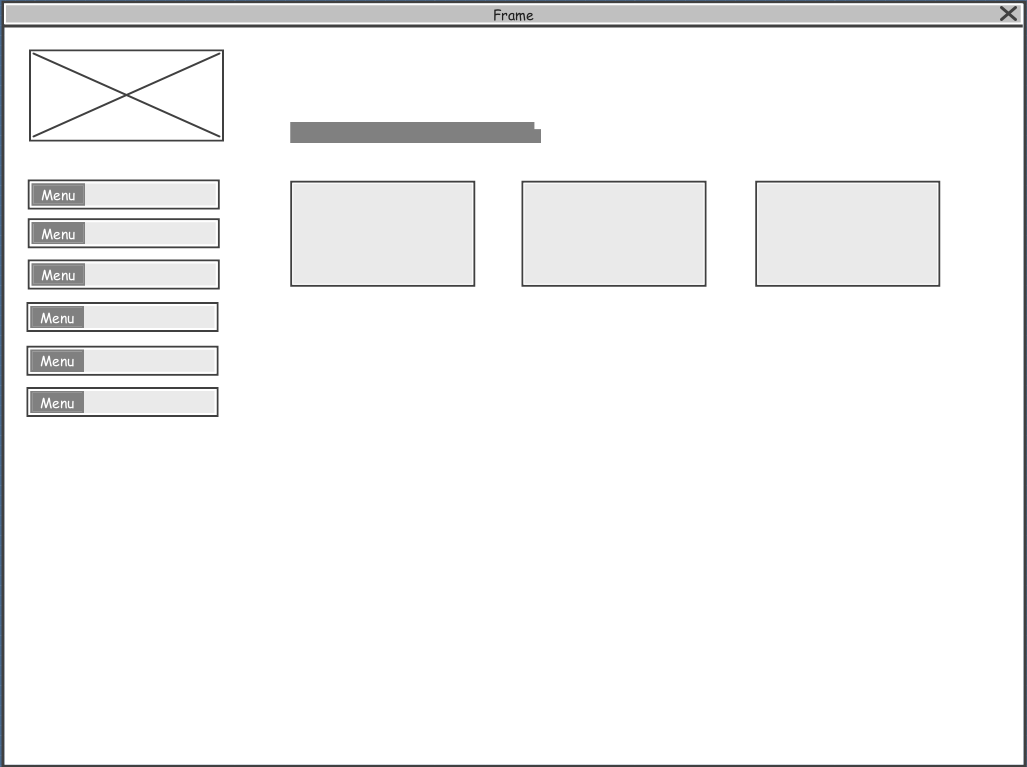


Fig. Doctor Dashboard Wireframe

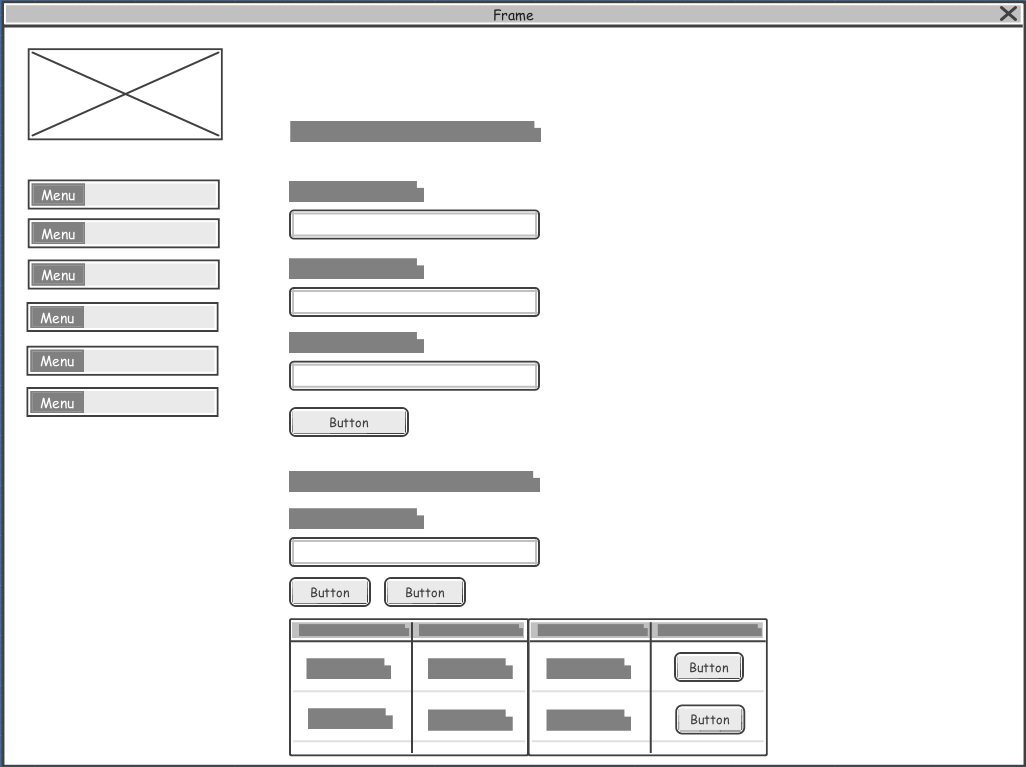


Fig. Doctor Clinic Register Wireframe

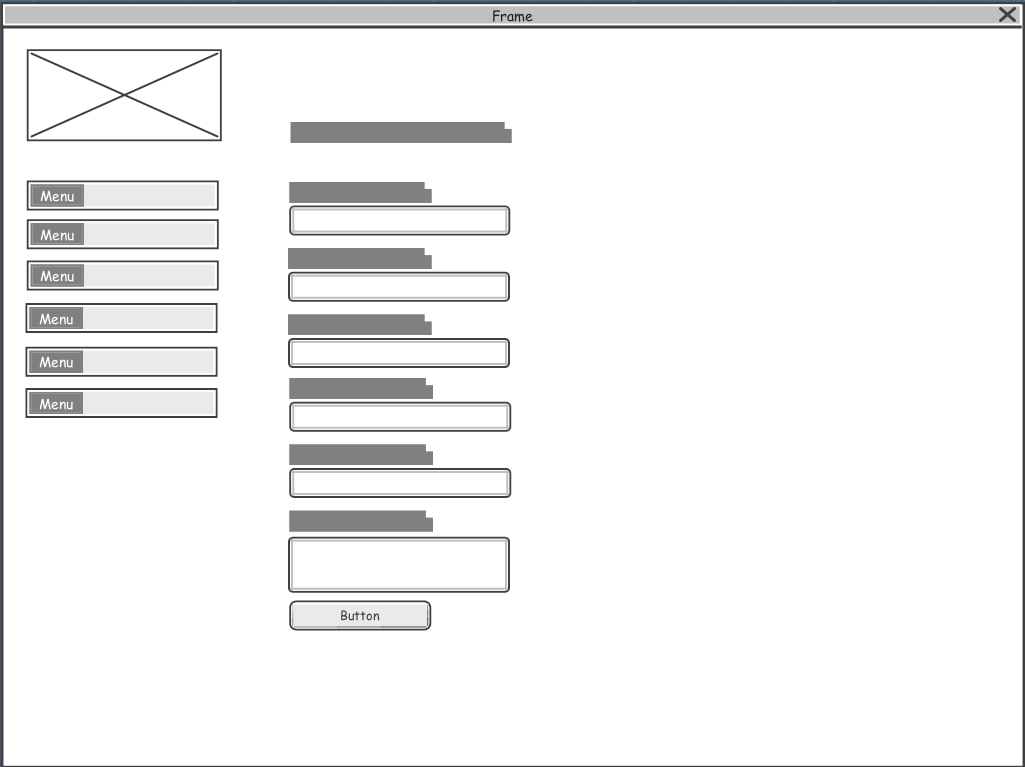


Fig. Doctor Update Details Wireframe

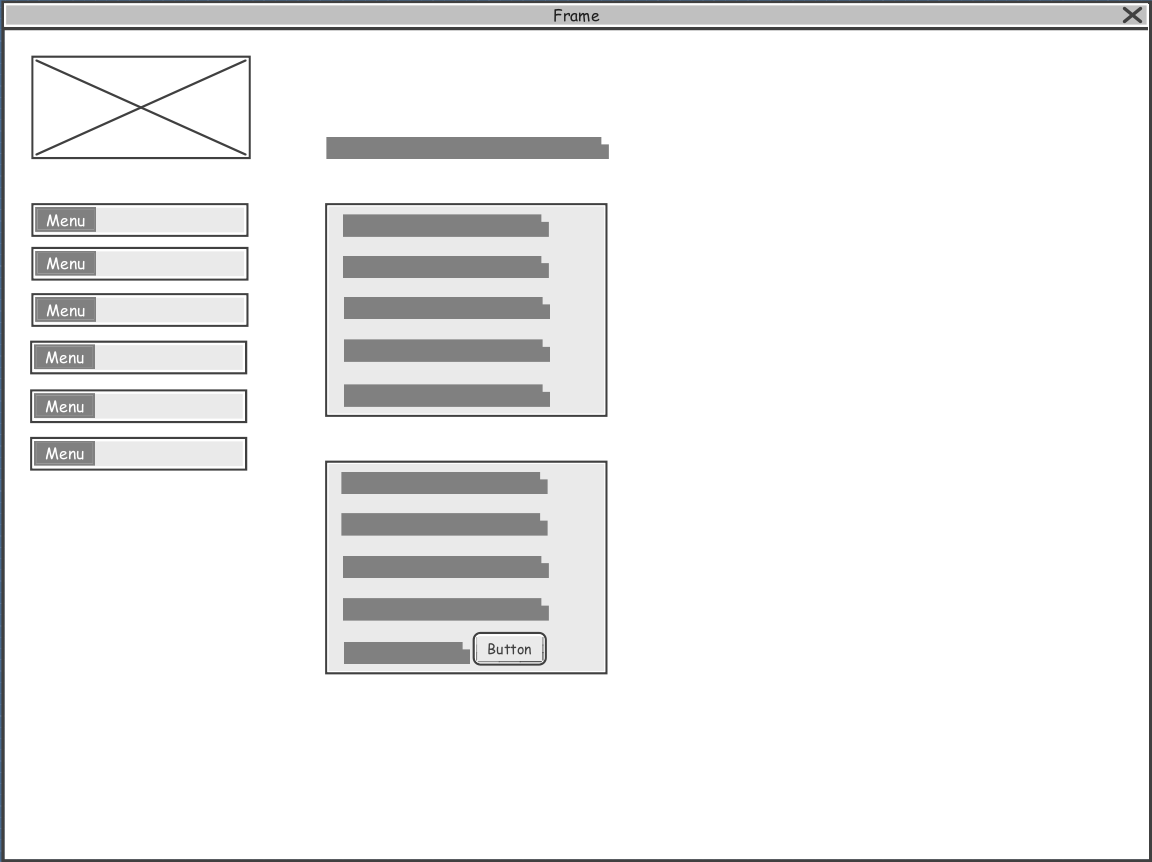


Fig. Doctor View Complaint Wireframe

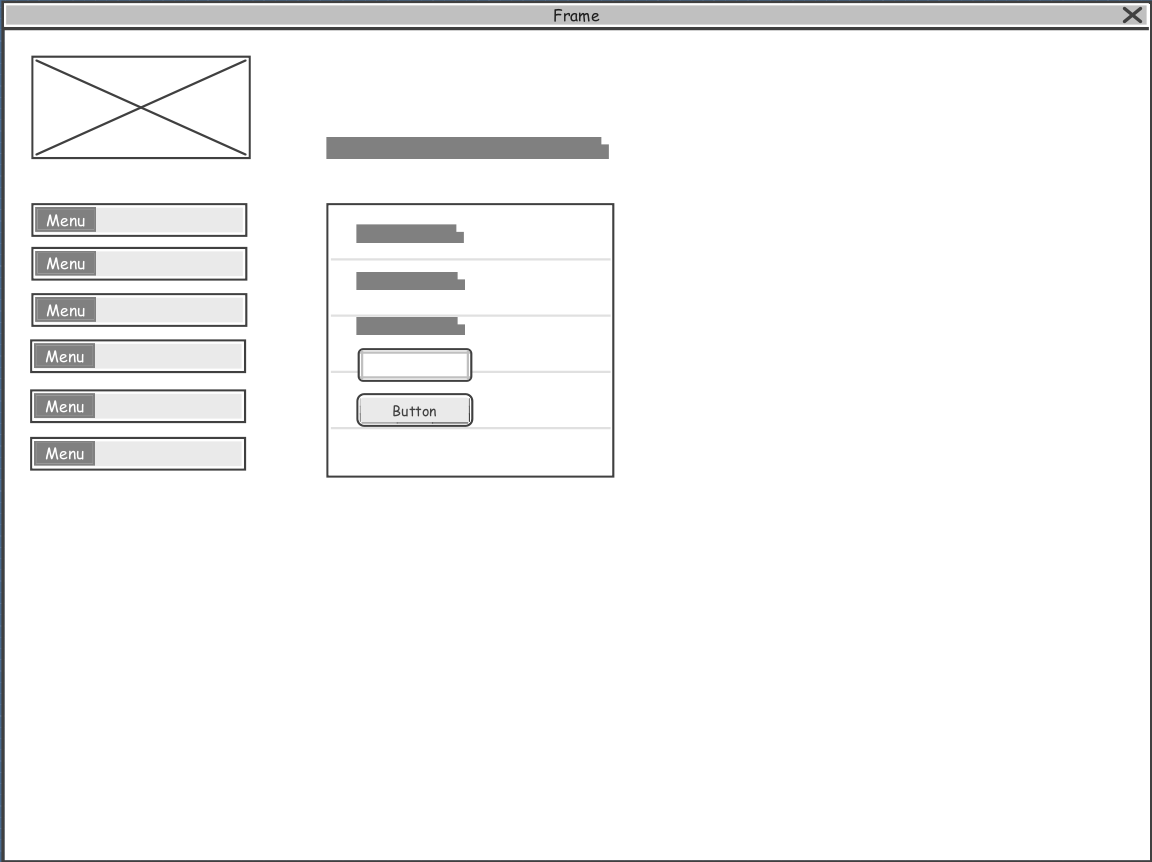


Fig. Doctor View AES Key Sharing Wireframe

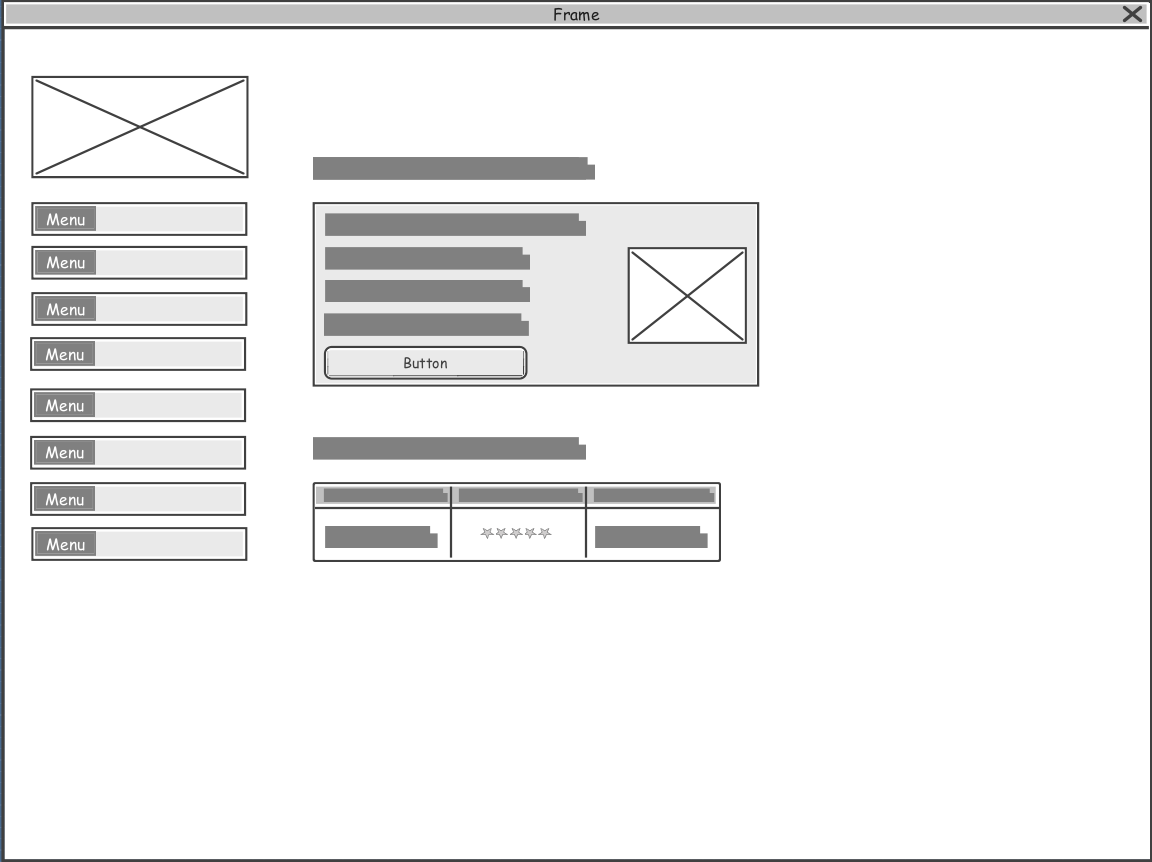


Fig. Doctor Profile Wireframe

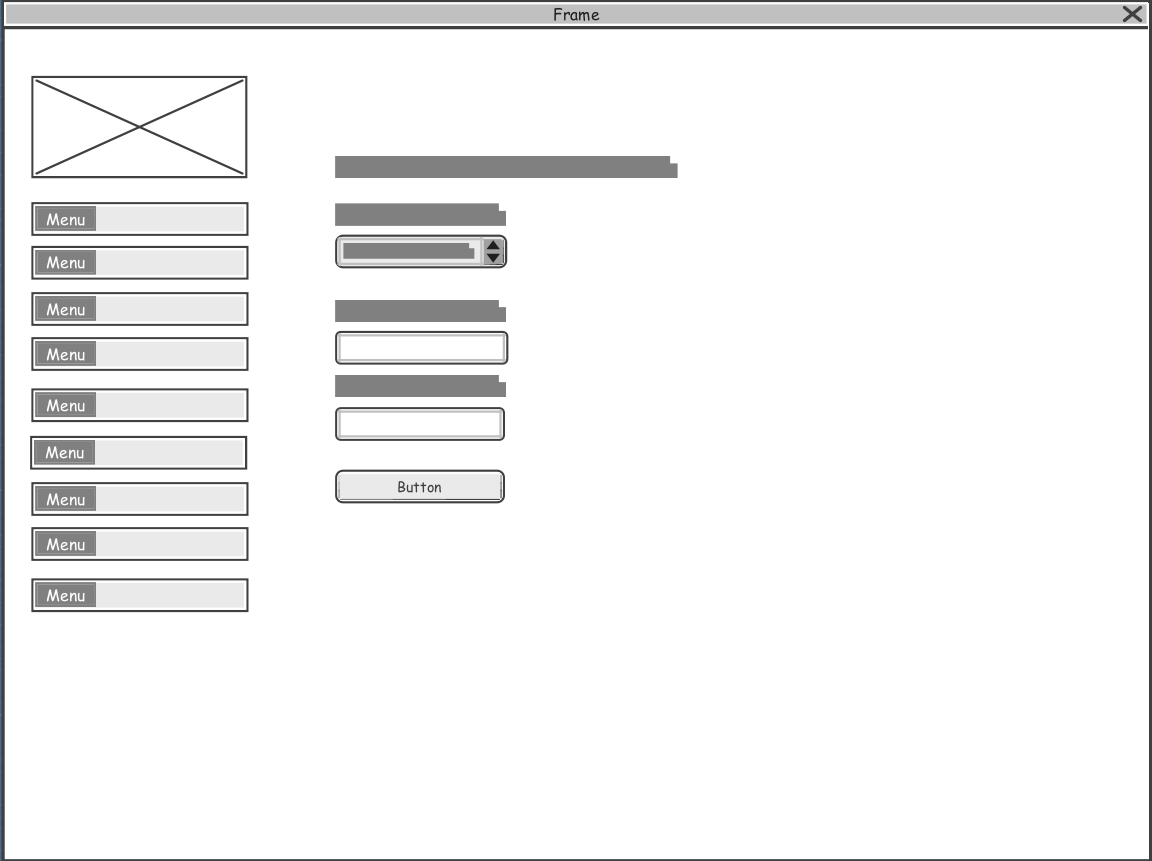


Fig. Doctor Apply Availability Wireframe

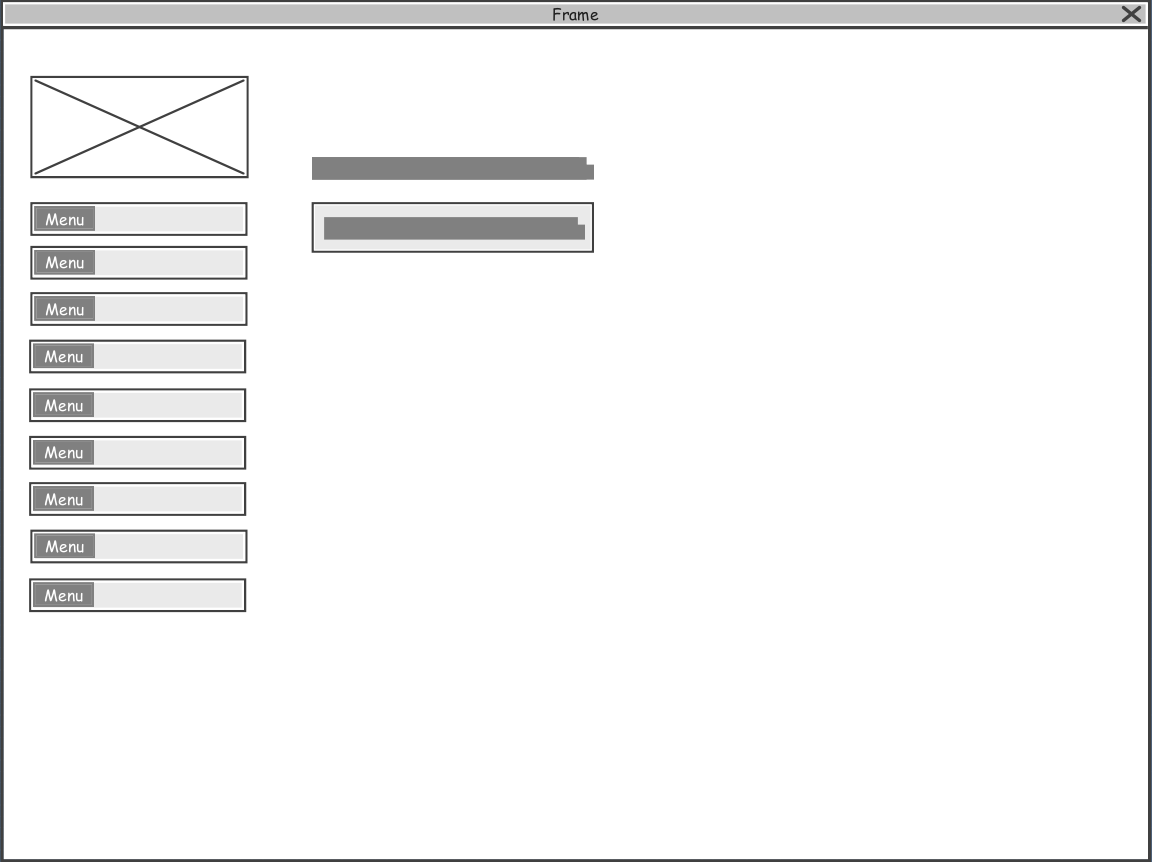


Fig. Doctor Appointments Wireframe

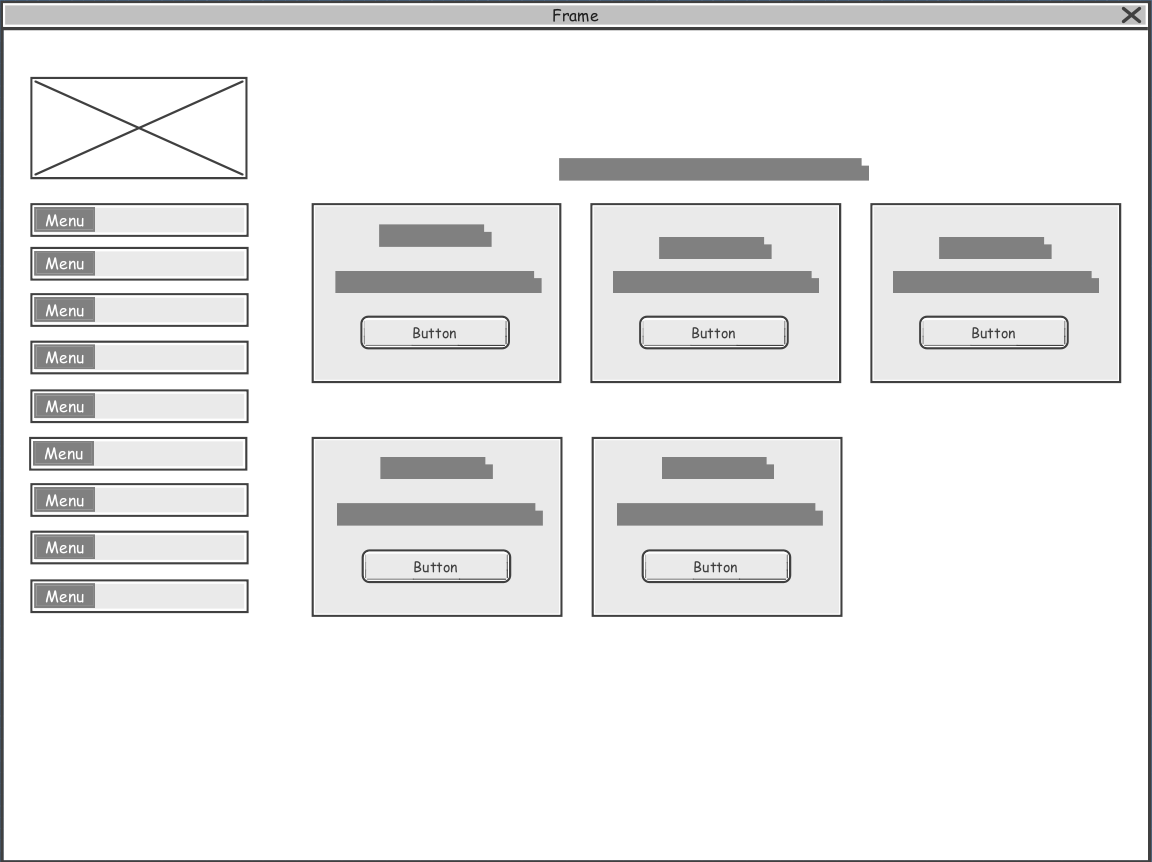


Fig. Doctor Availability Panel Wireframe

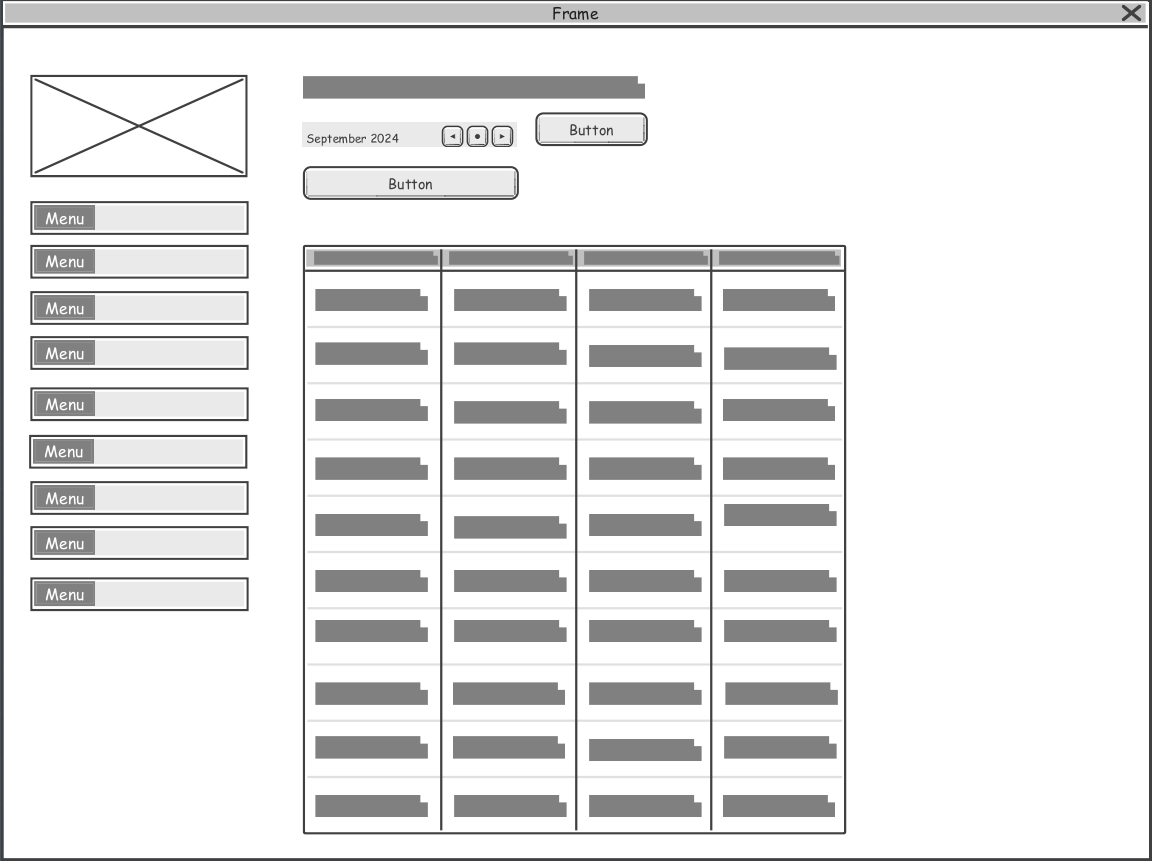


Fig. Doctor Check Availability Wireframe

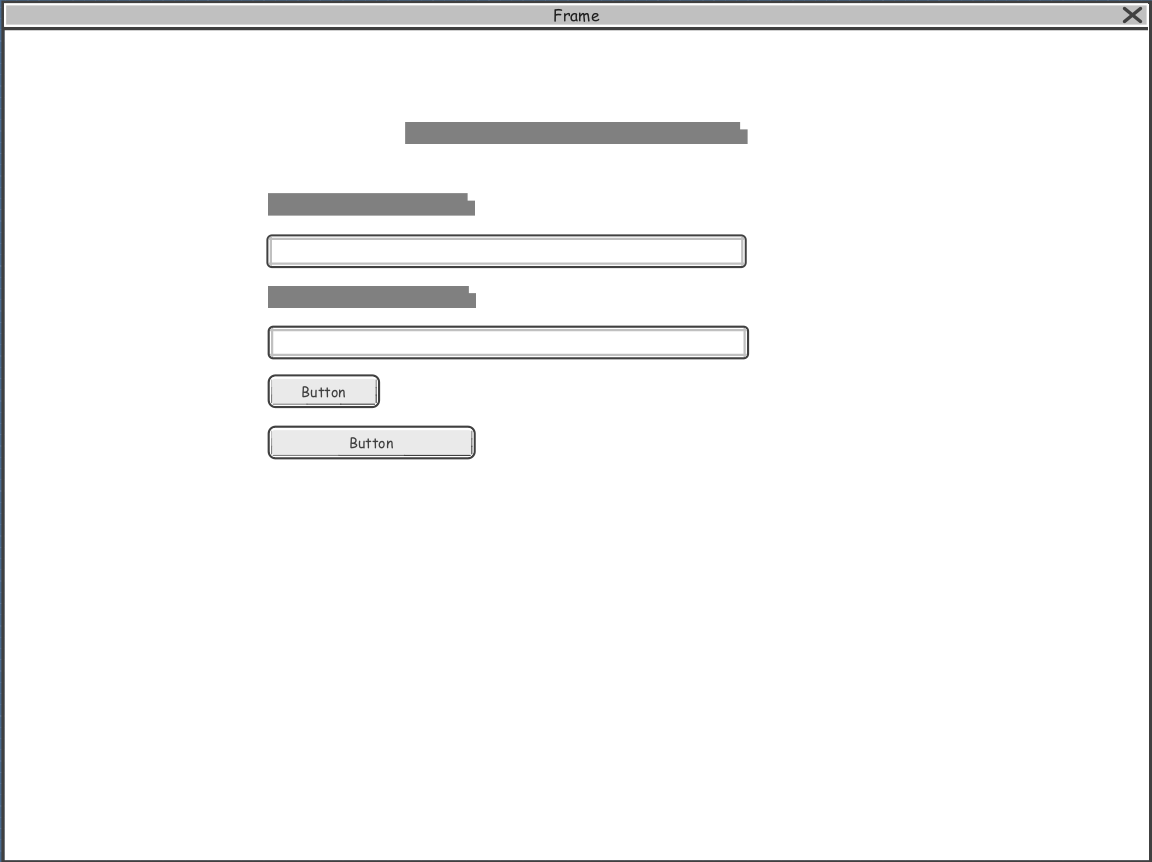


Fig. Doctor Clear Availability Wireframe

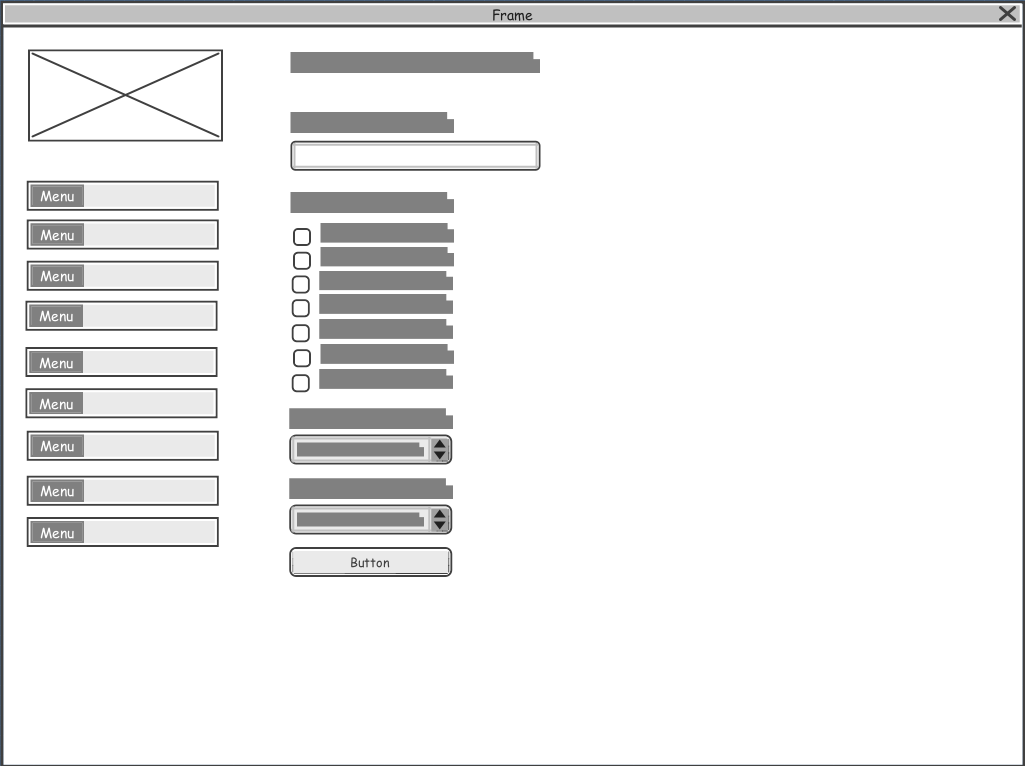


Fig. Doctor Create Availability Template Wireframe

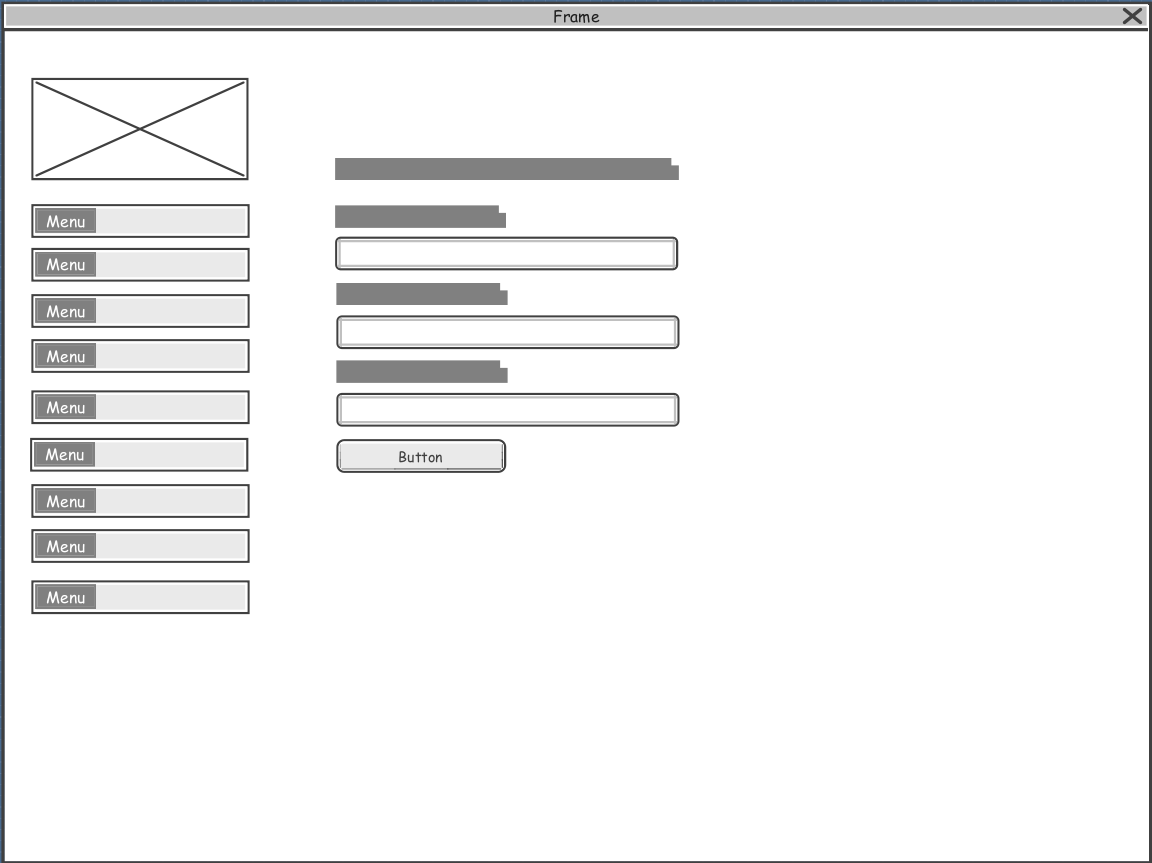


Fig. Doctor Set Specific Availability Wireframe

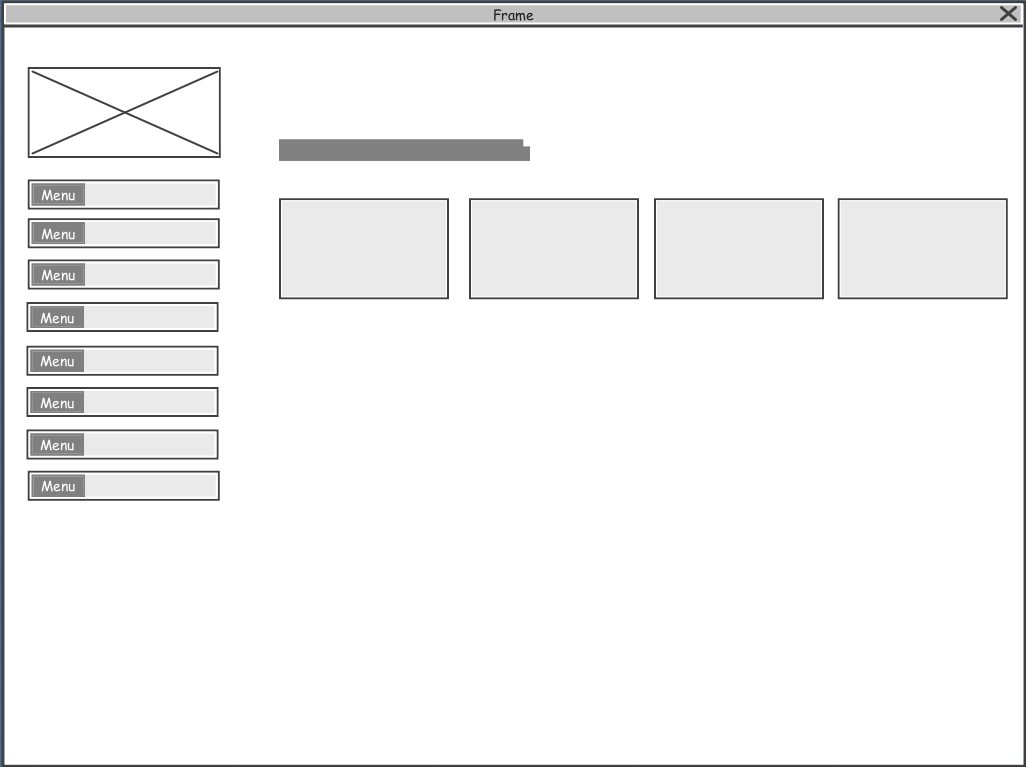


Fig. Patient Dashboard Wireframe

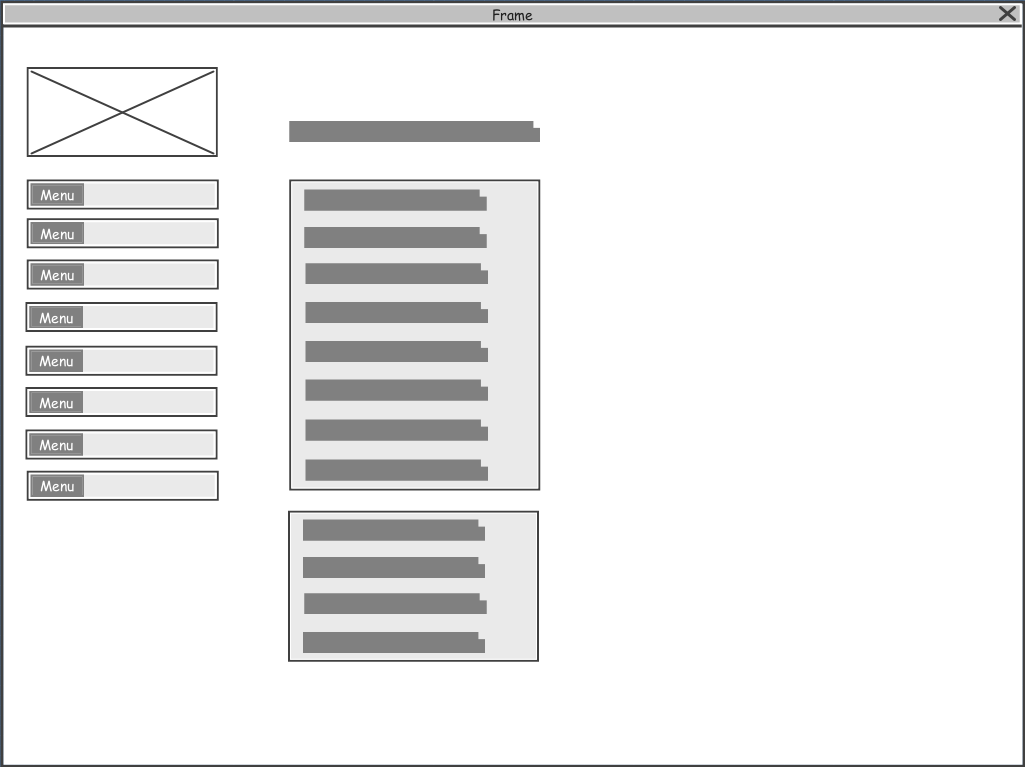


Fig. Patient View Profile Wireframe

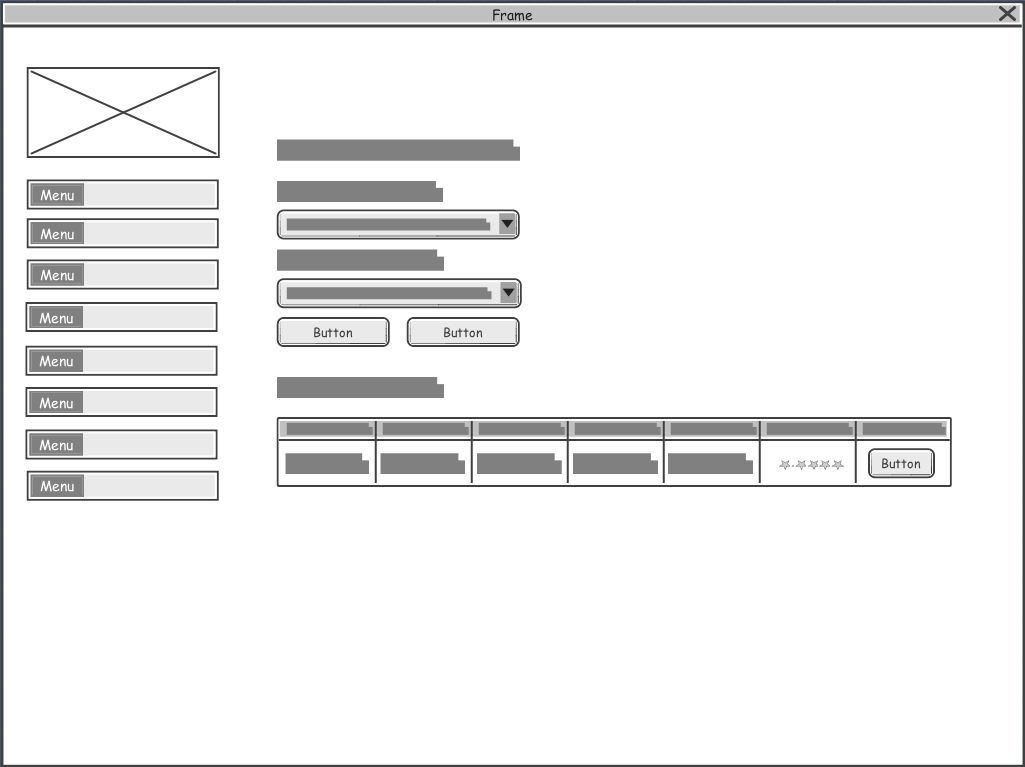


Fig. Patient Find Doctor Wireframe

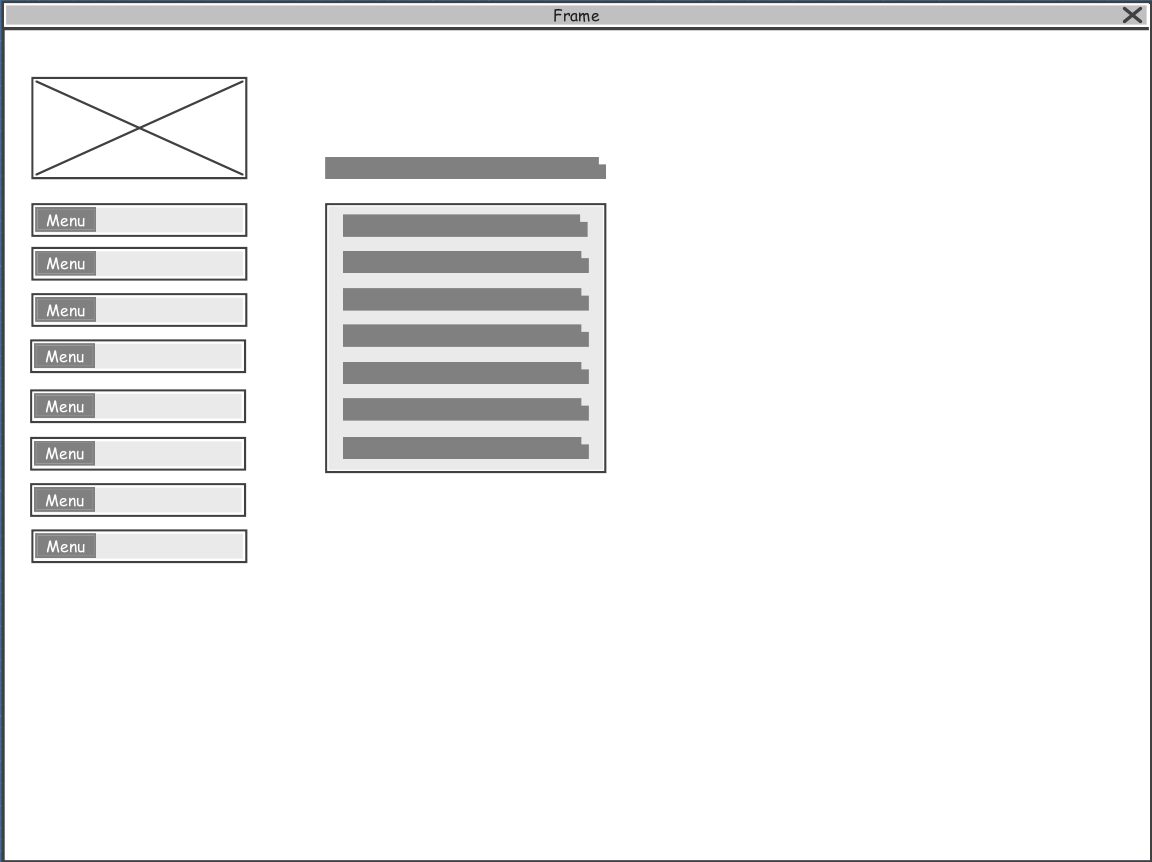


Fig. Patient View Reports Wireframe

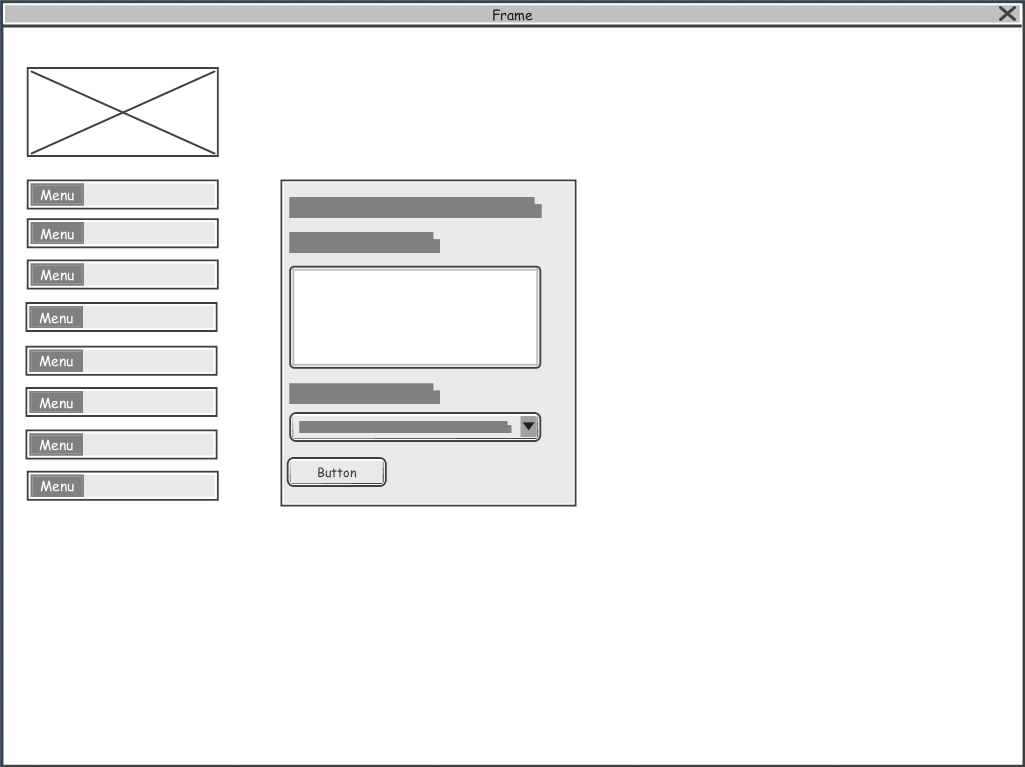


Fig. Patient Post Complaint Wireframe

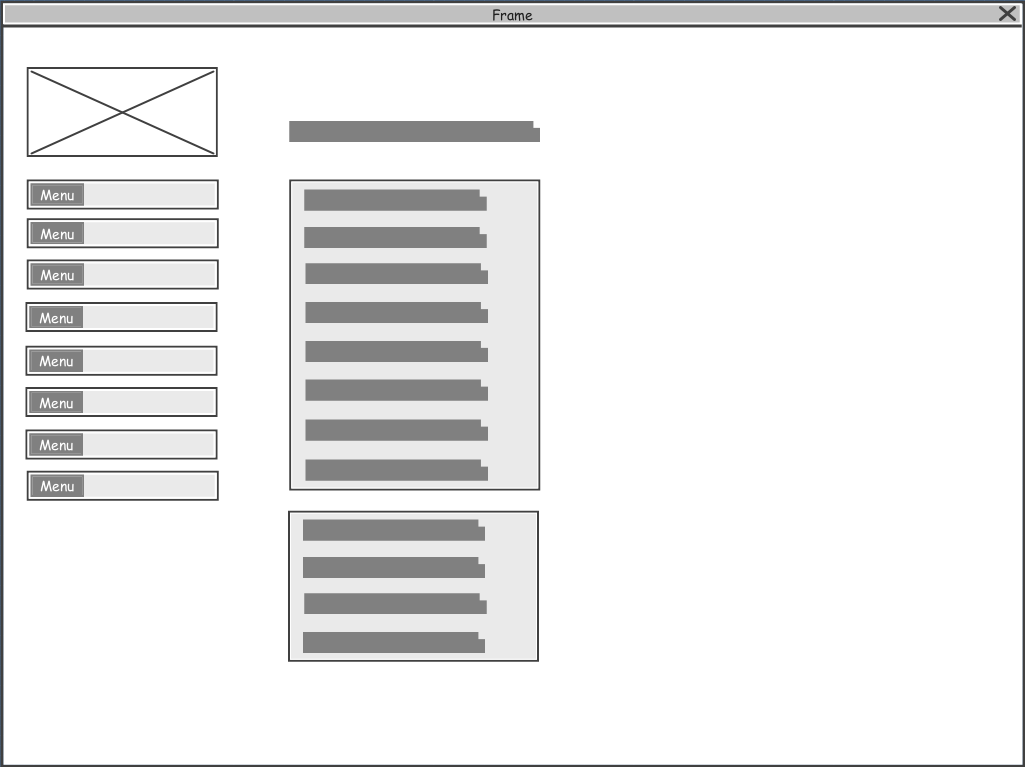


Fig. Patient View Complaint Wireframe

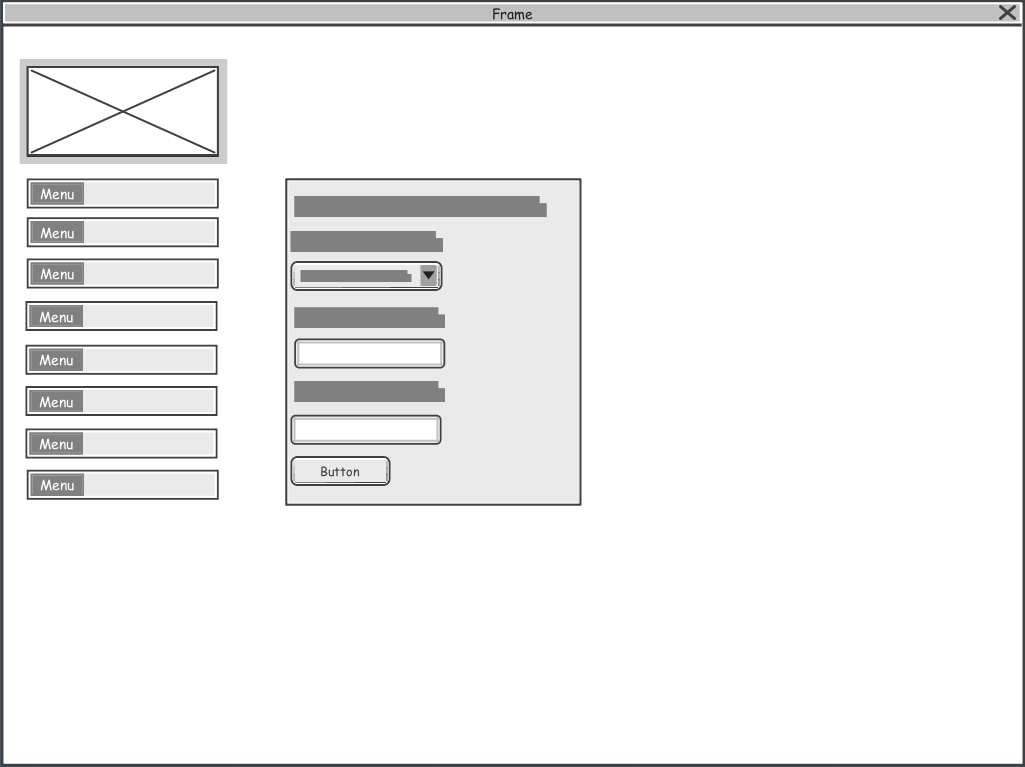


Fig. Patient Post AES Key Sharing Wireframe

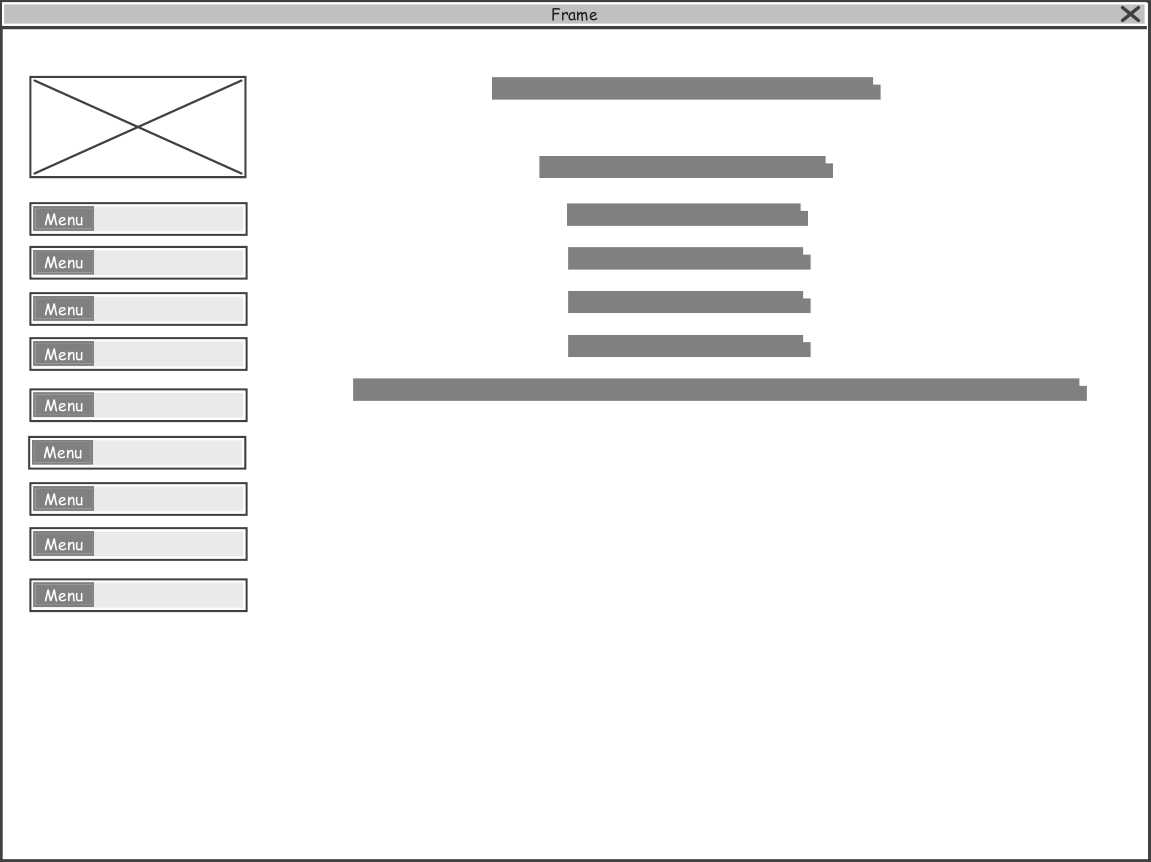


Fig. Patient Appointment Booked Successfully Wireframe

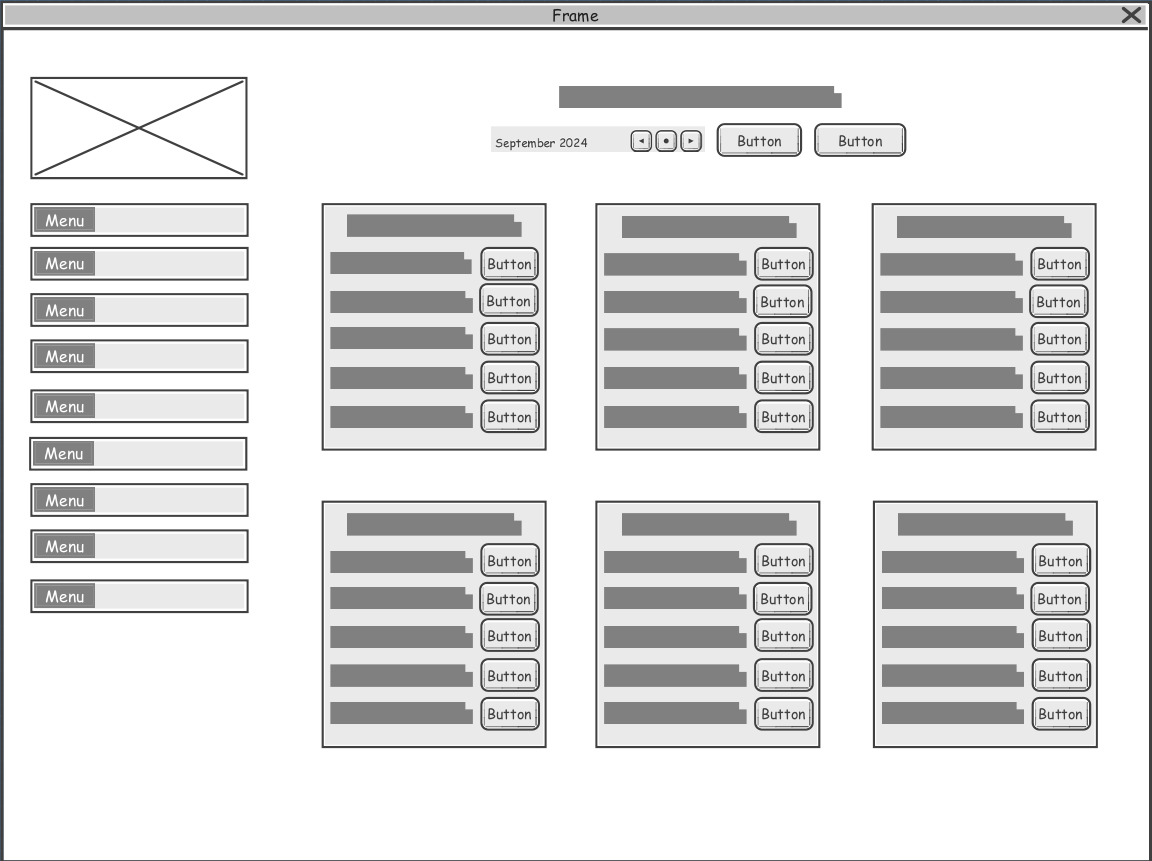


Fig. Patient Check Doctor Availability Wireframe

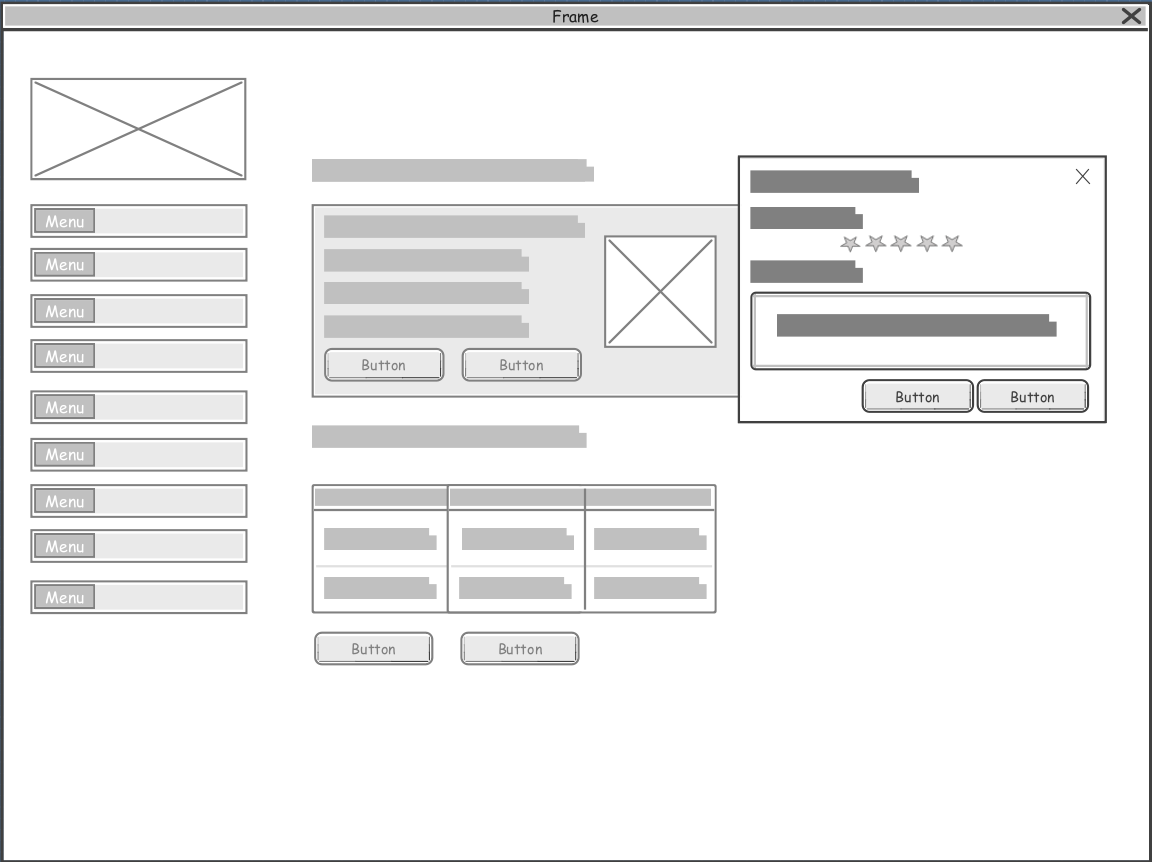


Fig. Patient Leave Review Wireframe

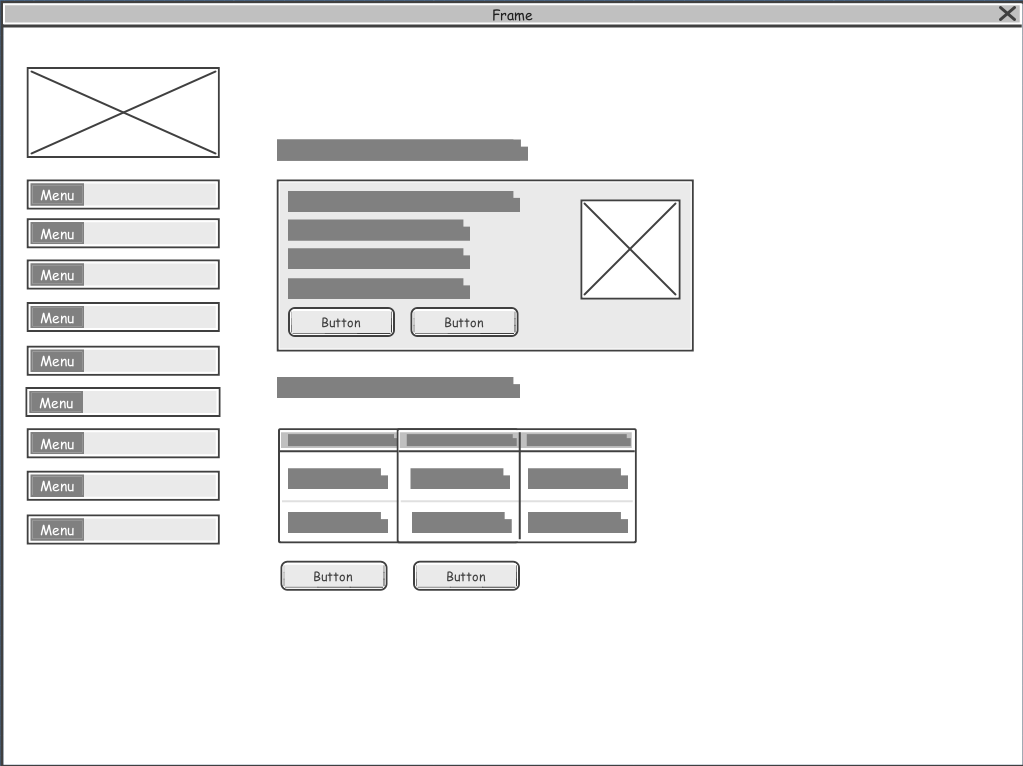


Fig. Patient See Doctor Profile Wireframe

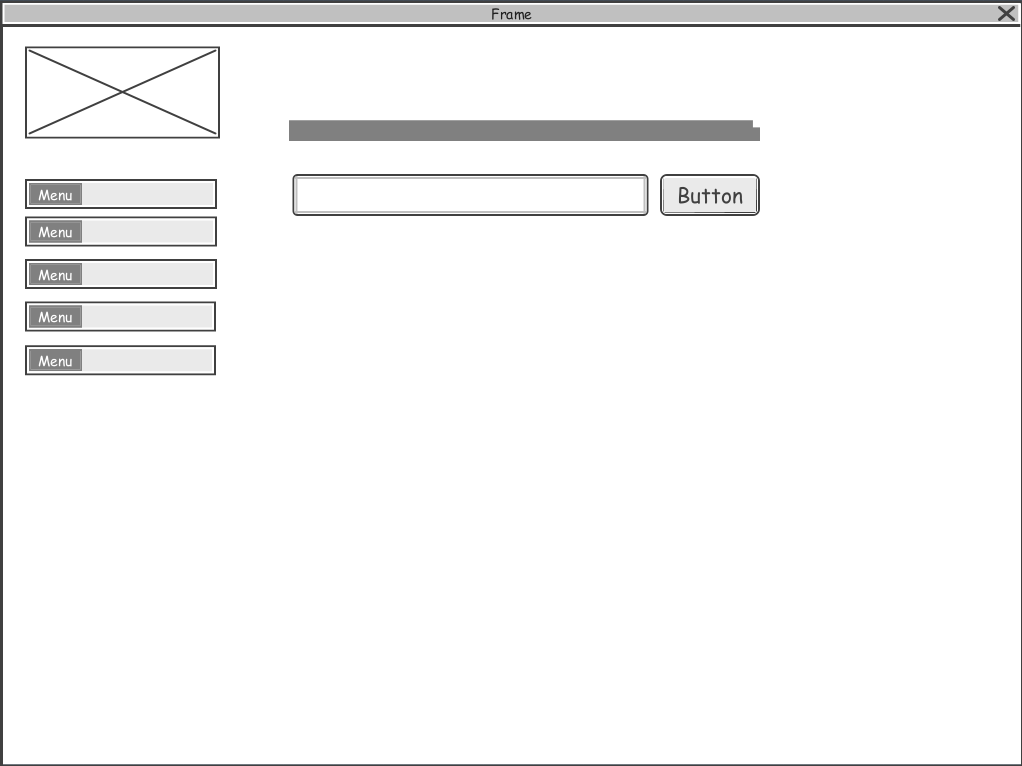
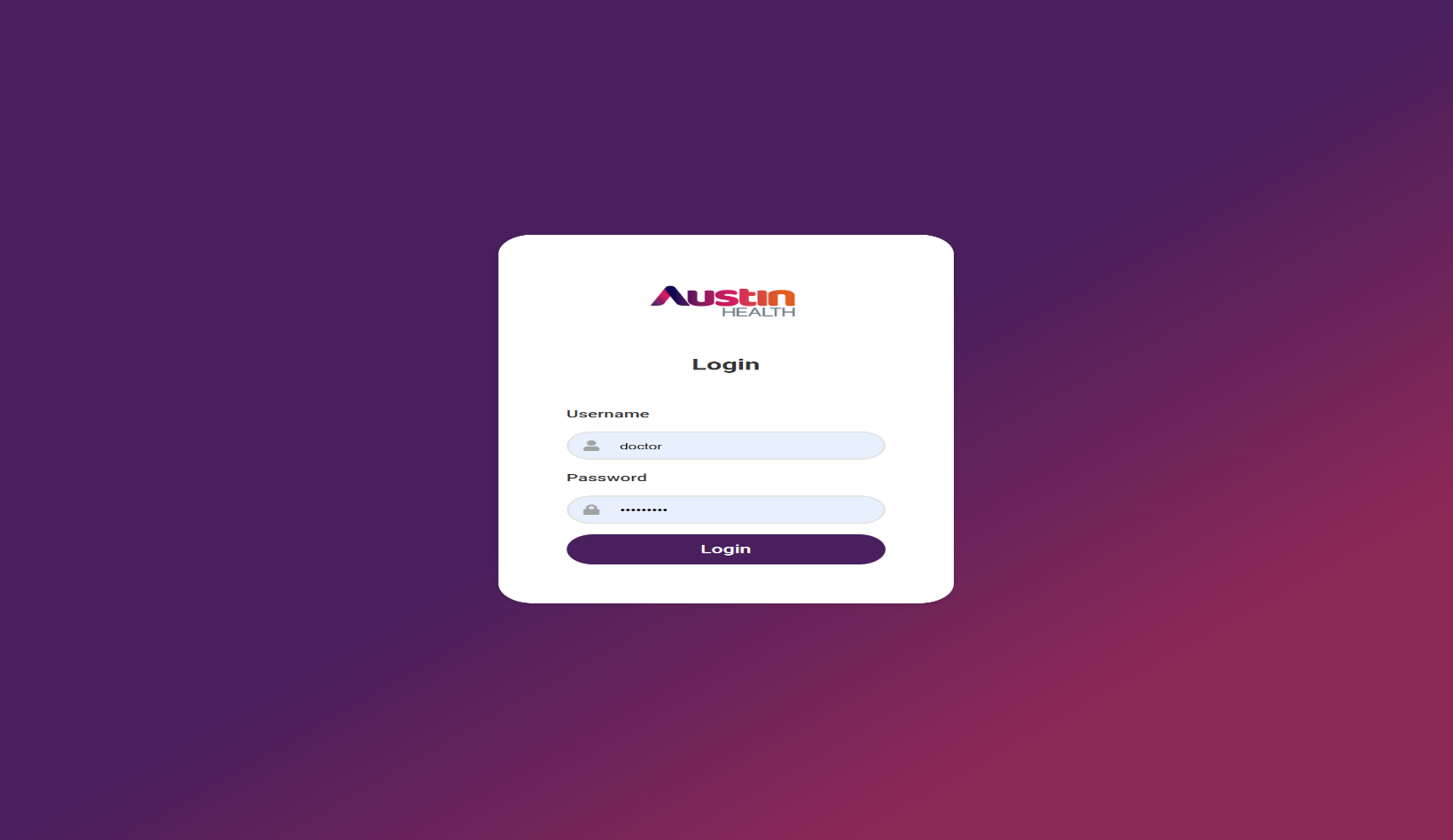


Fig. Chatbot AI Wireframe

**7.2.1 Screenshots of UI Design**

The following screenshots shows how the system user interface really looks like, involving technical skills and creativity based on system requirements:

Fig. Login UI

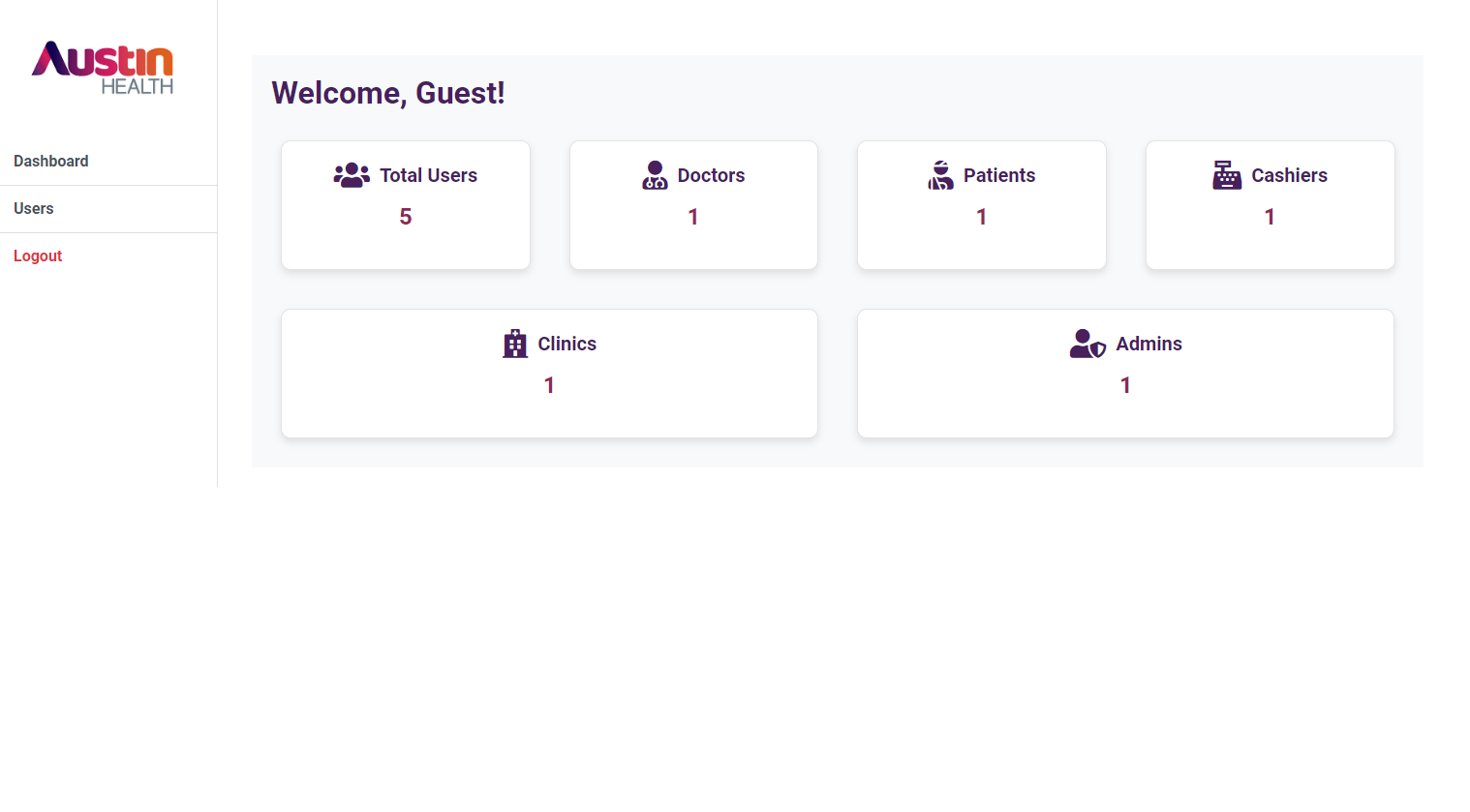


Fig. Admin Dashboard UI

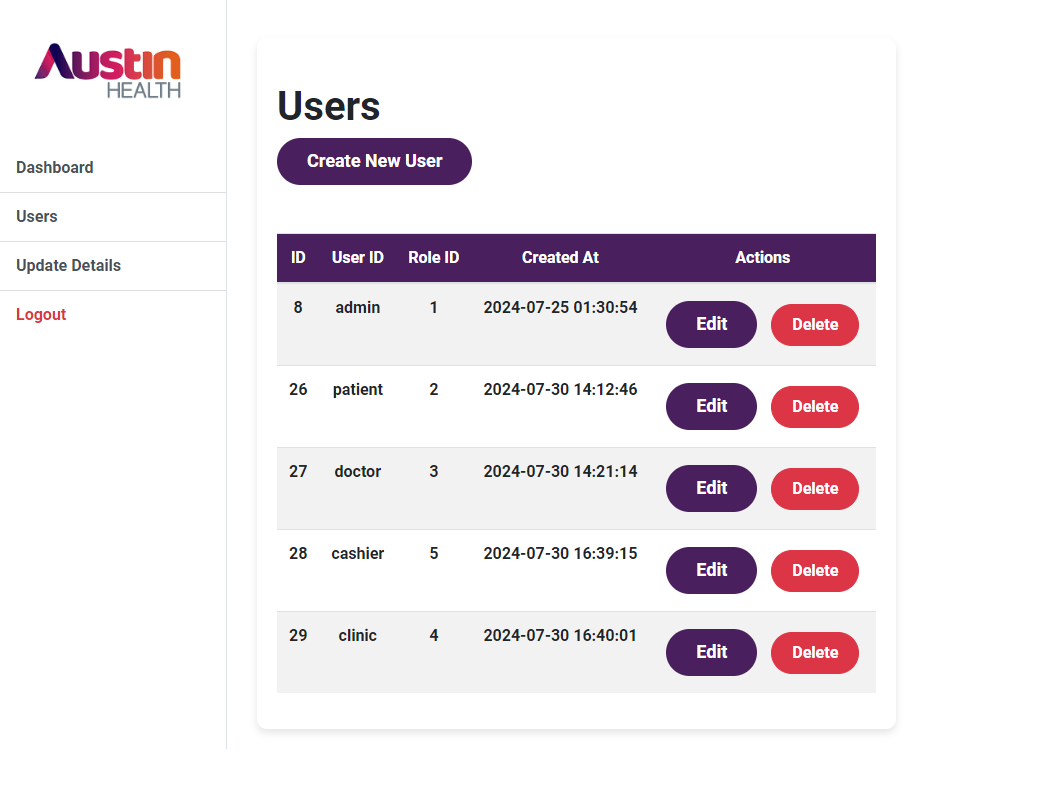


Fig. Admin Users UI

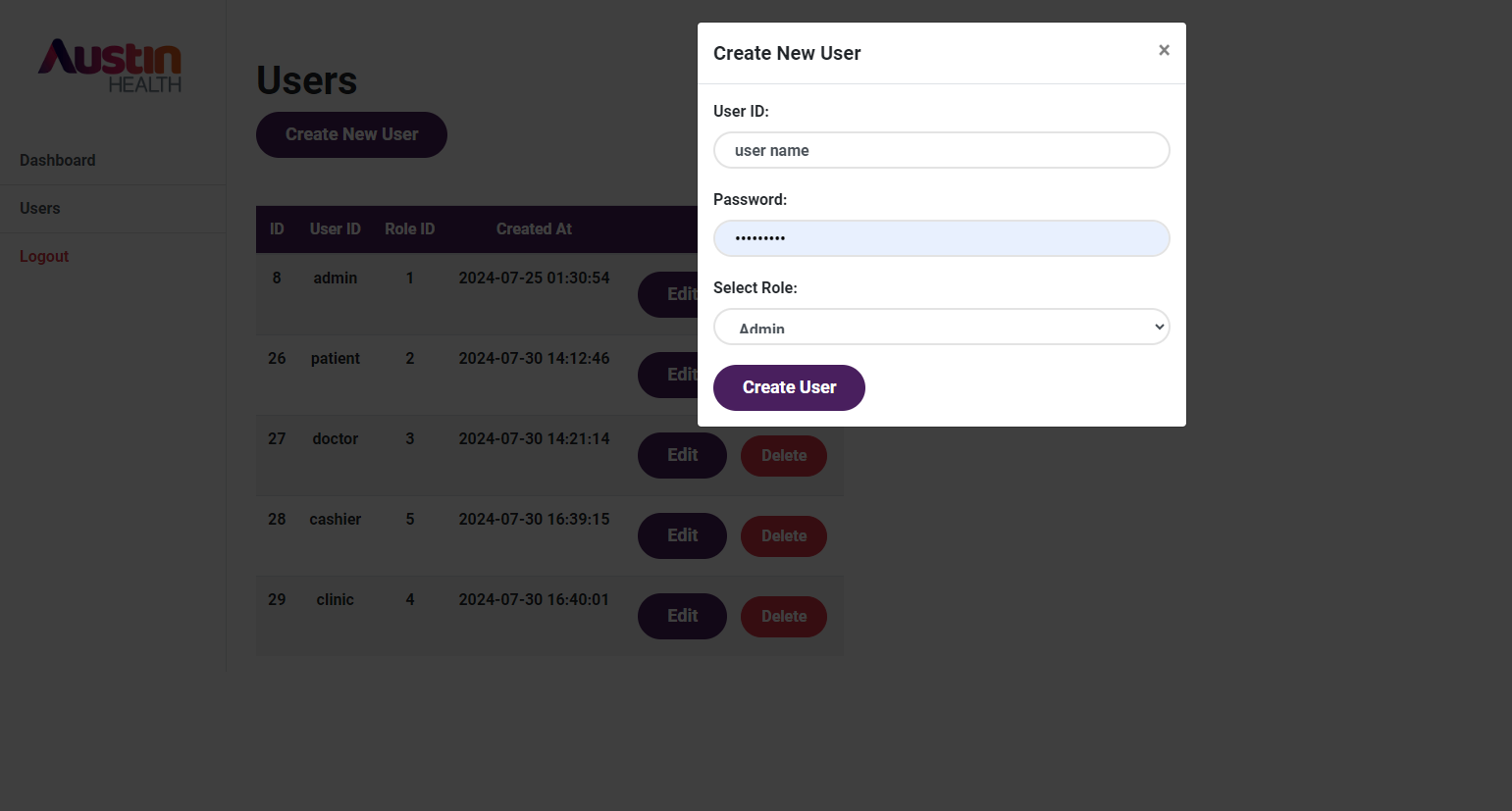


Fig. Admin Create New User UI

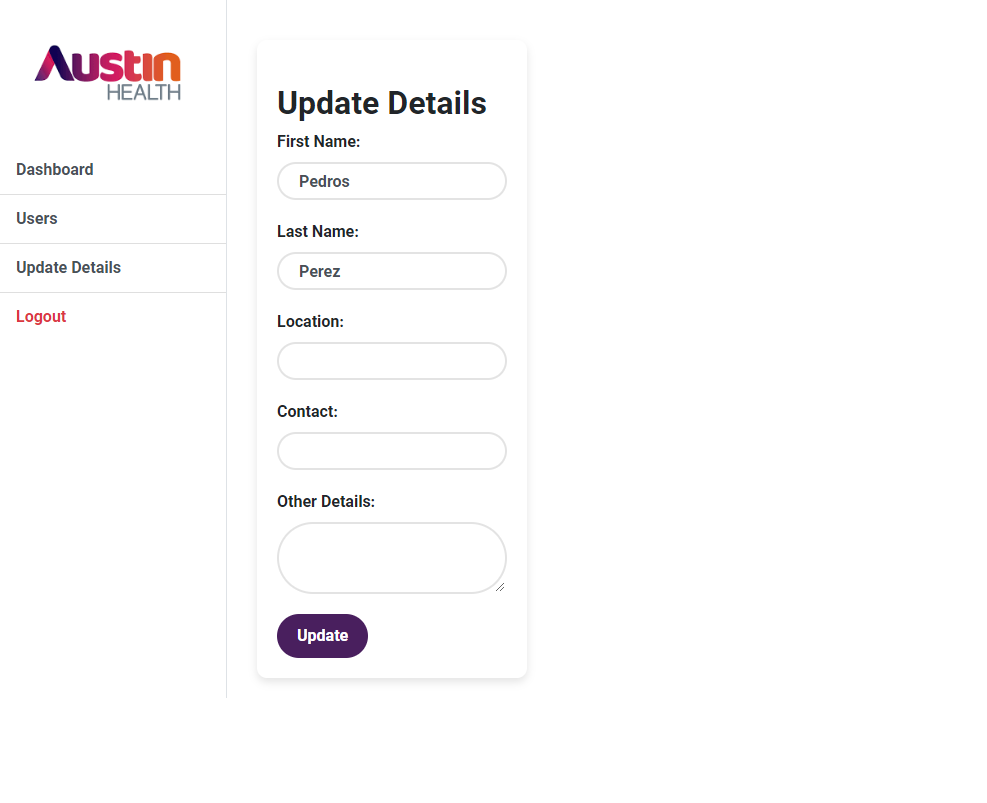


Fig. Admin Update Details UI

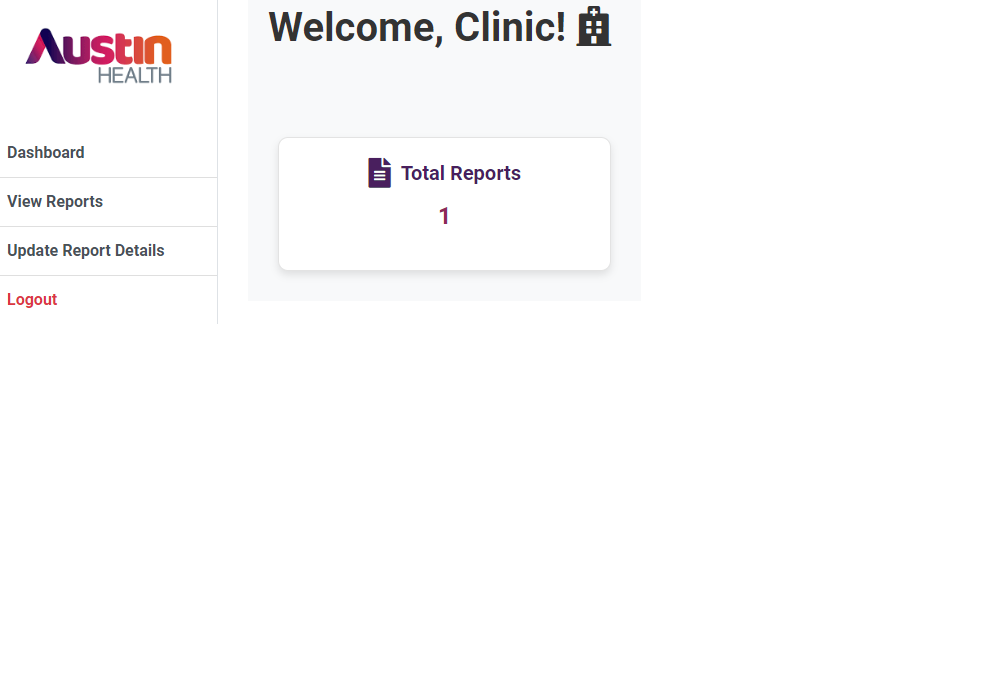


Fig. Clinic Dashboard UI

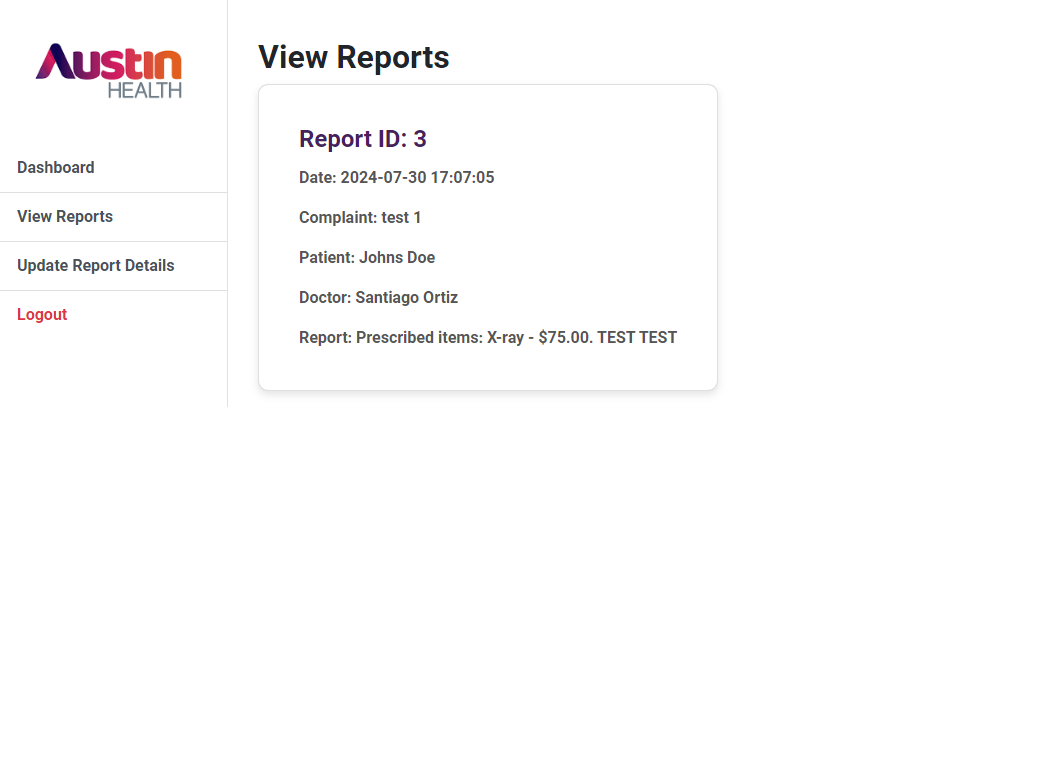


Fig. Clinic View Reports UI

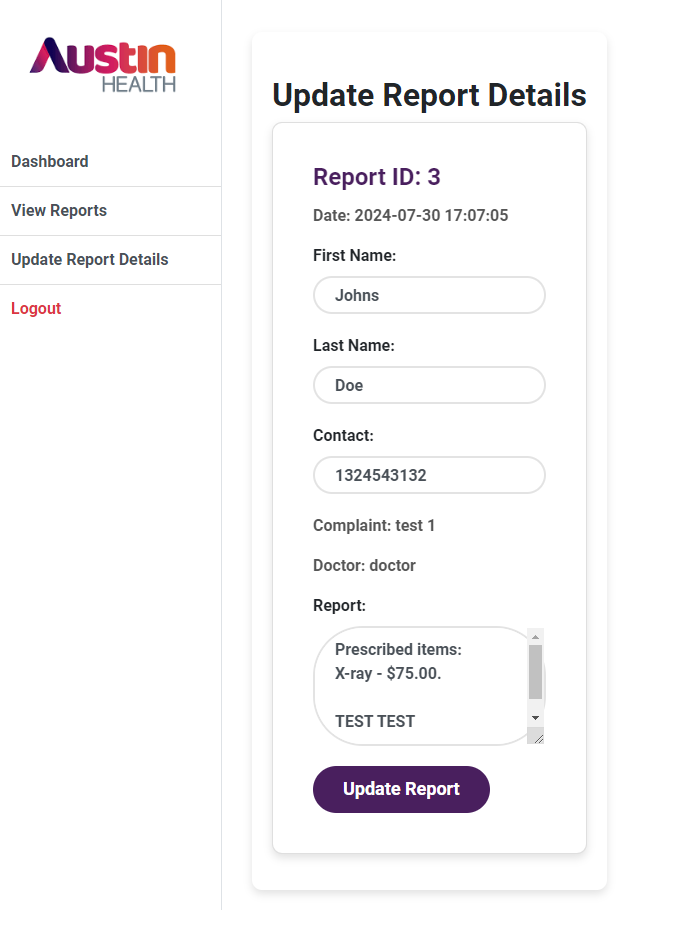


Fig. Clinic Update Report Details UI

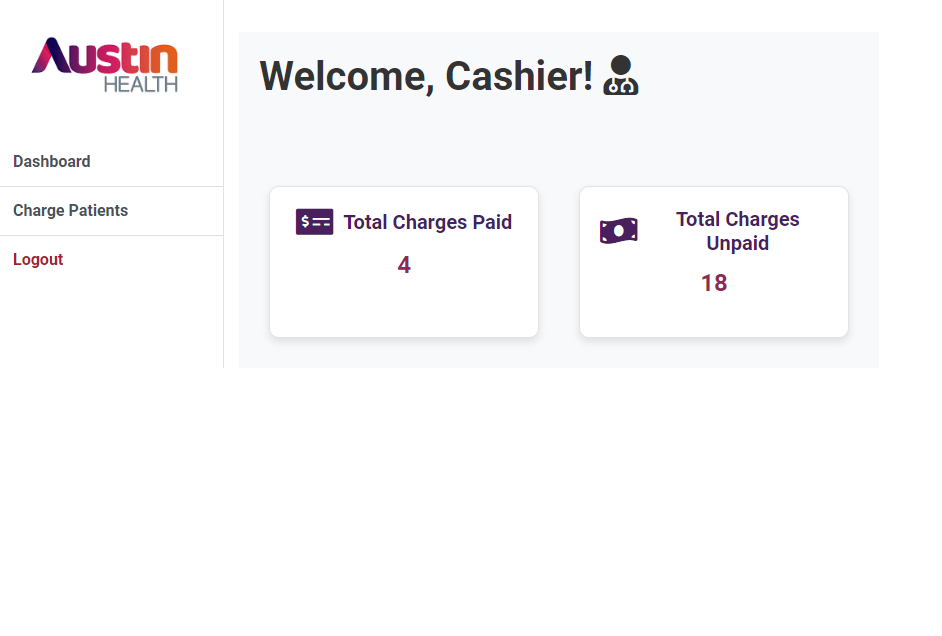


Fig. Cashier Dashboard UI

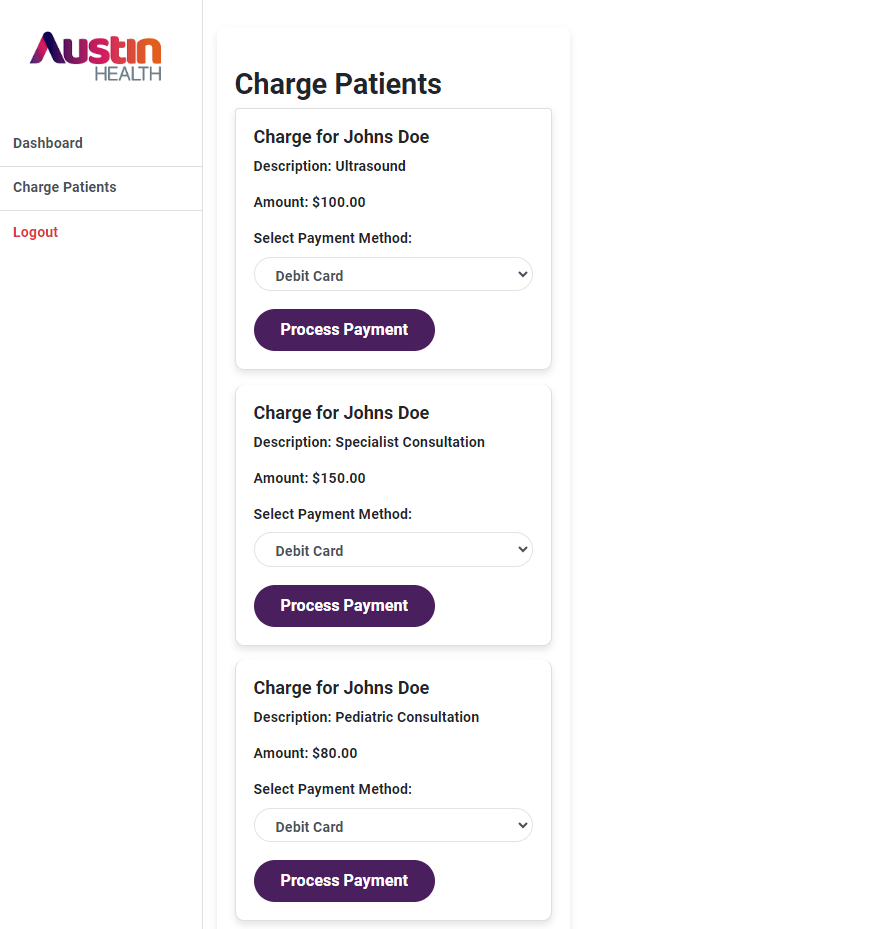


Fig. Cashier Charge Patients UI

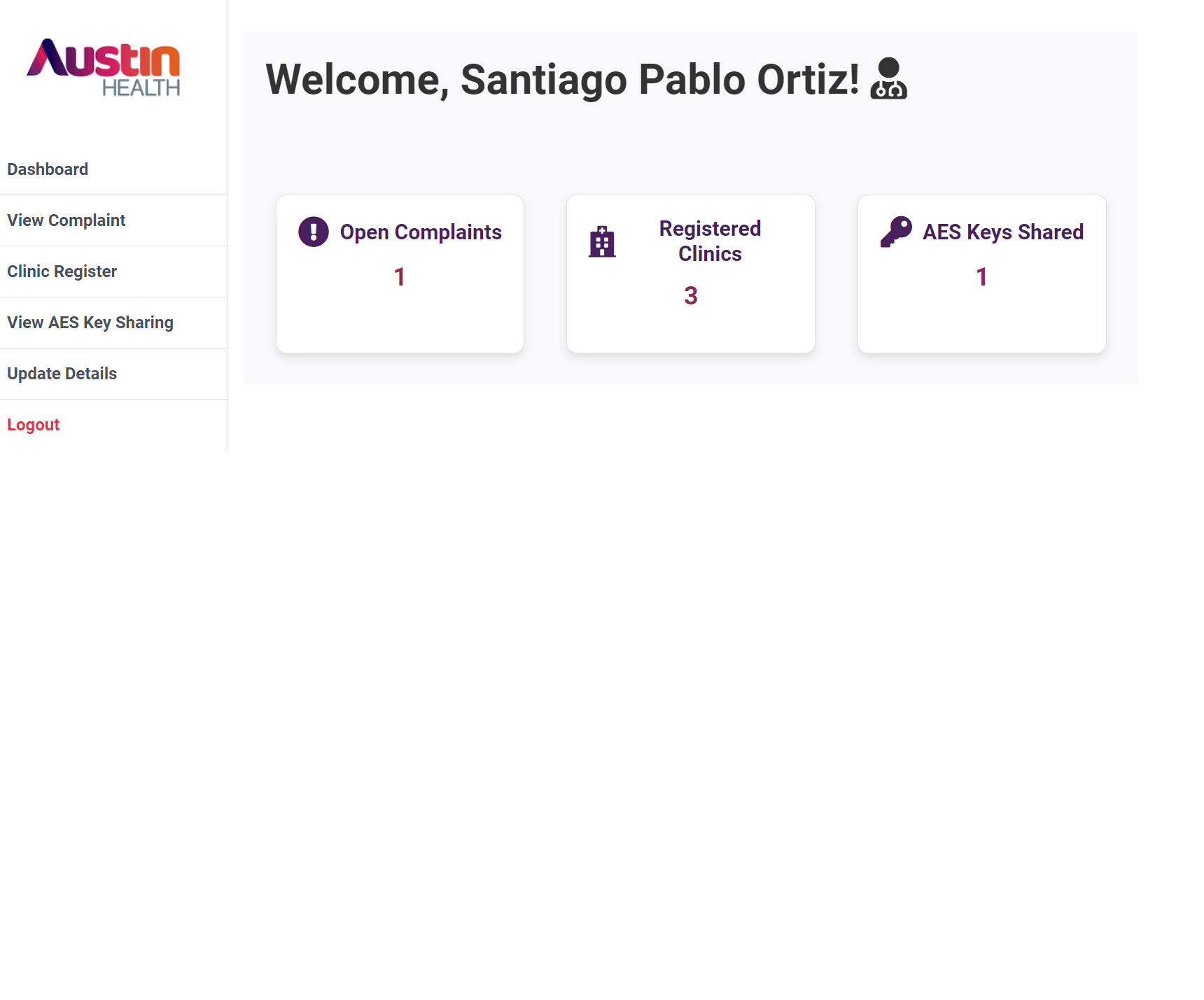


Fig. Doctor Dashboard UI

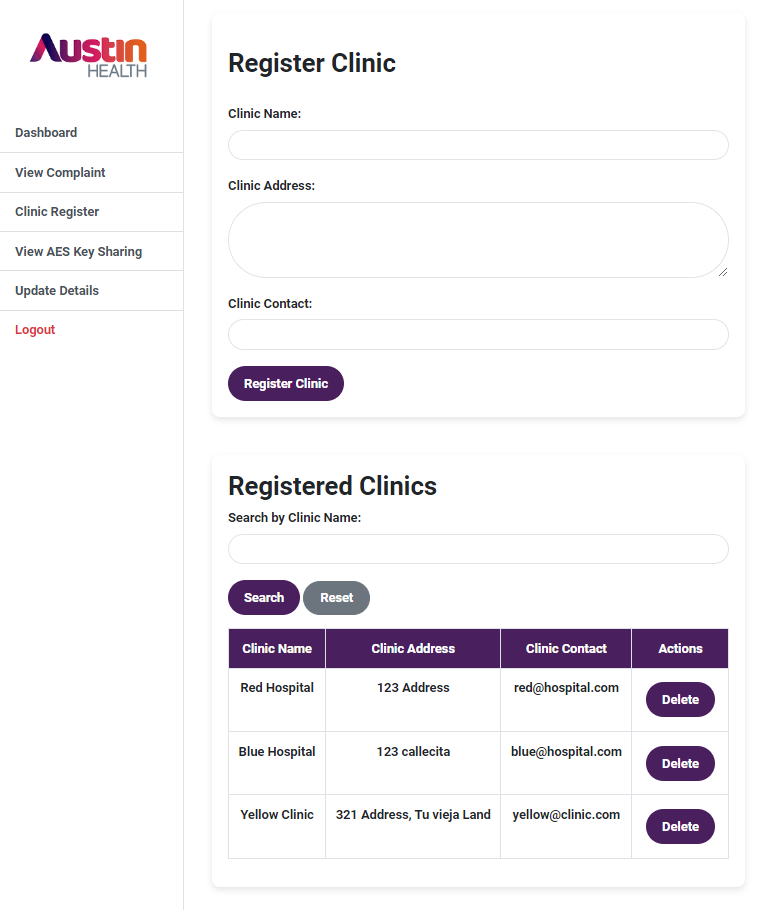


Fig. Doctor Clinic Register UI

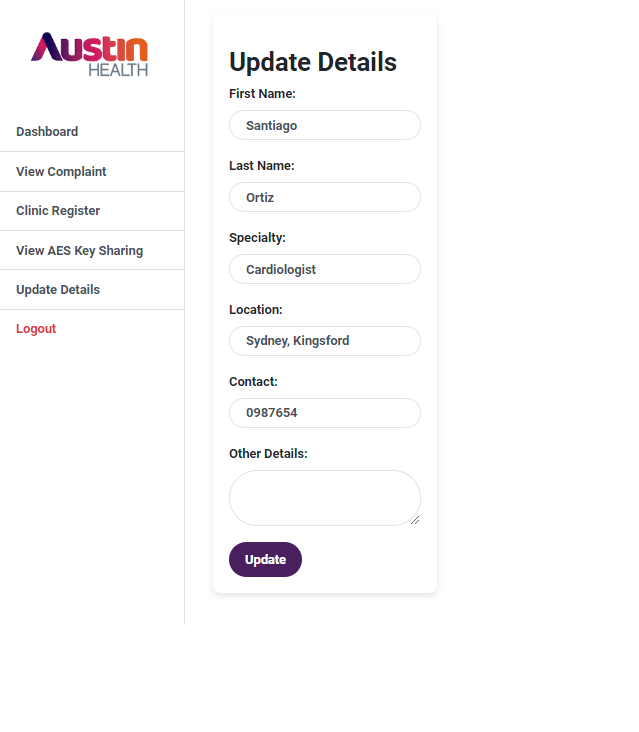


Fig. Doctor Update Details UI

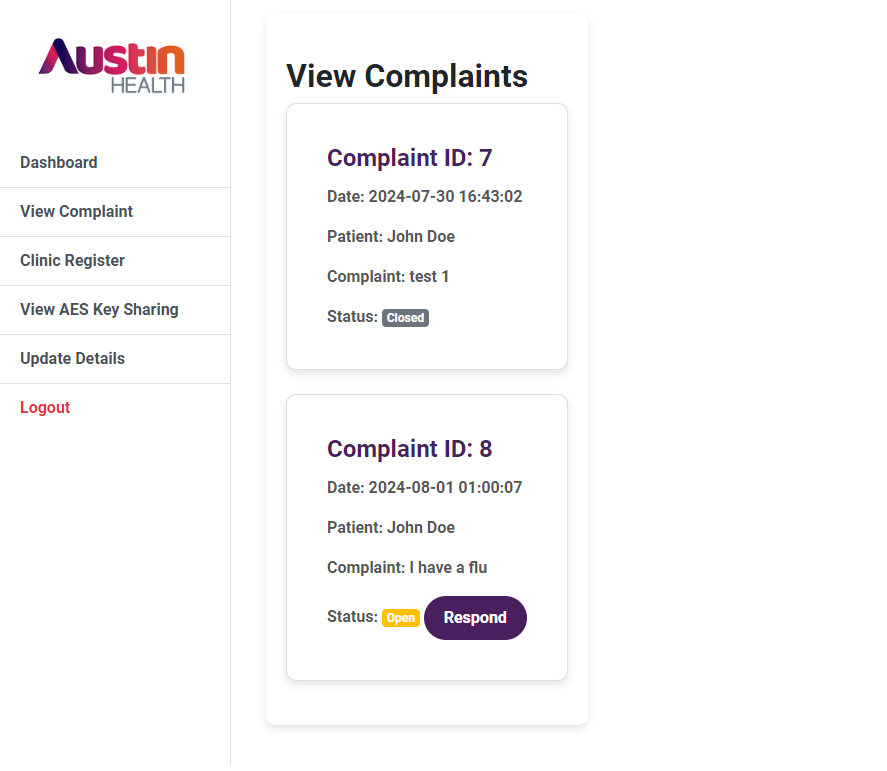
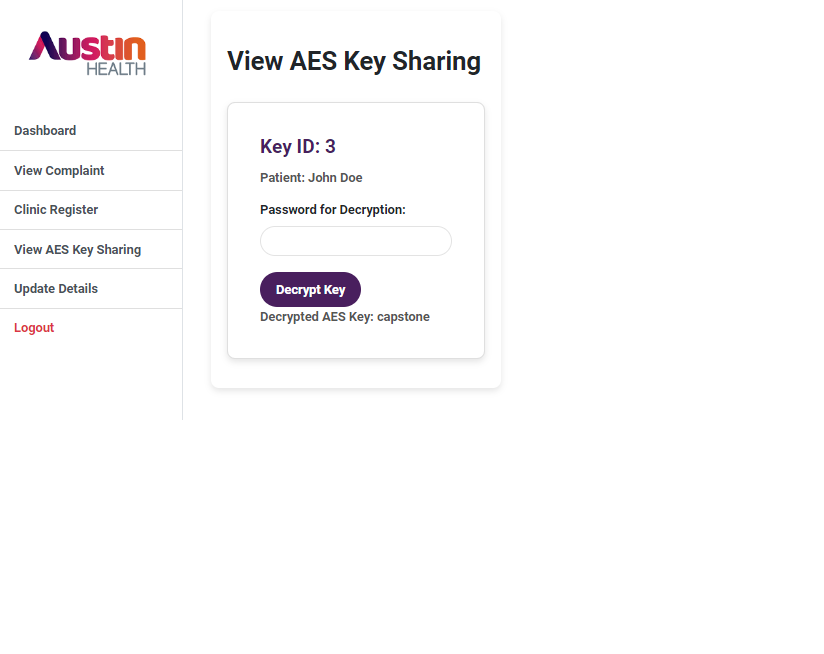


Fig. Doctor View Complaint UI

Fig. Doctor View AES Key Sharing UI

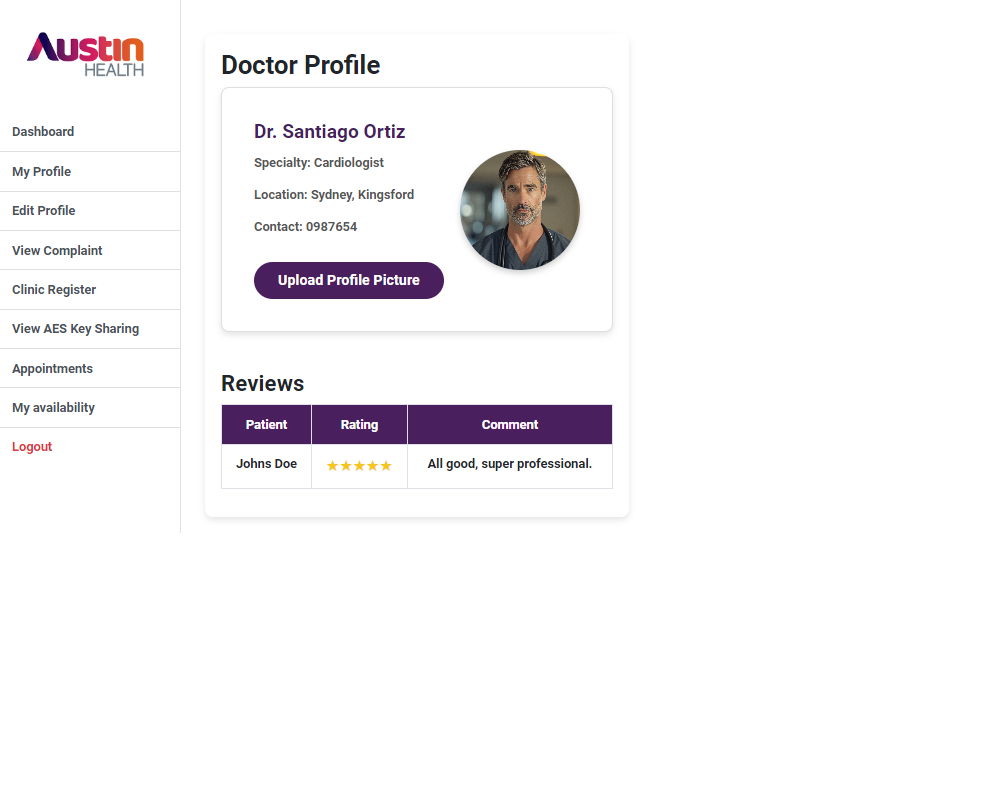


Fig. Doctor Profile UI

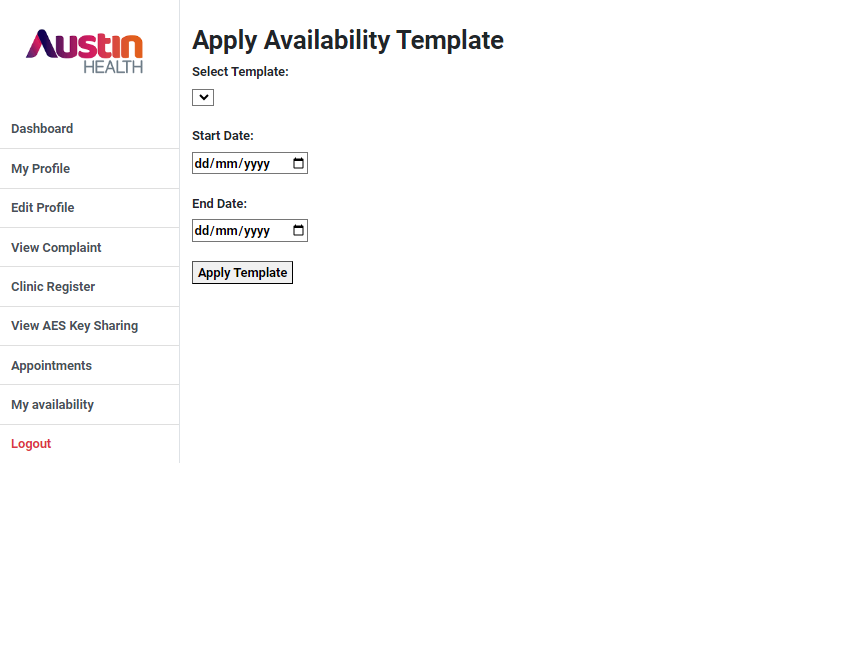


Fig. Doctor Apply Availability UI

A screenshot of a computer

Description automatically generated

Fig. Doctor Appointments UI

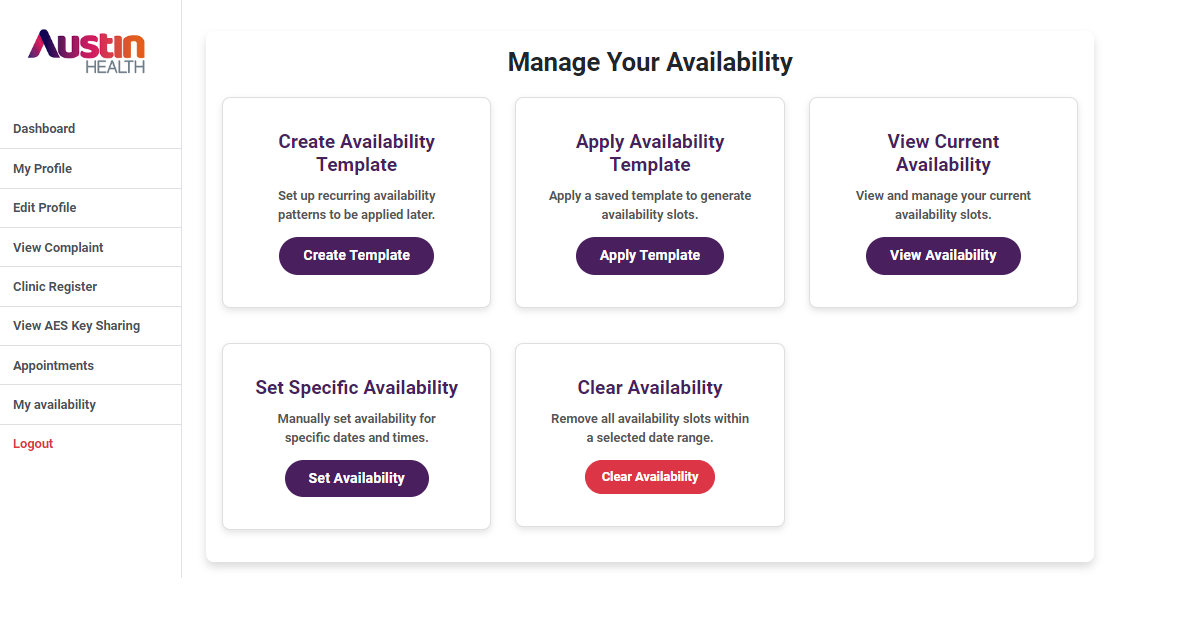


Fig. Doctor Availability Panel UI

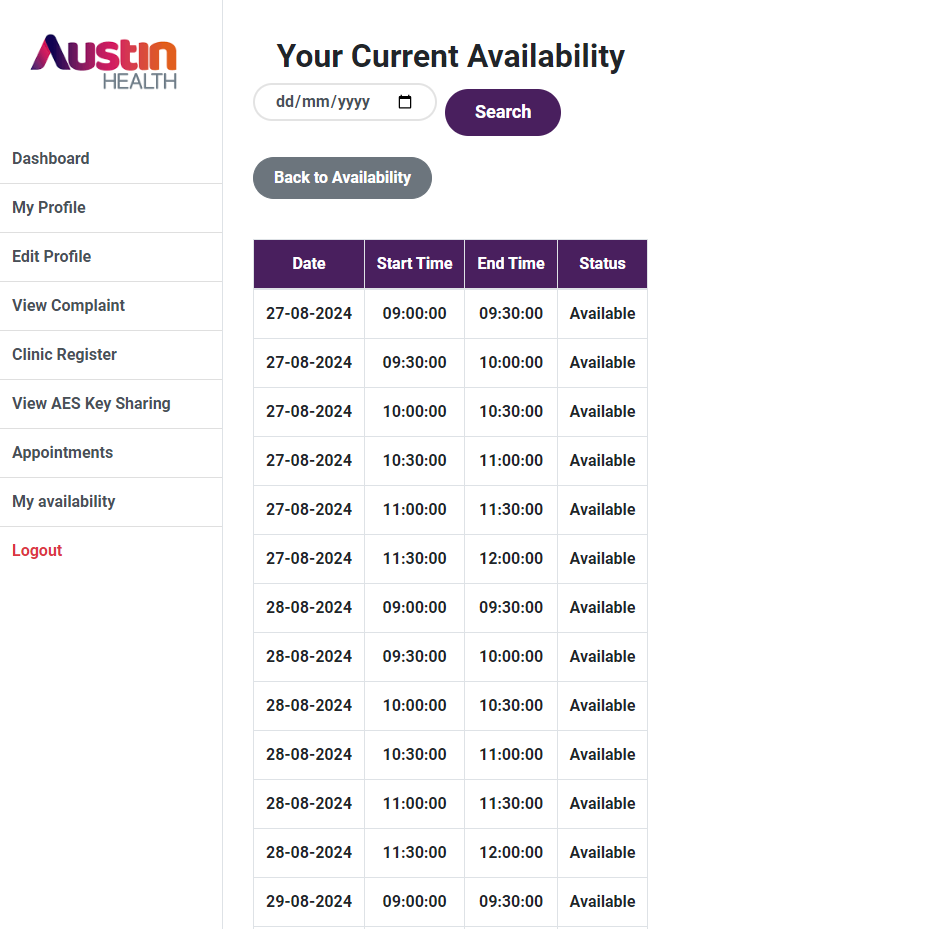


Fig. Doctor Check Availability UI

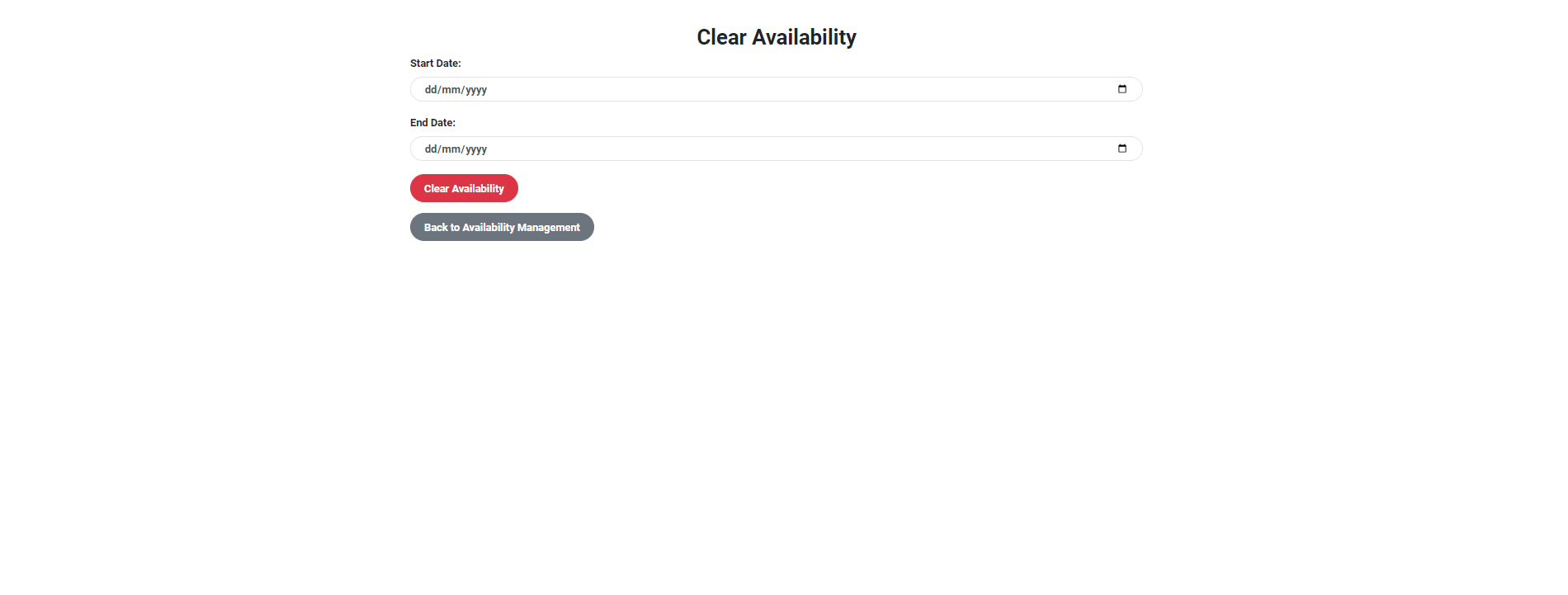


Fig. Doctor Clear Availability UI

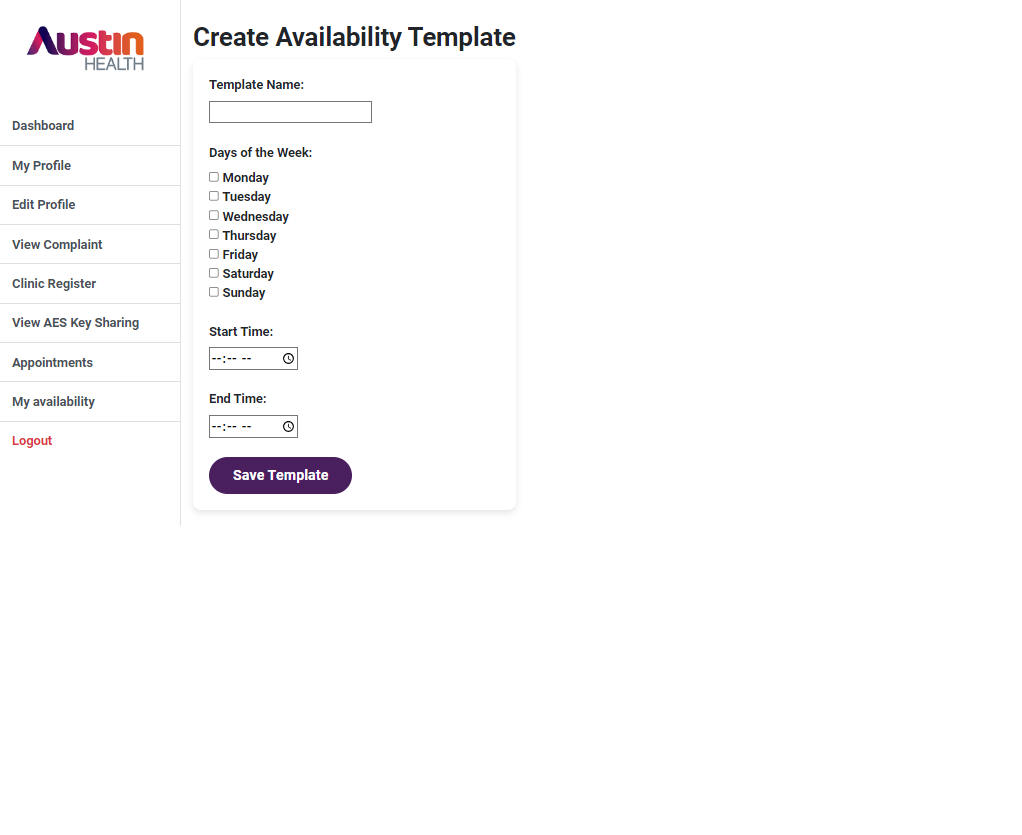


Fig. Doctor Create Availability Template UI

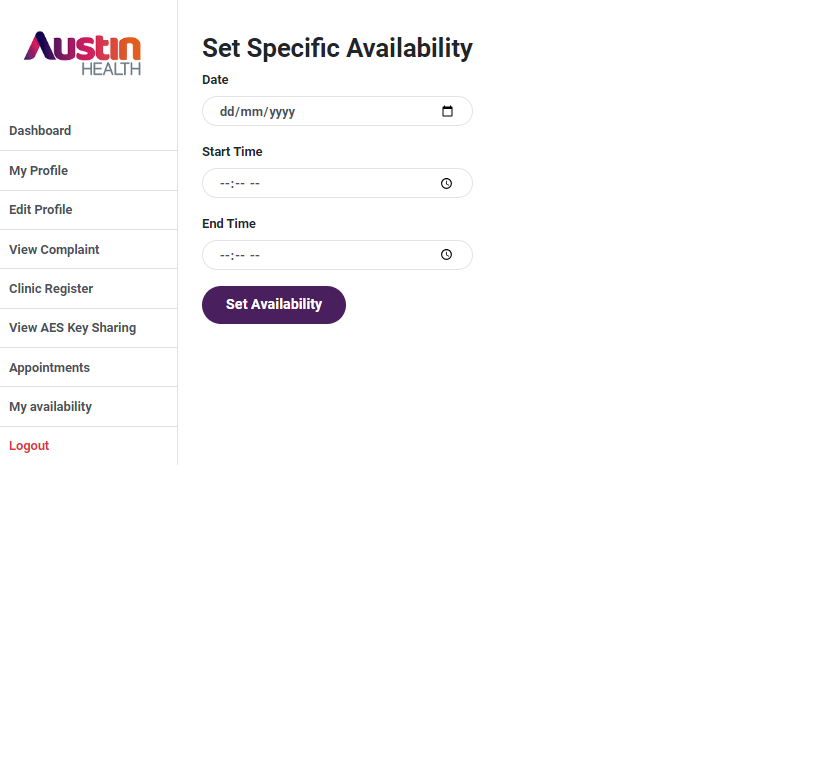
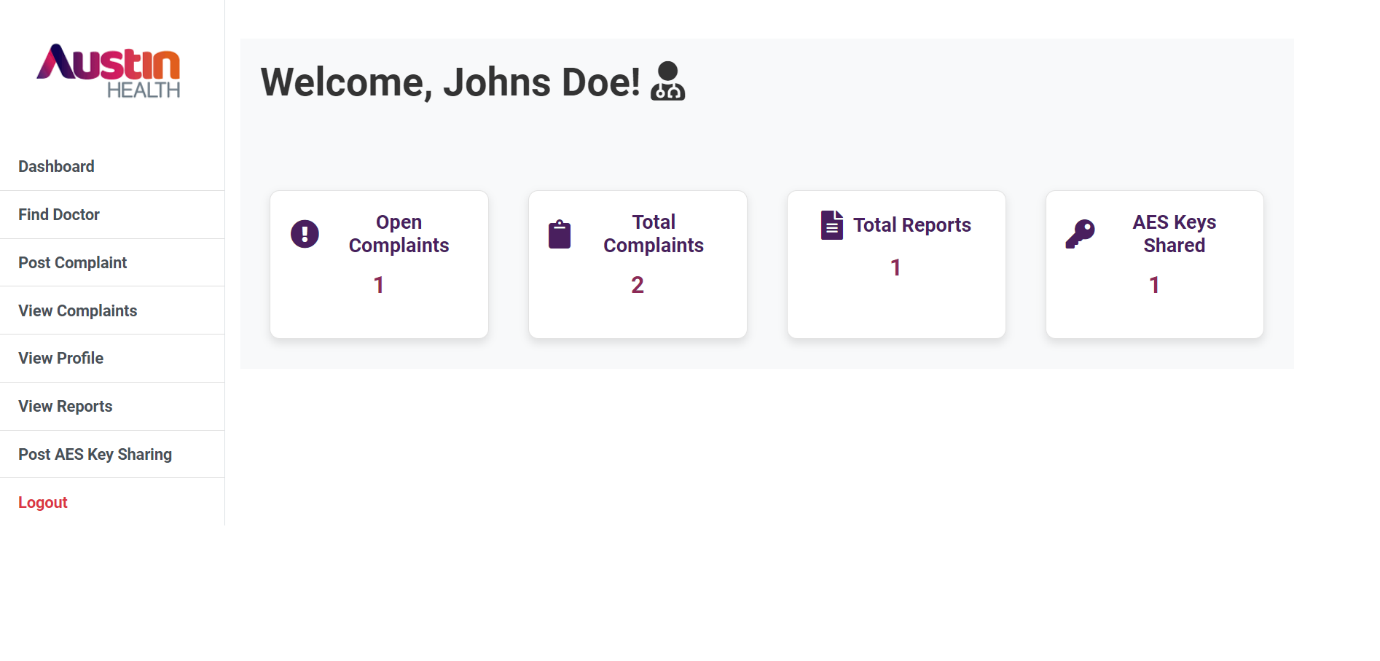


Fig. Doctor Set Specific Availability UI

fig. Patient Dashboard UI



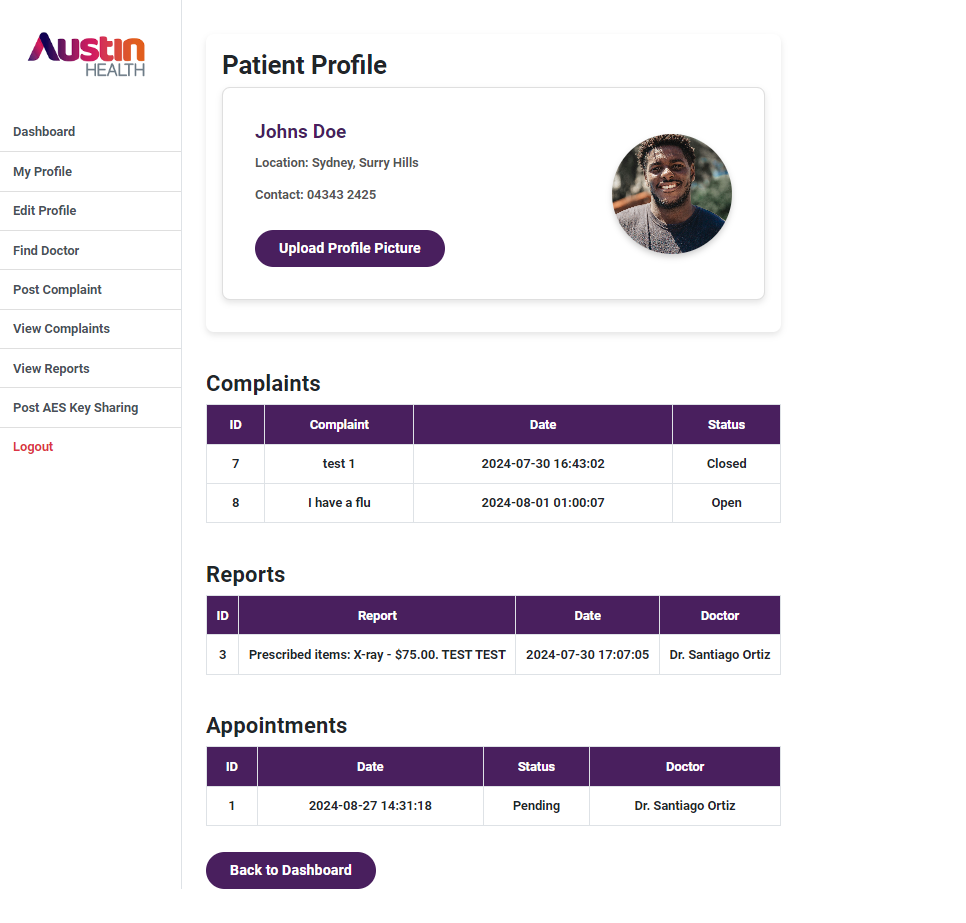


Fig. Patient View Profile UI

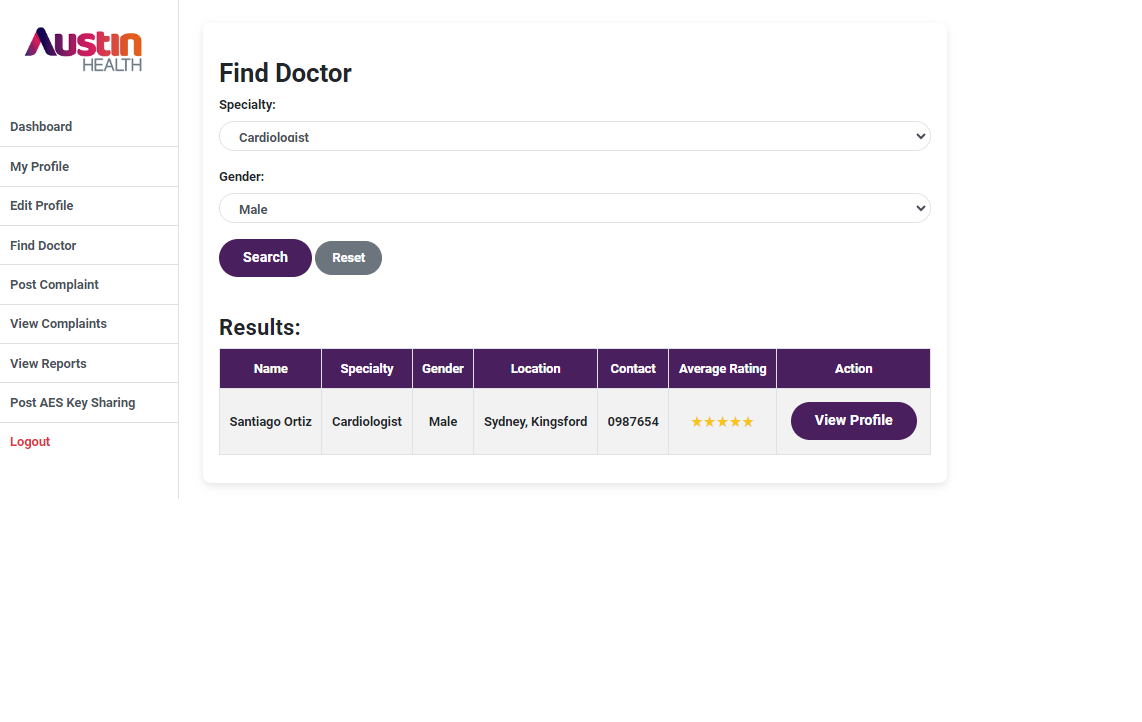


Fig. Patient Find Doctor UI

|  |  |
| --- | --- |
|  | Fig. Patient View Reports UI |
| Fig. Patient Post Complaint UI |  |
|  | Fig. Patient View Complaint UI |

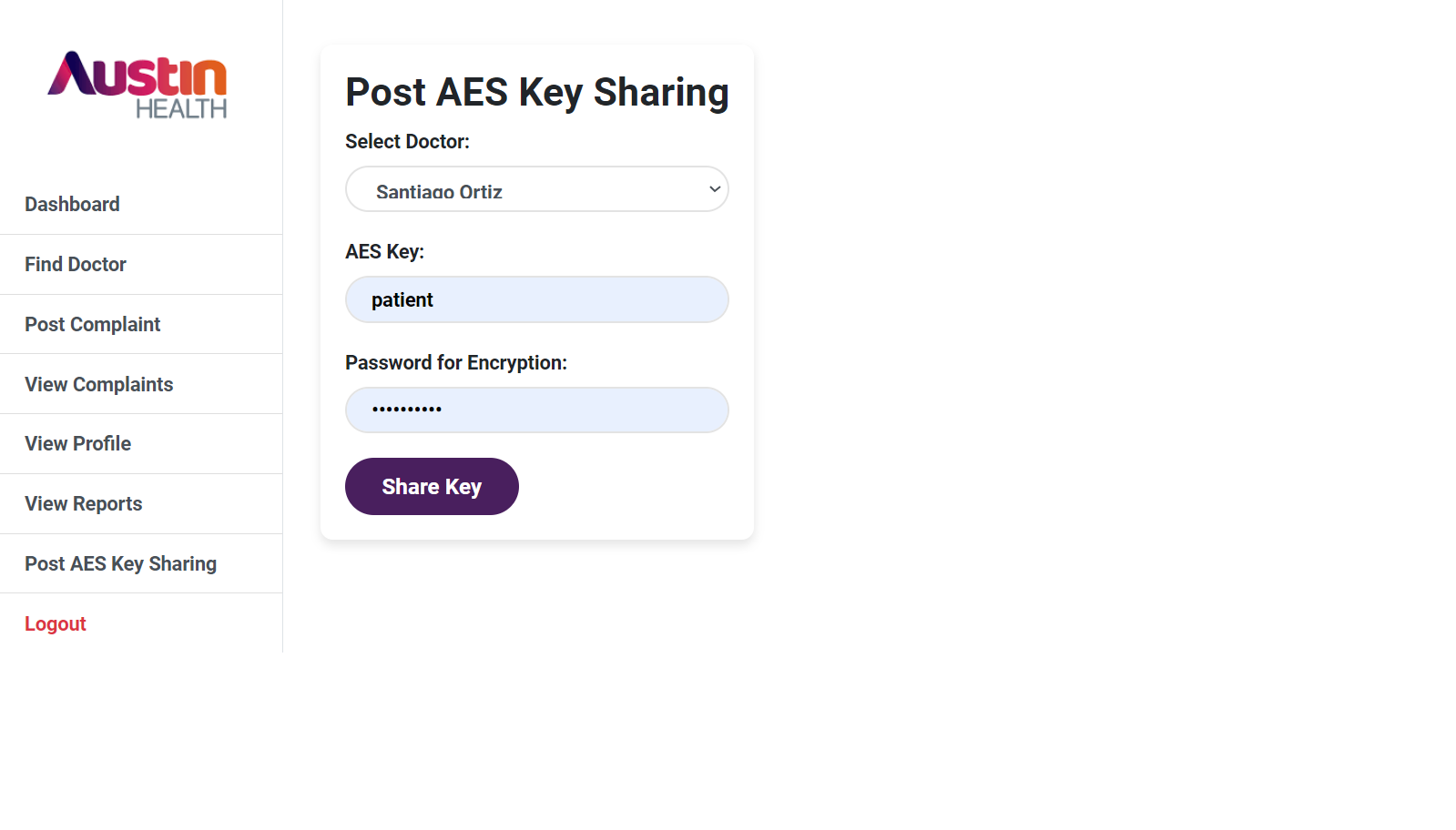


Fig. Patient Post AES Key Sharing UI



Fig. Patient Appointment Booked Successfully UI



Fig. Patient Check Doctor Availability UI

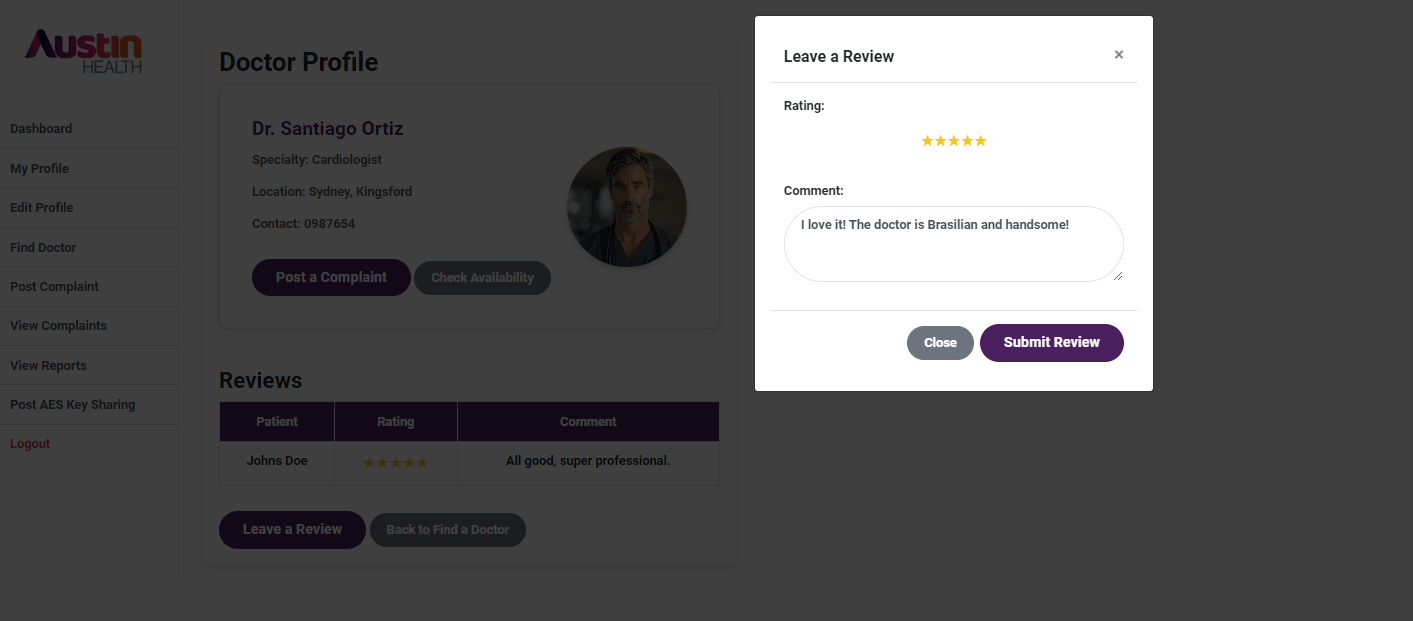


Fig. Patient Leave Review UI

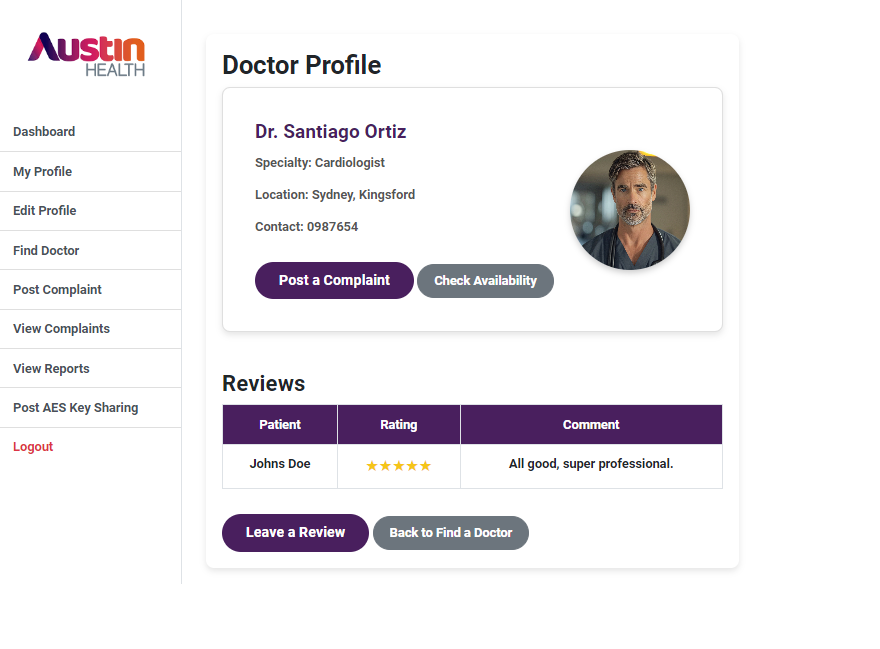


Fig. Patient See Doctor Profile UI

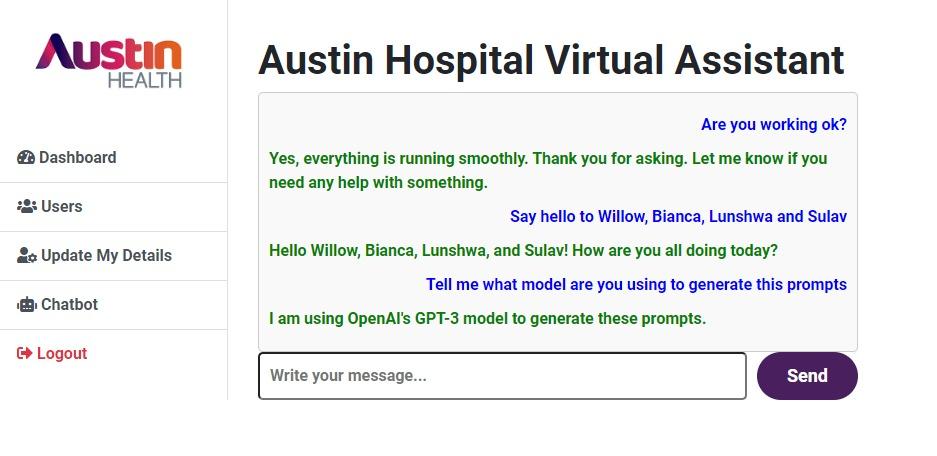


Fig. Chatbot AI UI

### **8. Programming and Implementation**

This part explains how the system code looks as well as how the code is written. Even though the current structure does not include an MVC pattern, the submissions display the current logic of the program regarding the user authentication, data processing and the user interfaces. The long-term plan, however, is to rewrite the code into an MVC architecture to resolve the issues of coupled dependencies.

#### **Code Overview**

This system comprises different sub systems that perform various tasks in the management of the hospital. Below is an overview of these components:

* **login\_process.php:** Manages the process of authentication of a user and duration of the user’s session.
* **dashboard.php:** Takes the user to the admin homepage in which they can see many statistics related to the system.
* **process.php:** Responsible for the process of generation of new records of users in the database.

#### **Frontend Code**

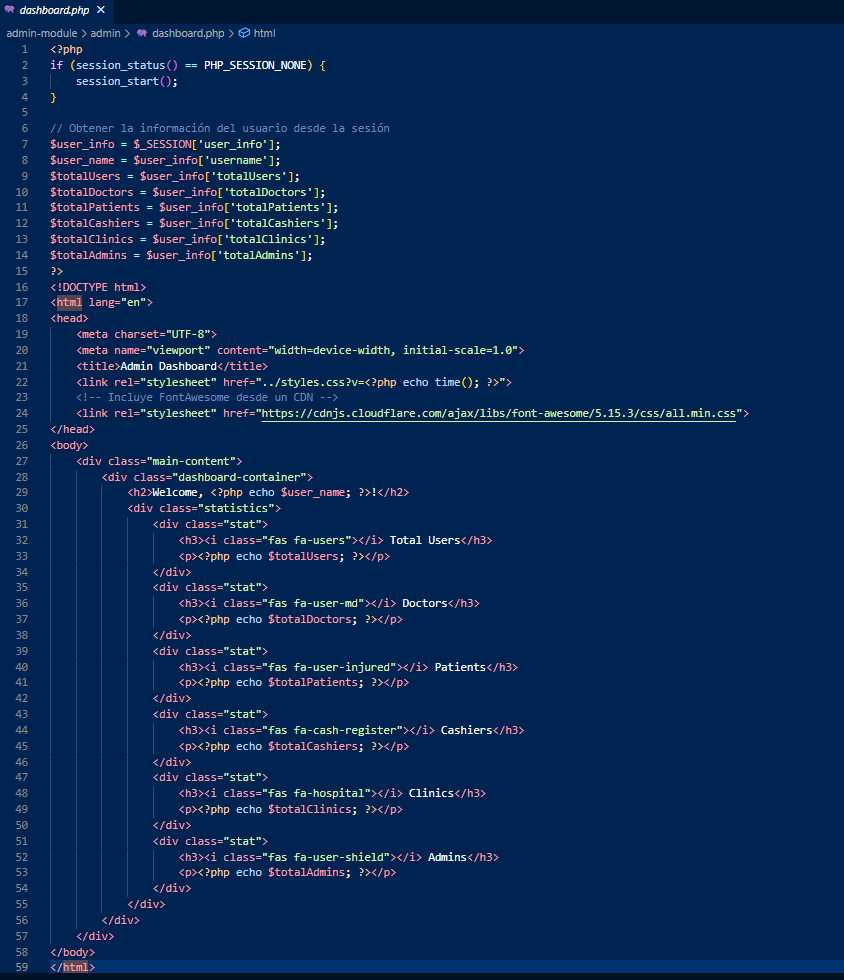
Frontend code contains and is mainly involved in rendering the user interfaces and for communication with the back end for get or post operations. HTML for the layout, CSS for design or look and feel and JavaScript for the interaction with the application.

#### **Technologies Used:**

* **HTML**: Provides the basic structure of the web pages.
* **CSS**: Handles the presentation and styling, ensuring the interface is visually appealing and consistent.
* **JavaScript**: Adds interactivity to the pages, allowing for DOM manipulation, real-time validation, and asynchronous server calls.

**Example: dashboard. php**

The dashboard.php file is an example of a bare Bones frontend page which provides key statistics to the admin user. It employs the HTML to structure the web page and on styling, we have the CSS by link which can be an external link. The information shown in the dashboard is obtained from the session variables, which are inputted during login.

****

This page demonstrates how different statistics are displayed to the admin using session data passed from the backend.  
  
Link to the file: <https://drive.google.com/file/d/1htg4gd4r2qZfPo_eUHE9sja79Q68SEDk/view?usp=sharing>

#### 

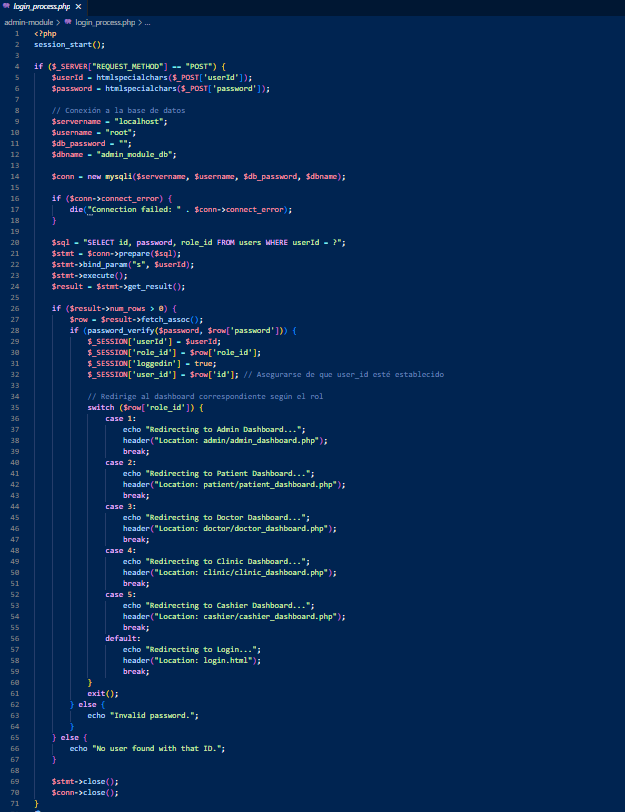
#### 

#### **Backend Code**

The backend code handles data processing, authentication, and interactions with the database. Below are examples from login\_process.php and process.php.

**Example: login\_process.php**

This file handles the user login process. It checks the user's credentials against the database, sets session variables upon successful authentication, and redirects the user to the appropriate dashboard based on their role.

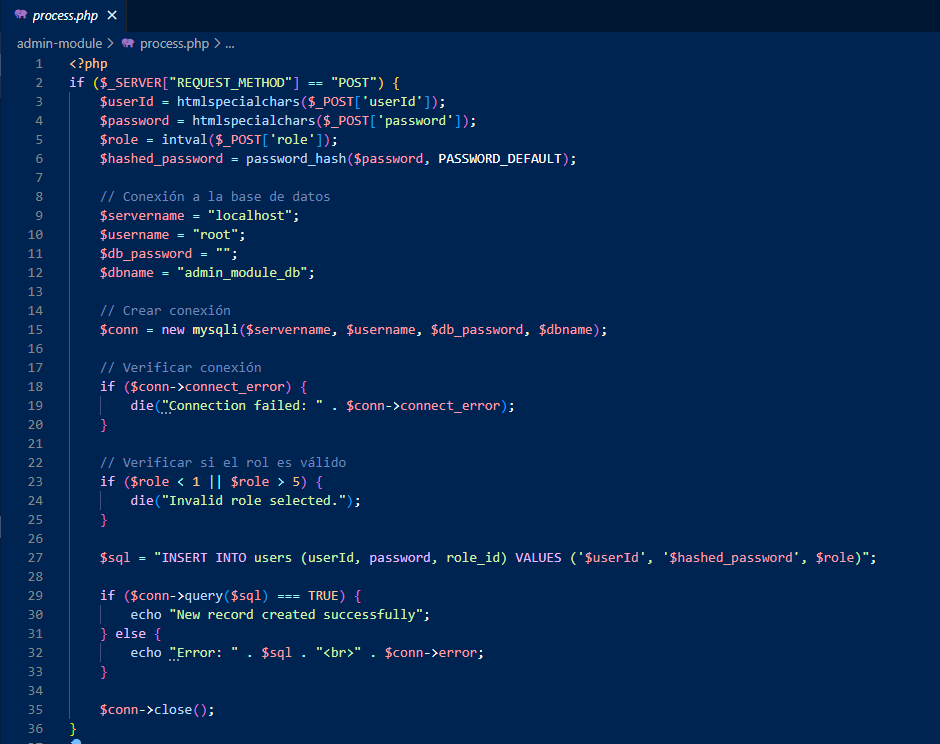
****

**Link to the file:**

<https://drive.google.com/file/d/1sM_c4uJ7vvPoNJdAhRSUv7jqyDTmV_q8/view?usp=sharing>

**Example: process.php**

This file manages the creation of new user records in the database. It validates the role, hashes the password, and inserts the new user data into the users table.



Link to the file:

<https://drive.google.com/file/d/1dYYkbNFq1laPvpiqD3zNcmXOIGRdHcfd/view?usp=sharing>

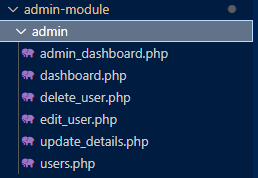
**Key Features:**

* **Session Management:** Both login\_process.php and dashboard.php demonstrate the use of PHP sessions to manage user login states and roles.
* **Security:** The backend uses prepared statements and input sanitization to prevent SQL injection attacks. Passwords are hashed using password\_hash() to ensure secure storage.
* **Database Interaction:** The backend communicates with the MySQL database to retrieve user information, validate credentials, and store new records.

### **9. Screenshots and File Attachments**

* **Screenshots of Files and Folders**

Austin Hospital System/

│

├── admin-module/

│ ├── admin/

│ │ ├── admin\_dashboard.php

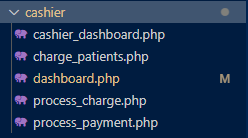
│ │ ├── dashboard.php

│ │ ├── delete\_user.php

│ │ ├── edit\_user.php

│ │ ├── update\_details.php

│ │ └── users.php

│ │

│ ├── cashier/

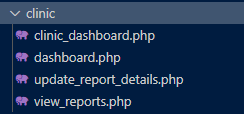
│ │ ├── cashier\_dashboard.php

│ │ ├── charge\_patients.php

│ │ ├── dashboard.php

│ │ ├── process\_charge.php

│ │ └── process\_payment.php

│ │

│ ├── clinic/

│ │ ├── clinic\_dashboard.php

│ │ ├── dashboard.php

│ │ ├── update\_report\_details.php

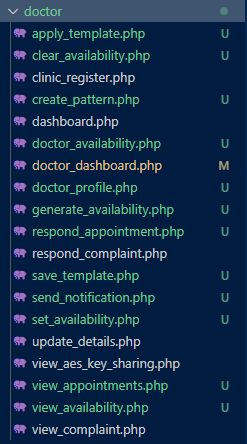
│ │ └── view\_reports.php

│ │

│ │

│ │

│ │

│ ├── doctor/

│ │ ├── apply\_template.php

│ │ ├── clear\_availability.php

│ │ ├── clinic\_register.php

│ │ ├── create\_pattern.php

│ │ ├── dashboard.php

│ │ ├── doctor\_availability.php

│ │ ├── doctor\_dashboard.php

│ │ ├── doctor\_profile.php

│ │ ├── generate\_availability.php

│ │ ├── respond\_appointment.php

│ │ ├── respond\_complaint.php

│ │ ├── save\_template.php

│ │ ├── send\_notification.php

│ │ ├── set\_availability.php

│ │ ├── update\_details.php

│ │ ├── view\_aes\_key\_sharing.php

│ │ ├── view\_appointments.php

│ │ ├── view\_availability.php

│ │ └── view\_complaint.php

│ │

│ │

│ │

│ │

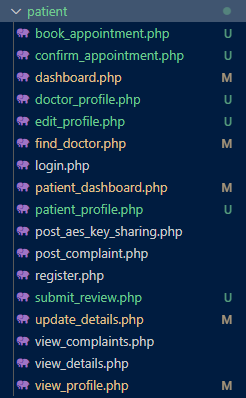
│ │

│ │

│ │

│ │

│ │

│ ├── patient/

│ │ ├── book\_appointment.php

│ │ ├── confirm\_appointment.php

│ │ ├── dashboard.php

│ │ ├── doctor\_profile.php

│ │ ├── edit\_profile.php

│ │ ├── find\_doctor.php

│ │ ├── login.php

│ │ ├── patient\_dashboard.php

│ │ ├── patient\_profile.php

│ │ ├── post\_aes\_key\_sharing.php

│ │ ├── post\_complaint.php

│ │ ├── register.php

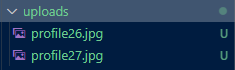
│ │ ├── submit\_review.php

│ │ ├── update\_details.php

│ │ ├── view\_complaints.php

│ │ ├── view\_details.php

│ │ └── view\_profile.php

│ │

│ ├── uploads/

│ │ └── (profile images)

│ │

│ │

│ │

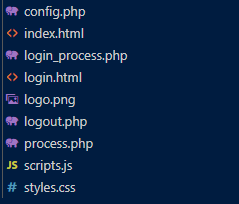
│ │

│ │

│ │

│ │

│ │

│ ├── config.php

│ ├── index.php

│ ├── login.php

│ ├── login\_process.php

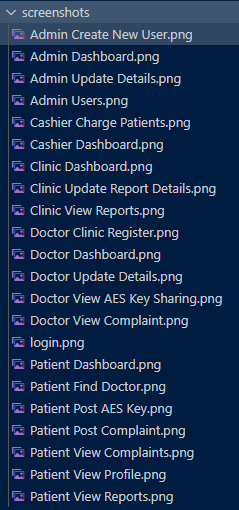
│ ├── logo.png

│ ├── logout.php

│ ├── process.php

│ ├── scripts.js

│ └── styles.css

│

├── screenshots/

│ └── (UI screenshots)

│

│

│

│

│

│

│

│

│

│

│

│

│

├── admin\_module\_db.sql

└── DFD.drawio

**Database SQL File**

Link to the file:

<https://drive.google.com/file/d/1p-vJuKi7v-fMapB_6buFTPUqBKcWKD7f/view?usp=sharing>

### **10. Conclusion**

The AHM System encapsulates the critical aspects of system development, from initial planning to final implementation. The project team had to be able to achieve rigorous project plans and effectively manage resources required to complete the project, the team was able to communicate the strategies and goals for each stage of the project effectively.

The team utilised collaborative and management tools to ensure that the project stages and sub-items were fully executed. The report above shows the blueprint for the subsequent design and development phases, the system architecture and database design were developed to ensure that the system is reliable, scalable and performance is optimal. The appropriate schemas and diagrams have been provided within the report to outline the business requirements. The development team where able to ensure the system is both robust and adaptable to future needs, and integrate new technologies/ security.

The AHMS project team has been successful in meeting the business requirements, objectives and deliverables for a user-friendly system for stakeholders. The core project has been able to demonstrate the team’s collaborative and scrum work throughout the timeline and shown proficiency in achieving the intended results.

### **11. References**

* Asana, T. (2024) ‘6 steps to requirements gathering for Project Success’, Asana. <https://asana.com/resources/requirements-gathering>
* Jama Software (2023) ‘A guide to requirements elicitation for product teams’. <https://www.jamasoftware.com/requirements-management-guide/requirements-gathering-and-management-processes/a-guide-to-requirements-elicitation-for-product-teams>
* Lucidchart (2021) ‘How to design software architecture: Top tips and best practices’. <https://www.lucidchart.com/blog/how-to-design-software-architecture>
* Tummers, J. et al. (2021) ‘Designing a reference architecture for Health Information Systems’, BMC Medical Informatics and Decision Making. BioMed Central. <https://bmcmedinformdecismak.biomedcentral.com/articles/10.1186/s12911-021-01570-2>