

Criterion B - Design

Pharmacy Management Application

Table of contents.....	1
Database Design using Entity Relationship Diagrams.....	3
Authentication, Frontend and Backend Connection.....	9
Menu Bar.....	11
Drawer.....	12
Home Page.....	13
Sign In Page.....	14
Login Page.....	16
Logout.....	18
Administrator Profile.....	18
Homepage for Administrator Profile.....	20
Pages in Administrator.....	20
View Lab Tests.....	21
Add New Medicines.....	22
Add Existing Medicines.....	23
Doctor Profile.....	24
Front Desk Profile.....	26
Token Creation Logic.....	27
Figure 25: Token Creation Logic when creating an appointment.....	28
Pharmacy Profile.....	28
Test plan.....	29

The Pharmacy Management Application has four main user profiles, each granting access to different system functionalities. The profiles are Admin, Doctor, Front Desk, and Pharmacy.

Figure 1 illustrates the features offered to each profile.

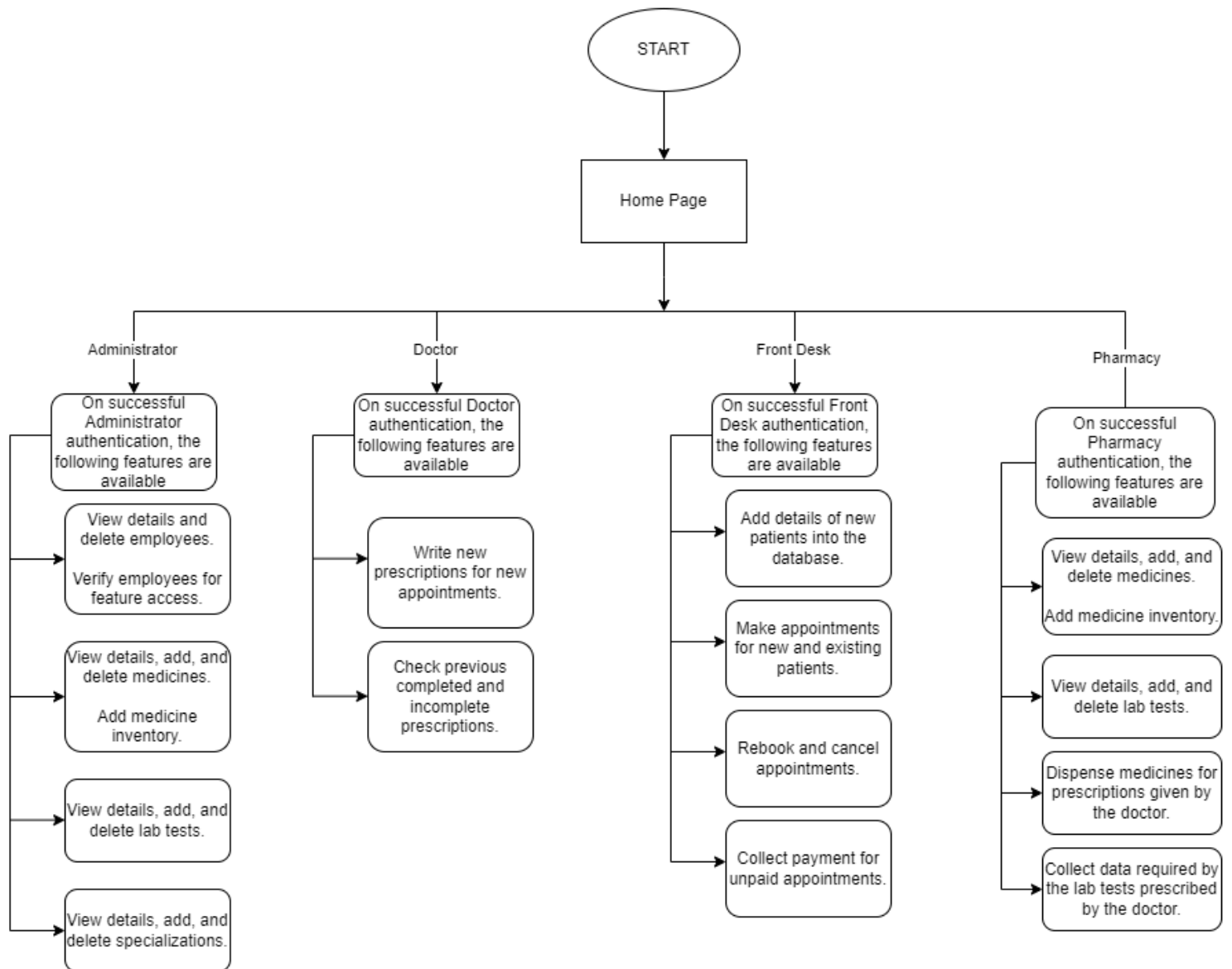


Figure 1: Overview of the Permissions and Features for Profiles

Database Design using Entity Relationship Diagrams

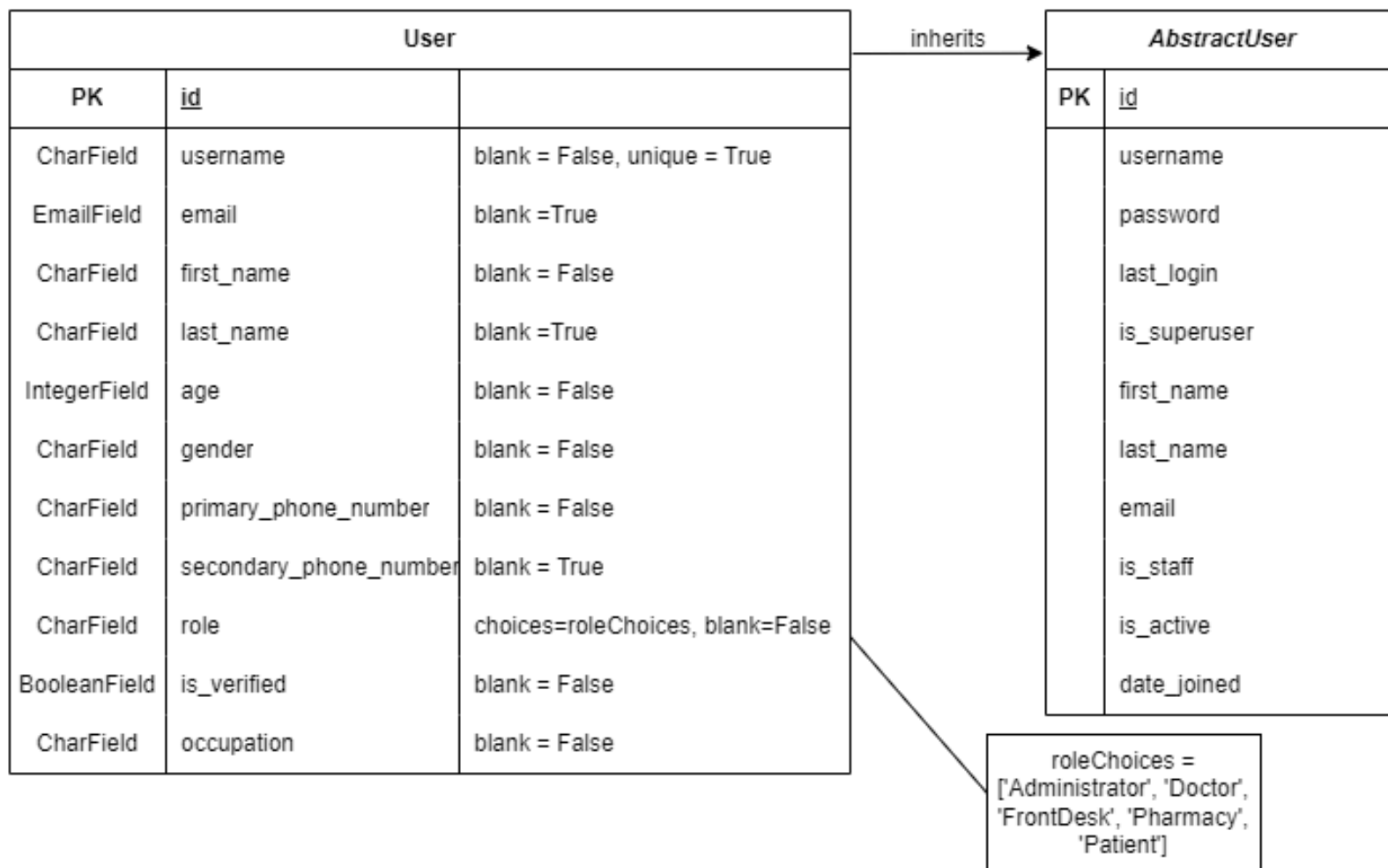


Figure 2: User model

All user data are stored in the **User** table, which extends the *AbstractUser* class defined by Django, as shown in Figure 2. This approach enables the table to use the security and authentication features Django provides for users while also increasing the flexibility and customisation of the user model to fit application requirements. Custom validation and creation of user entries will also be employed through a customised user manager. The User table adds fields, including `is_verified`, `occupation`, `first_name`, and `last_name`. It also overrides some fields, such as `username` and `email`, to fit the application's needs. The role is a

special CharField, which can only take the values defined in the variable roleChoices variable.

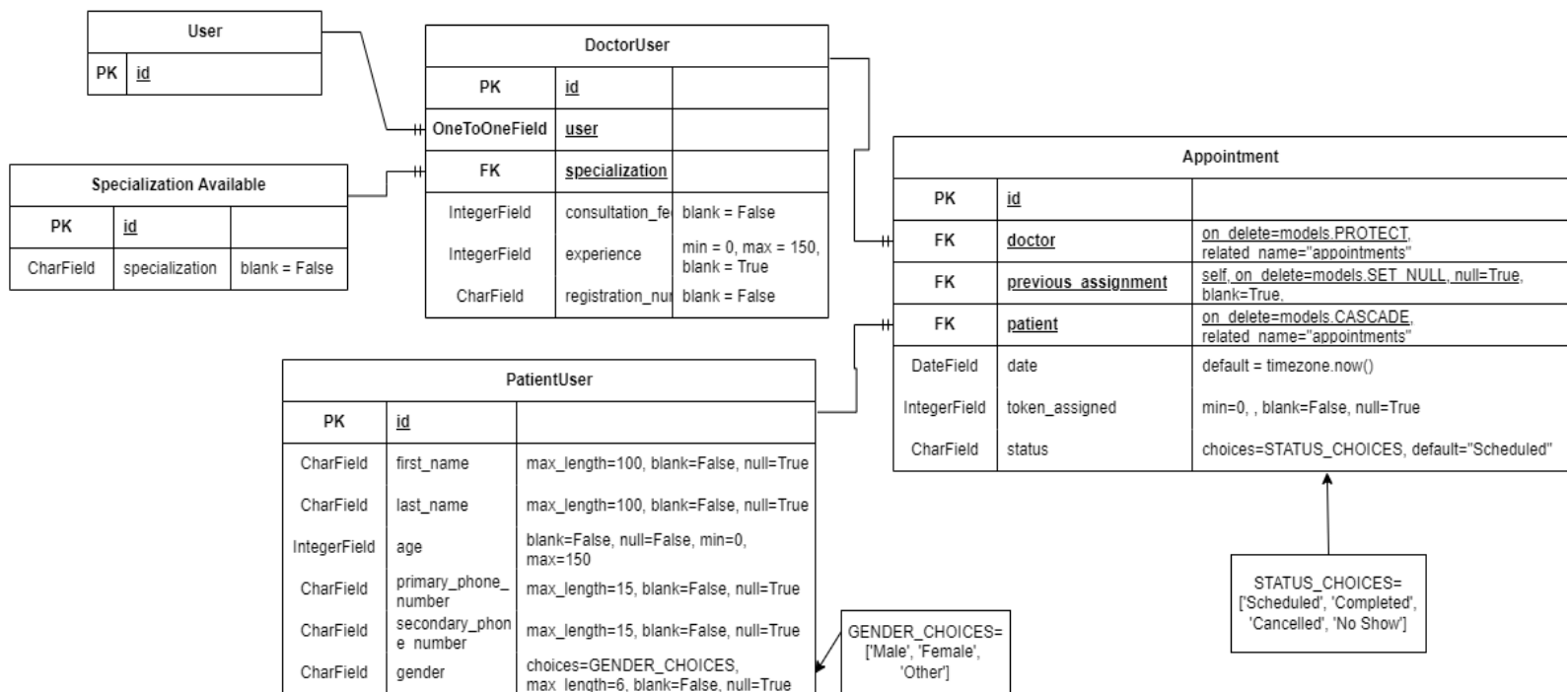


Figure 3: Other Users and Appointment Models

As can be seen in Figure 3, The **DoctorUser** model has a One-to-One Field relationship with the **User** class. This ensures that for a User with role="Doctor," there is only one entry in the **DoctorUser** model. The **DoctorUser** model defines more unique fields for a doctor user, such as consultation fee, experience, and registration number. The **SpecializationAvailable** model stores all specialization instances that can be assigned to a doctor. Therefore, the **DoctorUser** model has a foreign key reference to the available specialization model, which indicates the doctor's specialization.

The **PatientUser** model does not have a Foreign Key reference to the **User** model (Figure 3), as it does not need to have rights to access the application. Therefore, it implements its own fields, even though some fields have been repeated from the **User** model.

Both **PatientUser** and **DoctorUser** are also referenced as Foreign Keys in the **Appointment** model (Figure 3), which stores the details of a patient's appointment with a doctor. All 3 models are defined in the doctor application. The **Appointment** model stores the token assigned to the patient for that particular doctor, indicating the order. It also has a status field, that can only store values stored in the STATUS CHOICES variable. This field helps to track the progress of each record, maintaining consistency and preventing invalid entries.

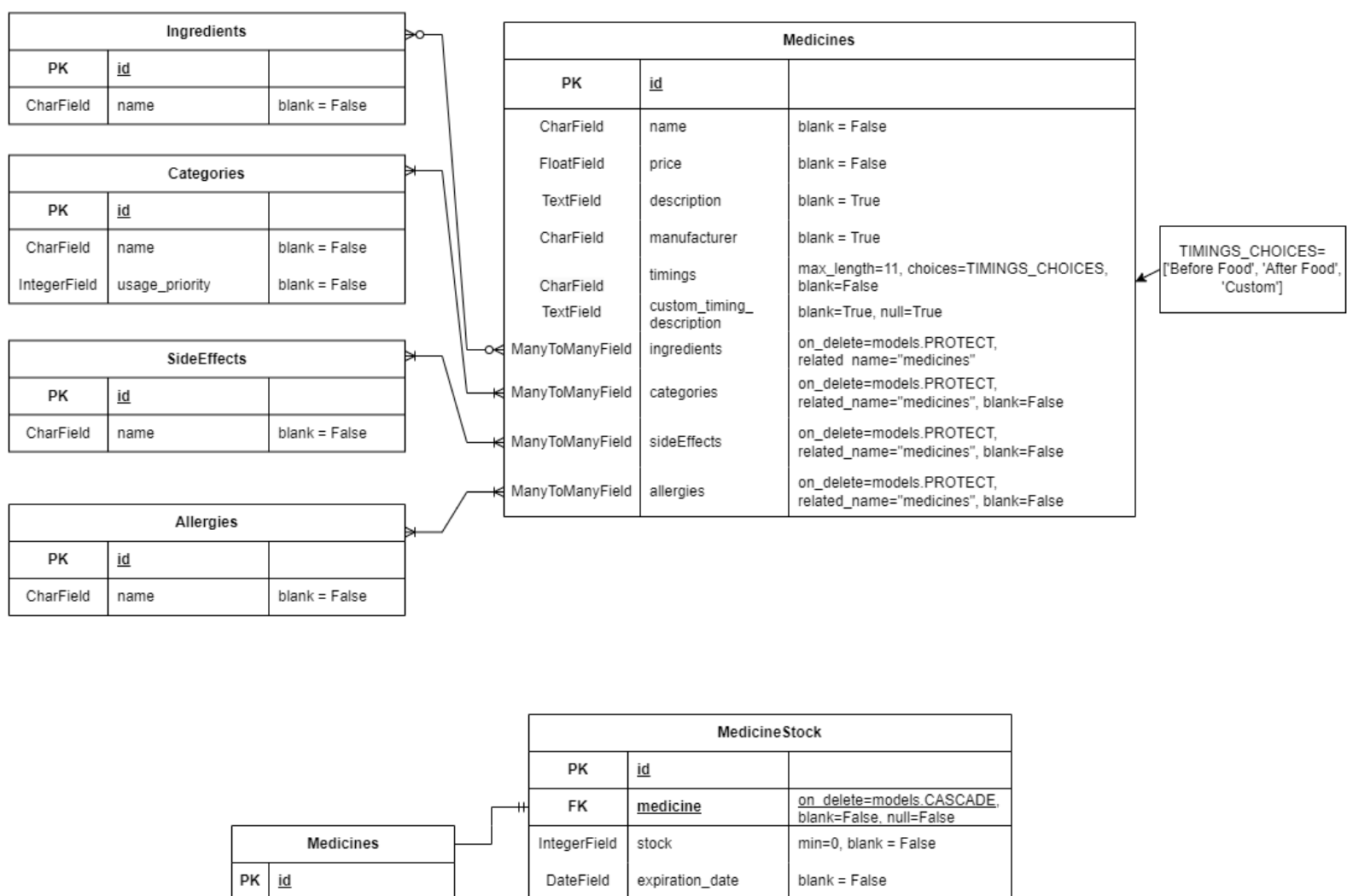


Figure 4: Medicine, MedicineStock and Related Models

The pharmacy application defines the model of **Medicine**, which establishes multiple Many-To-Many relationships with other entities in the database, such as **Ingredients**, **Categories**, **SideEffects**, and **Allergies** (Figure 4). A few models are used to inform the patient, such as **SideEffects** and **Allergies**, which will be mentioned in the final prescription for their convenience. The Many-To-Many relationship enables the required number of instances from the foreign model to be associated with one instance of the Medicine model.

To keep track of the stock available in the pharmacy and the expiration date for each of the stocks, a separate model by the name of **MedicineStock** is created (Figure 4). It has a Foreign Key reference to a **Medicine** instance, helping in keeping track of stock for each medicine.

LabTests		
PK	<u>id</u>	
CharField	name	blank = False
TextField	description	blank = True, null = True
FloatField	test_cost	blank=False
CharField	sample_required	max_length=100
TextField	pre_test_requirements	blank = True, null = True
CharField	provider	max_length=255, blank=False

Figure 5: LabTests models

LabTests model (Figure 5) defines all the Lab Tests that are being offered by the pharmacy. It contains some essential information, like what the test should be used for indicated by the description field, the price for taking the test, along with other information that will be useful for the pharmacist to successfully conduct the lab test.

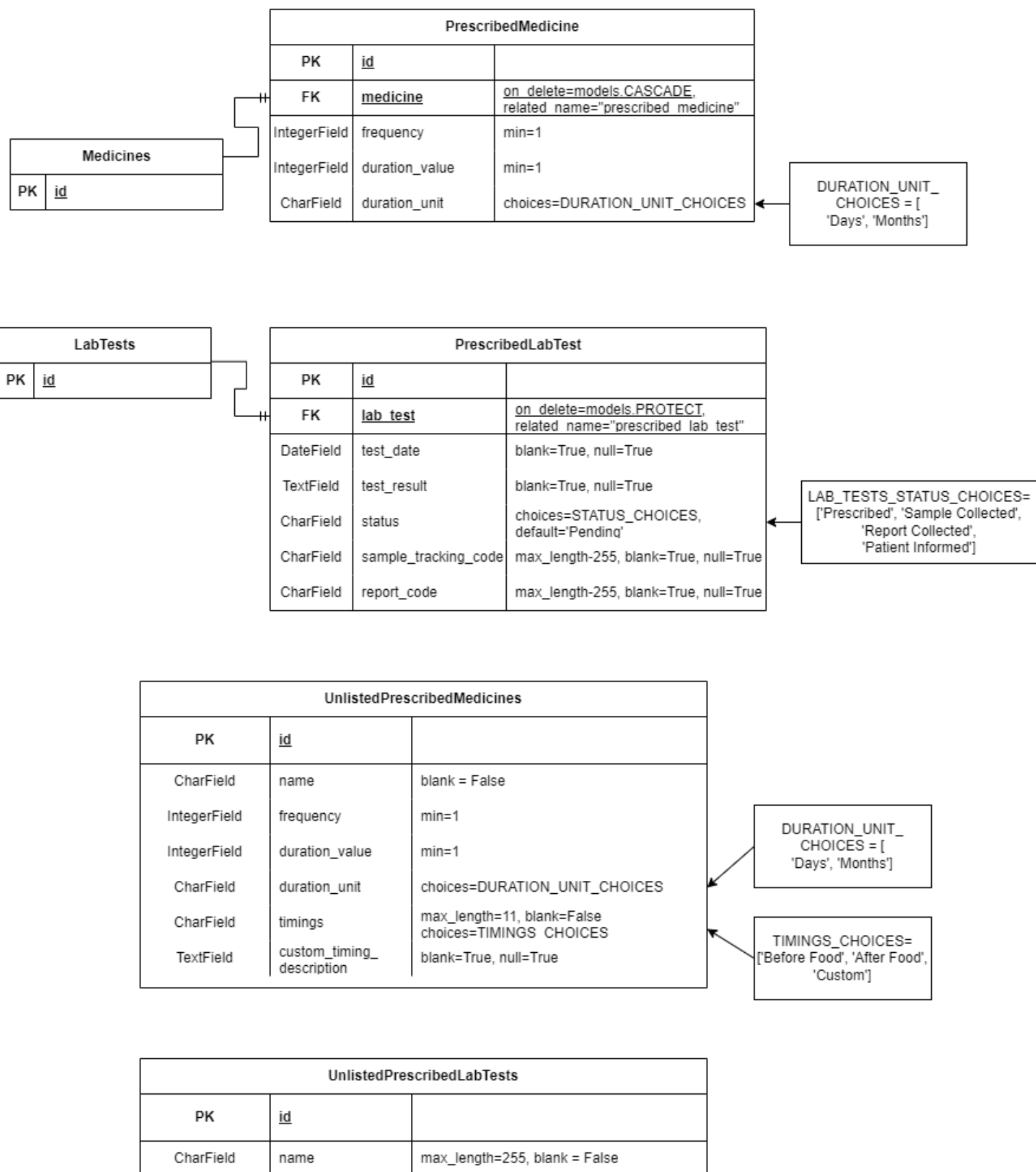


Figure 6: Models that are used for making a prescription instance

The doctor user when writing a prescription will create **PrescribedMedicine** and **PrescribedLabTest** models for each individual medicine and lab test respectively, as they extend the **Medicines** and **LabTests** model (Figure 6) to include more detail that is specific to each patient's requirement and their prescription. For example, a child under 5 years old will have a different dosage or frequency of consumption for a particular medicine than an adult for the same medicine.

At the same time, if the doctor wants to prescribe a medicine or lab test that is not available or provided by the pharmacy, the required information will be stored in **UnlistedPrescribedMedicines** and **UnlistedPrescribedLabTests** models respectively (Figure 6).

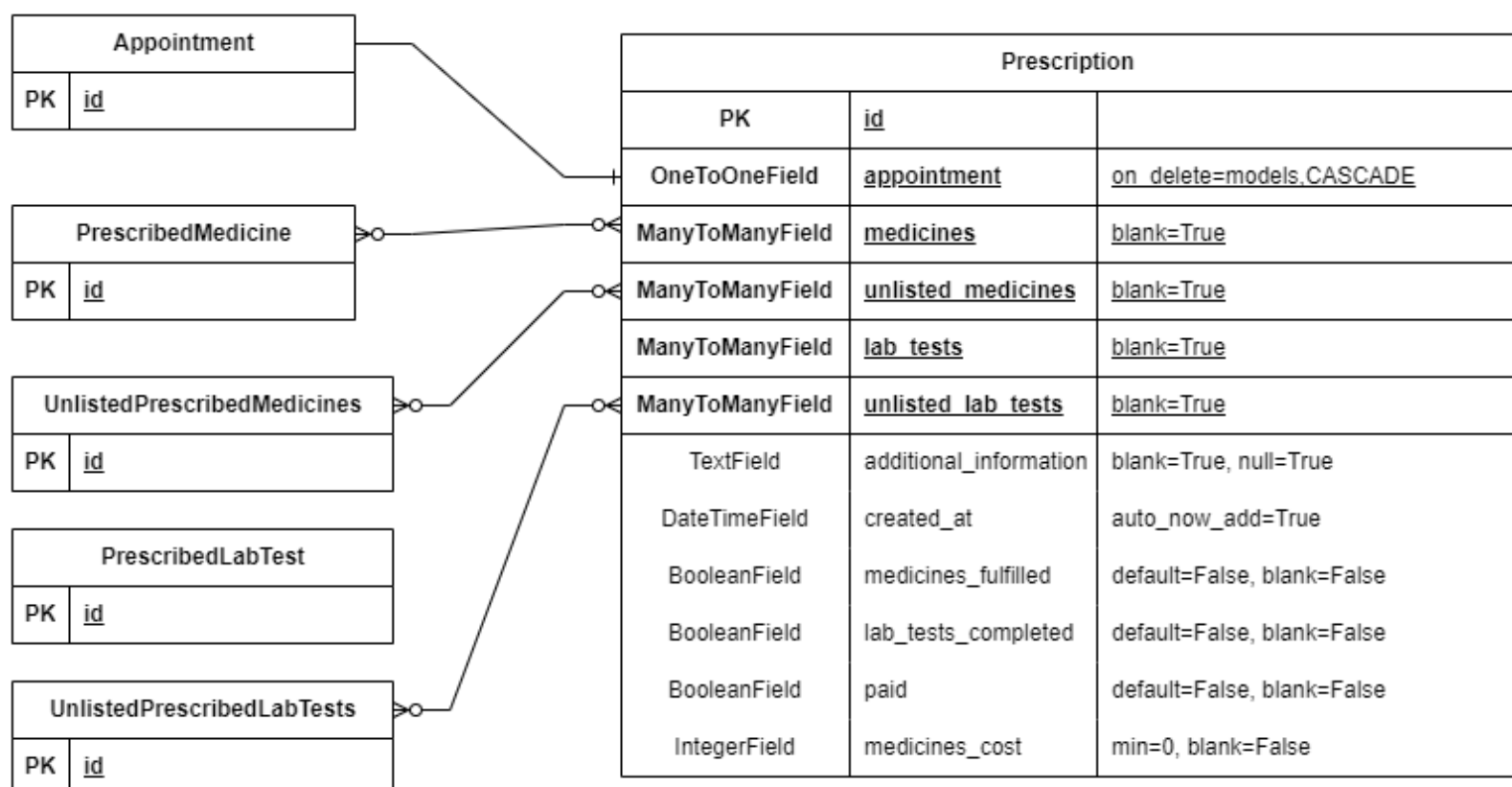


Figure 7: Final Prescription

The doctor when writing prescription for the patient will create an instance in the **Prescription** model. It has a One-To-One-Field reference to the Appointment model so that each appointment can have a prescription attached (Figure 7). It has Many-To-Many-Field references to **PrescribedMedicines**, **PrescribedLabTests**, **UnlistedPrescribedMedicines**, and **UnlistedPrescribedLabTests** (Figure 7), allowing doctors to prescribe both available medicines and lab tests from the pharmacy as well as those that are not in stock or are not

provided by the pharmacy.

Authentication, Frontend and Backend Connection

As the front and back end are deployed on two different servers and base URLs, there must be a communication method between the two points. This is provided by making API calls through HTTP requests, handled by Axios in the front end and the rest framework in the backend. JWT (JSON Web Tokens) is used to enable authentication. The logic can be demonstrated in the flowchart beside (Figure 8).

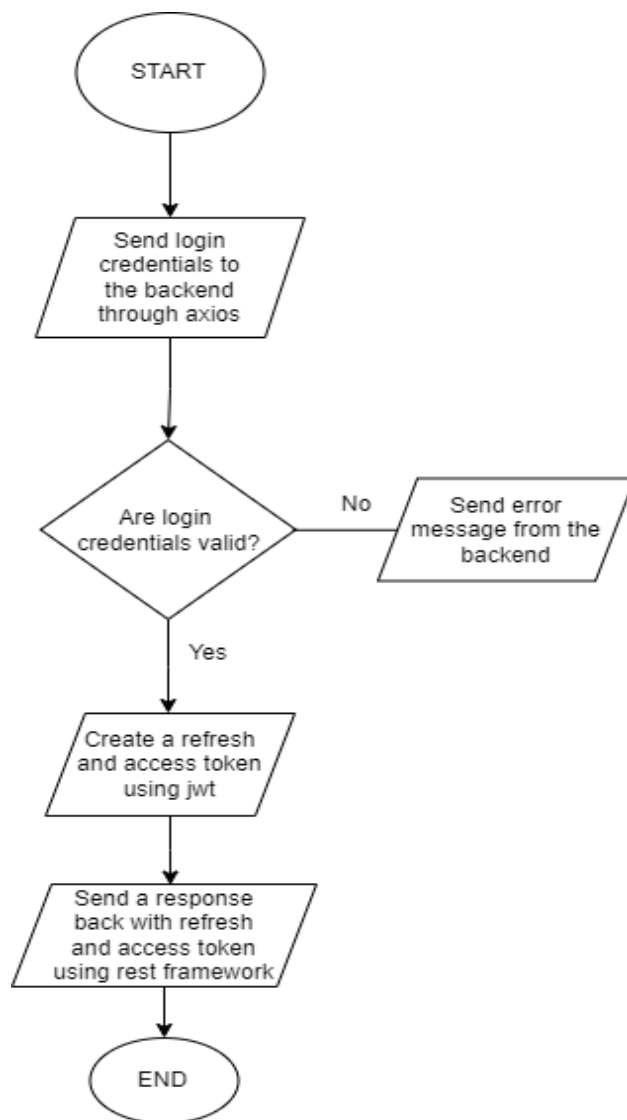


Figure 8: Authentication Logic

The authorisation process after authentication relies on two types of tokens:

1. **Access Token:** A short-lived token valid for 30 minutes used to authorise requests to protected endpoints i.e. endpoints which require user to be logged in and have the required credentials and permissions.
2. **Refresh Token:** A long-lived token with a 24-hour validity, used to obtain a new access token after every 30 minutes, when the current one expires.

Both tokens are generated in the backend using Rest Framework's SimpleJWT authentication.

The application of these two tokens is demonstrated in the flowchart below (Figure 9):

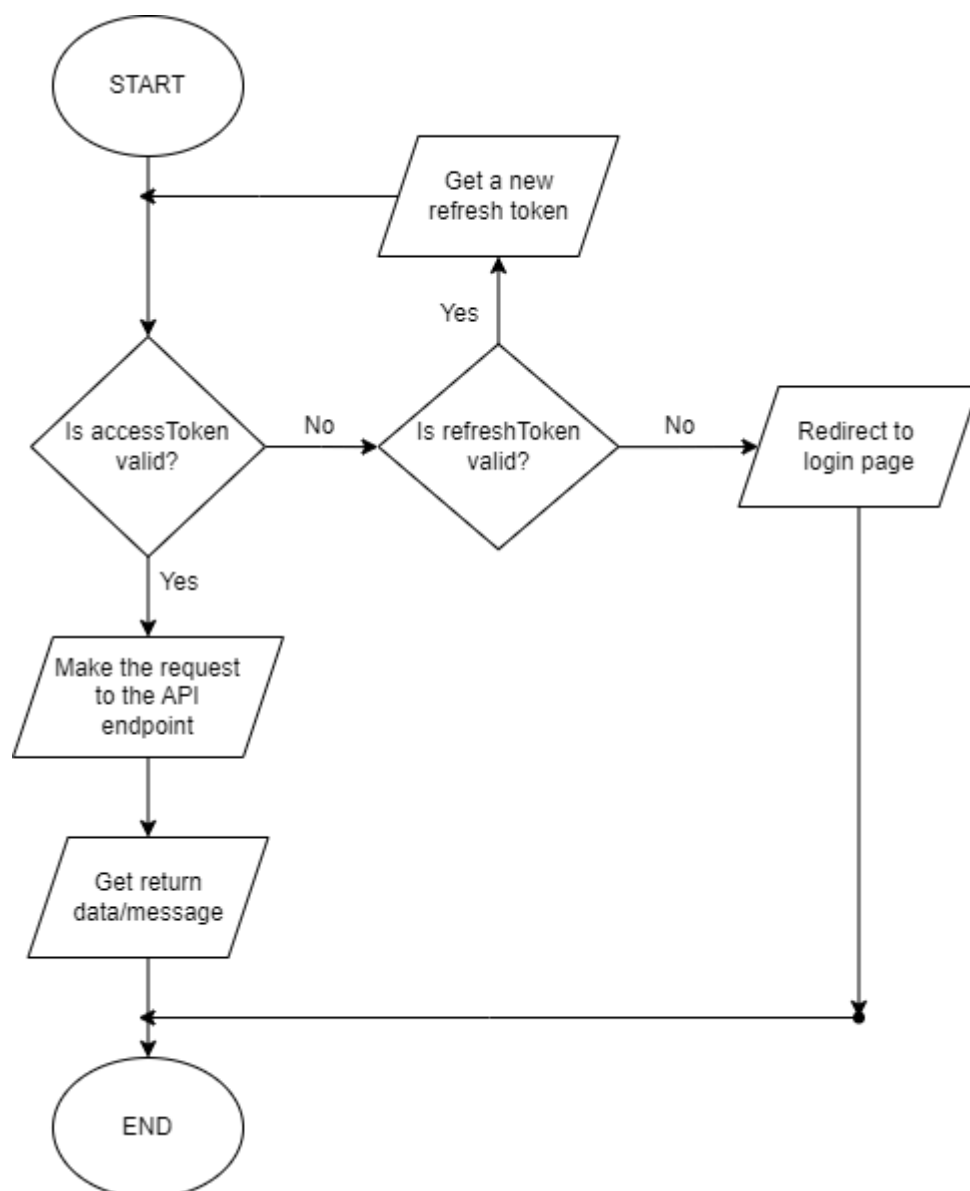


Figure 9: Authorisation Logic after Authentication

The access token must be included in the request headers for authorisation from the front end. This is also considered more secure than including the token in the body of the data packet. This is done with the help of interceptors, which help include additional logic before sending a request or after receiving a response.

In the backend, the access token is verified using Rest Framework's SimpleJWT Authenticator. If the token is deemed valid, the API endpoint is granted access. However, due to the short lifespan of the access token, the refresh token is used to get a new access token until it expires. When the refresh token expires, the user has to log in again. All these checks are also done using **Interceptors**.

Front end design

Menu Bar

The Menu Bar is located at the top of the page and shows commonly used features in the application for that profile. The Menu Bar has two states, one when logged out and the other when logged in, as showcased in Figures 10 and 11.



Figure 10: Menu Bar when logged out

Clicking the home button when the user is logged out will redirect the user back to the application's home page, where the user can view sign-in and log-in options for all profiles. It also has options to navigate to the profile's sign-in and log-in pages. For example, if the user selects the 'Doctor' profile on the home page of the application, clicking the sign-in and log-in button will redirect the user to sign-in and log-in pages for that application.

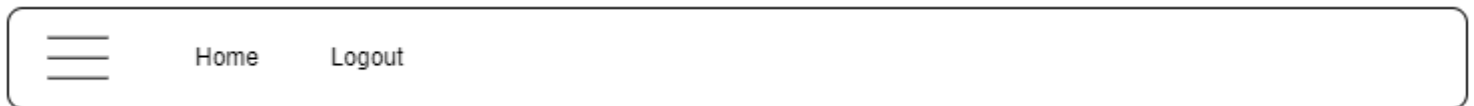
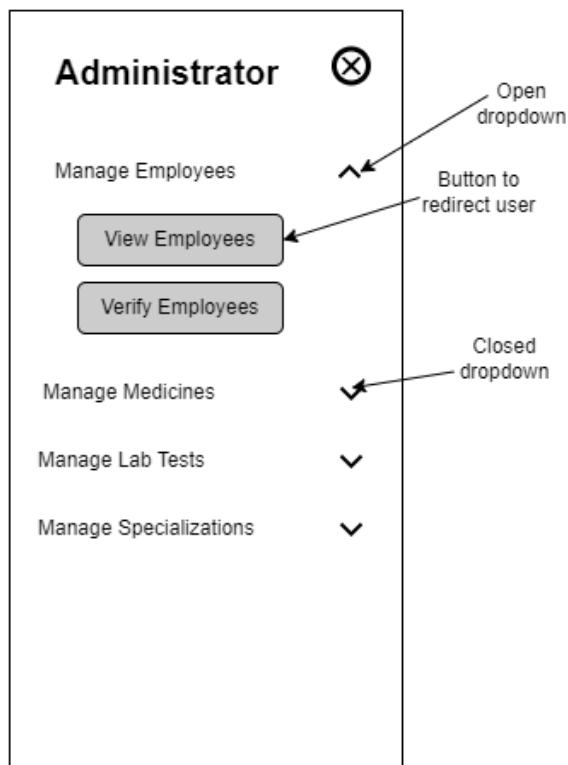


Figure 11: Menu Bar when logged in

Clicking the home button when the user is logged in will redirect the user to the home page of the profile the user is logged into. The triple bar can be used to access the drawer which can be used to access all parts of the application. The triple bars are only visible when the user is logged in. Additionally, the Logout button redirects the user to a common logout page

Drawer



The Drawer can only be opened from the Menu Bar by clicking on the triple bars. The drawer contains all the links to the features available for that profile, effectively categorised for easier navigation.

The adjacent drawer belongs to the Administrator application. The dropdown for employees is expanded, displaying buttons that allow the user to navigate to the 'View

Figure 12: Drawer

Employees' or 'Verify Employees' pages. For the closed dropdowns, only the header is shown. Similar logic has been followed for the rest of the profiles.

Home Page for the application

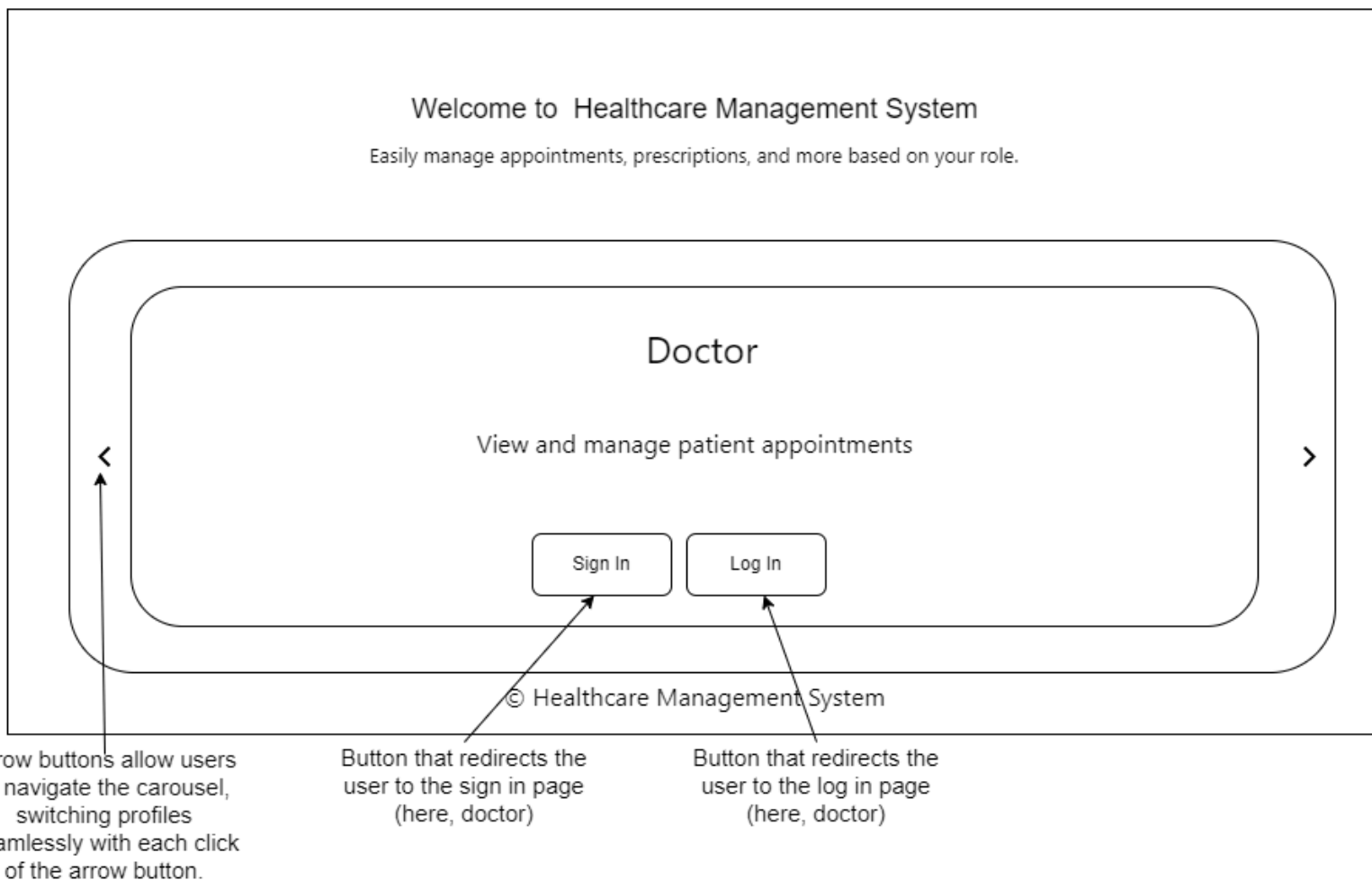


Figure 13: Home Page of the application

This is the first page of the application that any user uses. If any other part of the application is accessed without the proper authentication, the user is redirected back to this page. It provides a way to log in or to create a new user. As can be seen in Figure 13, the current slide from the carousel shows options to Sign In and Log In for the doctor user. The carousel can

be navigated using the arrow keys, and the slides also move at a specified interval. Except for the admin, where the only option is to Log In, all the other profiles have the option to Log In or Sign In.

Sign In Page

The Sign-in Page lets users create their account in the system and make a new entry into the database. It allows for future logins and access to the application. For all employees, verification by the administrator is required before further access to the application's features is granted. Upon a new user Sign-in, the administrator is notified and can verify the user to grant access or delete unrecognised accounts, maintaining the application's security.

Since the integrity of the application depends on administrators, the only way to make an administrator account is by inputting the user's details directly into the API Endpoint in a specific format. Only one administrator can be present at one point in the application. This minimises the risk of unauthorised administrator account creation. The GUI Layout for the admin application is demonstrated in Figure 14 and all the other users have a similar layout to the one shown in Figure 15.

Sign In

API View for administrator sign in

HTTP 405 Method Not Allowed
ALLOW: POST, OPTIONS
Content-Type: application/json
Vary: Accept

```
{  
  "detail": "Method \"GET\" not allowed."  
}
```

Media Type: application/json

Content:

This dropdown menu signifies the type of the inputted data


The details of the user can be inputted here in a JSON format

Figure 14: Sign In page for Administrator

This is the API endpoint for the administrator Sign-in, as provided by Django's APIView. The features of the admin, such as the Email ID, first name, last name, etc. has to be submitted in the Content section of the application. The data can then be posted, and will only be accepted if all the details are valid, and the syntax of the data sent matches.

Sign In

Gender* ▼



"Medicine is not only a science; it is also an art. It does not consist of compounding pills and plasters; it deals with the very process of life, which must be understood before they may be guided." - Paracelsus

Input Fields for the user to input all the fields. Required fields are marked with a star.

A Custom Password component which shows the strength of the password as the user starts typing

A special unique code assigned to a doctor by a regulatory licensing authority, which is part of the username

A Drop Down Menu for selecting one of the three gender options: Male, Female, or Other.

Figure 15: Sign In page for Doctor

Each field has a custom validation check to ensure the user's input is valid. For instance, the consultation fee for a doctor cannot be lower than 0. If the custom validation passes, a final validation is made to ensure all required fields (indicated by asterisk symbols) are filled up, and the password matches the confirmation password, along with a few others, to ensure minimal chances of error. On successful completion, a new entry is made to the database, and the user is redirected to the login page. However, the user has to wait for approval from the admin before proceeding with the application.

Login Page

The login page allows the user to authenticate his/her credentials for further access to the different parts of the application, as granted by the profile. Unlike the sign-in page, the GUI Layout for this page does not differ much from profile to profile, except for the doctor's and pharmacist's login page, where a unique registration number is assigned to them by a regulatory licensing authority. The Administrator, Front Desk, and Pharmacy login page is illustrated in Figure 16, followed by the login page for Doctor and in Figure 17.

The diagram illustrates the GUI layout for the Administrator and Front Desk Login page. The form is titled "Login" and is contained within a rounded rectangular box. The form is divided into two main sections by a vertical line. The left section contains the following fields and buttons:

- First Name**: An input field for specifying the first name of the user.
- Last Name**: An input field for specifying the last name of the user.
- Password**: A password field for inputting the password.
- Confirm Password**: A confirm password field for confirming the password.
- Submit**: A button to make a request to the backend to verify information.

The right section of the form contains a quote by Paracelsus: "Medicine is not only a science; it is also an art. It does not consist of compounding pills and plasters; it deals with the very process of life, which must be understood before they may be guided." - Paracelsus.

Figure 16: GUI layout for Administrator and Front Desk Login page

The diagram illustrates the GUI layout for the Doctor and Pharmacist login page. It features a central form titled "Login" with the following components:

- First Name**: Input field for specifying the first name of the user.
- Last Name**: Input field for specifying the first name of the user.
- Password**: Password field for inputting the password.
- Confirm Password**: Confirm Password field for confirming the password.
- Registration Number**: Unique Registration Number assigned that uniquely identifies a doctor/pharmacist to prevent users with the same username.
- Submit**: Button to make a request to the backend to verify information.

On the right side of the form, there is a quote: "Medicine is not only a science; it is also an art. It does not consist of compounding pills and plasters; it deals with the very process of life, which must be understood before they may be guided." - Paracelsus

Figure 17: GUI layout for Doctor and Pharmacist page

All the profiles have fundamentally the same logic. When the submit button is clicked, all the necessary validation is done, and then a request is sent to the backend with the login information. If the credentials are valid, JWT tokens are sent back to the front end for further HTTP requests and API calls.

Logout

All profiles are redirected to the same logout page, where a request is sent to the backend the blacklist the tokens from further use. If the token is successfully blacklisted, using the blacklisted token will return an error. However, on a successful request, the tokens are removed from the front end and require re-login to get a new access and refresh token.

Administrator Profile

The administrator profile is central to the application's operation. It possesses the highest level of permissions, from managing and verifying employees to auditing logs, charts, and performance graphs, which can provide valuable insights into the organisation's operations and overall performance. All the features granted to the administrator profile can be seen in Figure 18.

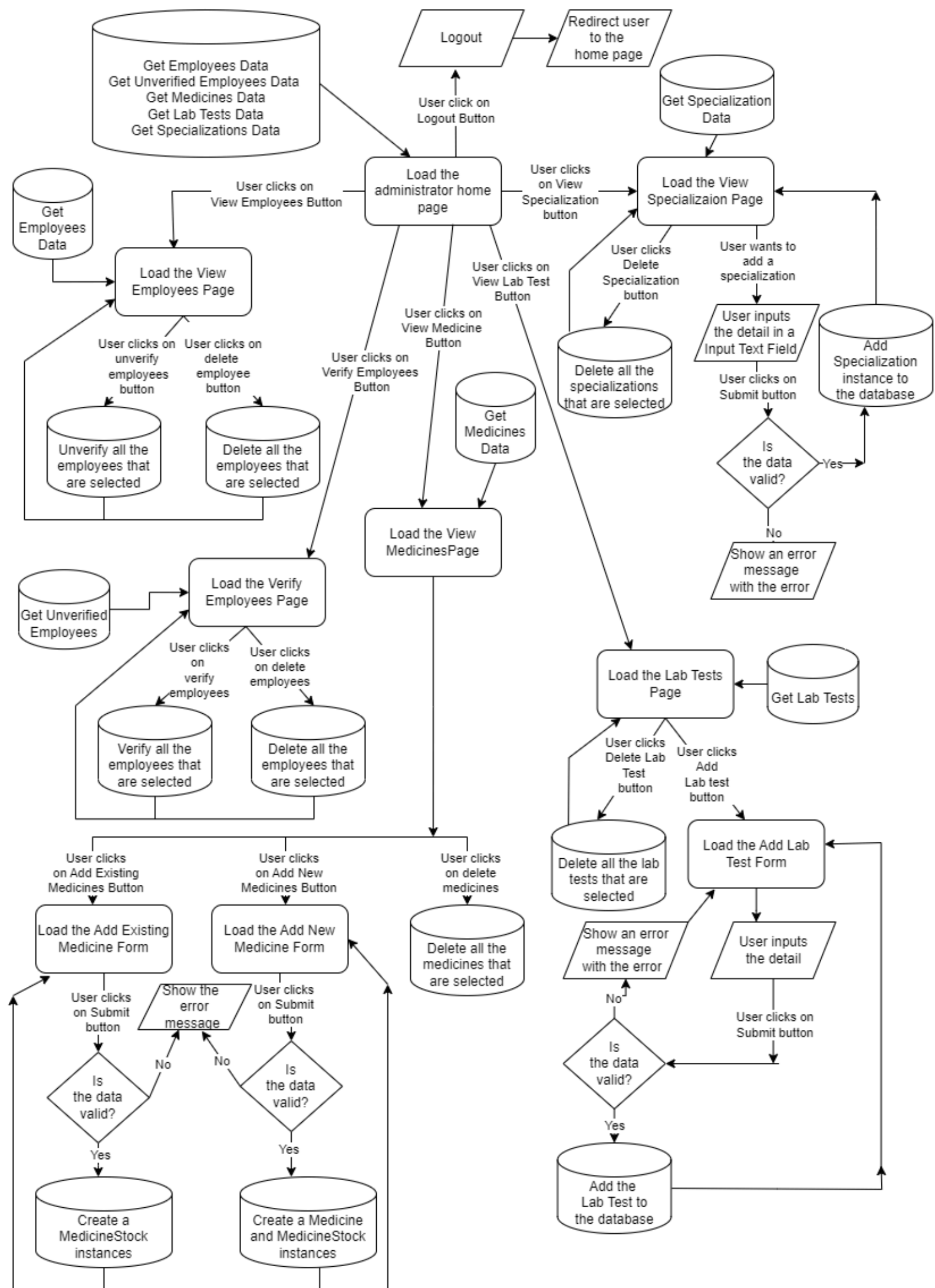


Figure 18: Administrator Profile Functionalities

Homepage for Administrator Profile

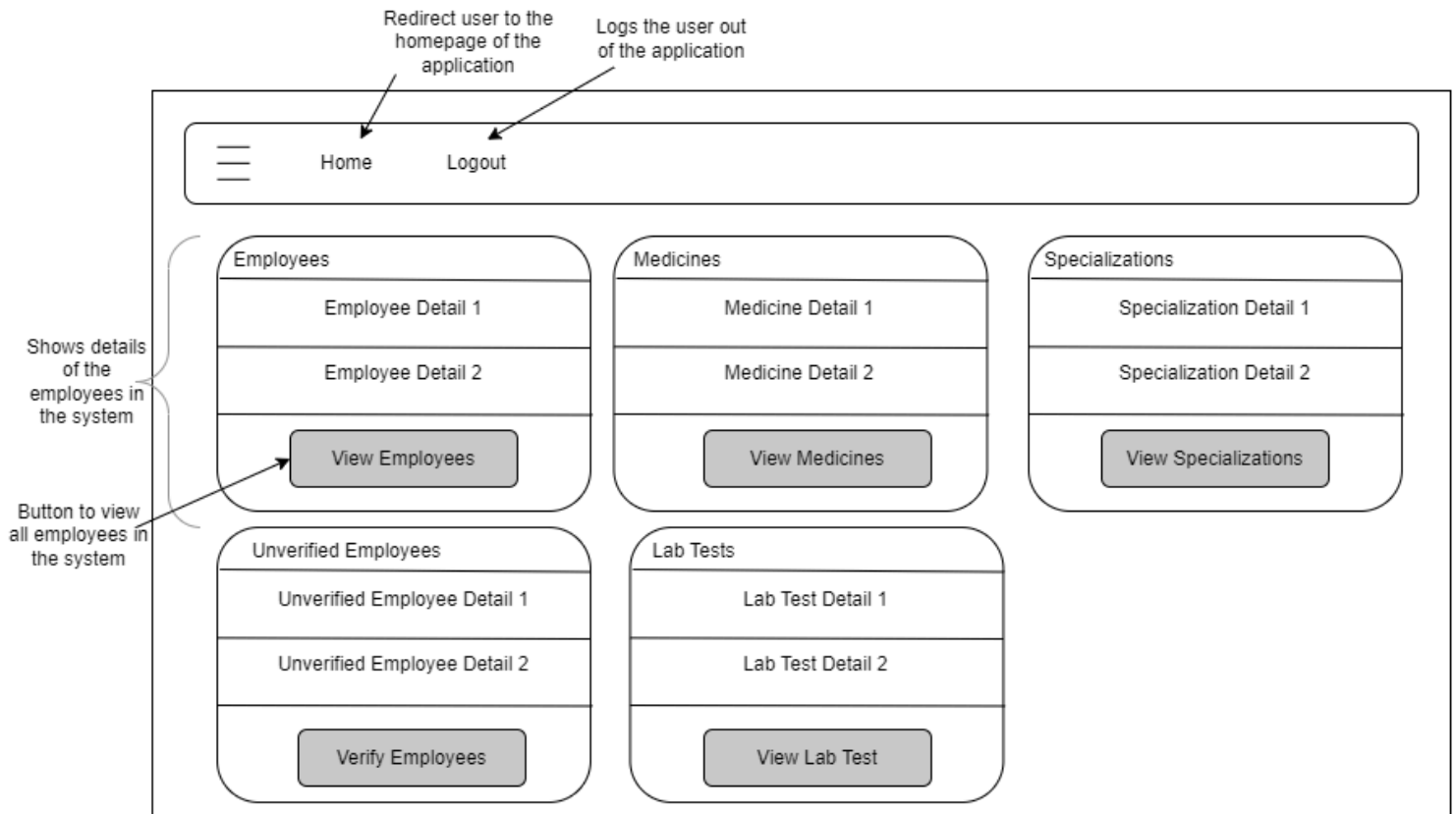


Figure 19: Home Page for the administrator profile

The home page for each of the profiles follows a similar layout to the one for the administrator showcased in Figure 19. According to the features and jobs assigned for each role, the tables and buttons on the homepage will vary. Clicking a button redirects the user to a dedicated page where they can perform their assigned tasks efficiently. Additionally, each profile's homepage is designed to provide quick access to essential features, ensuring a streamlined workflow for all users.

Pages in Administrator

All the pages in the application almost follow the same layout as the following pages taken from the administrator profile. While each of them has their own logic according to the requirements of the page, they maintain a consistent design and user experience, ensuring seamless navigation throughout the application.

View Lab Tests

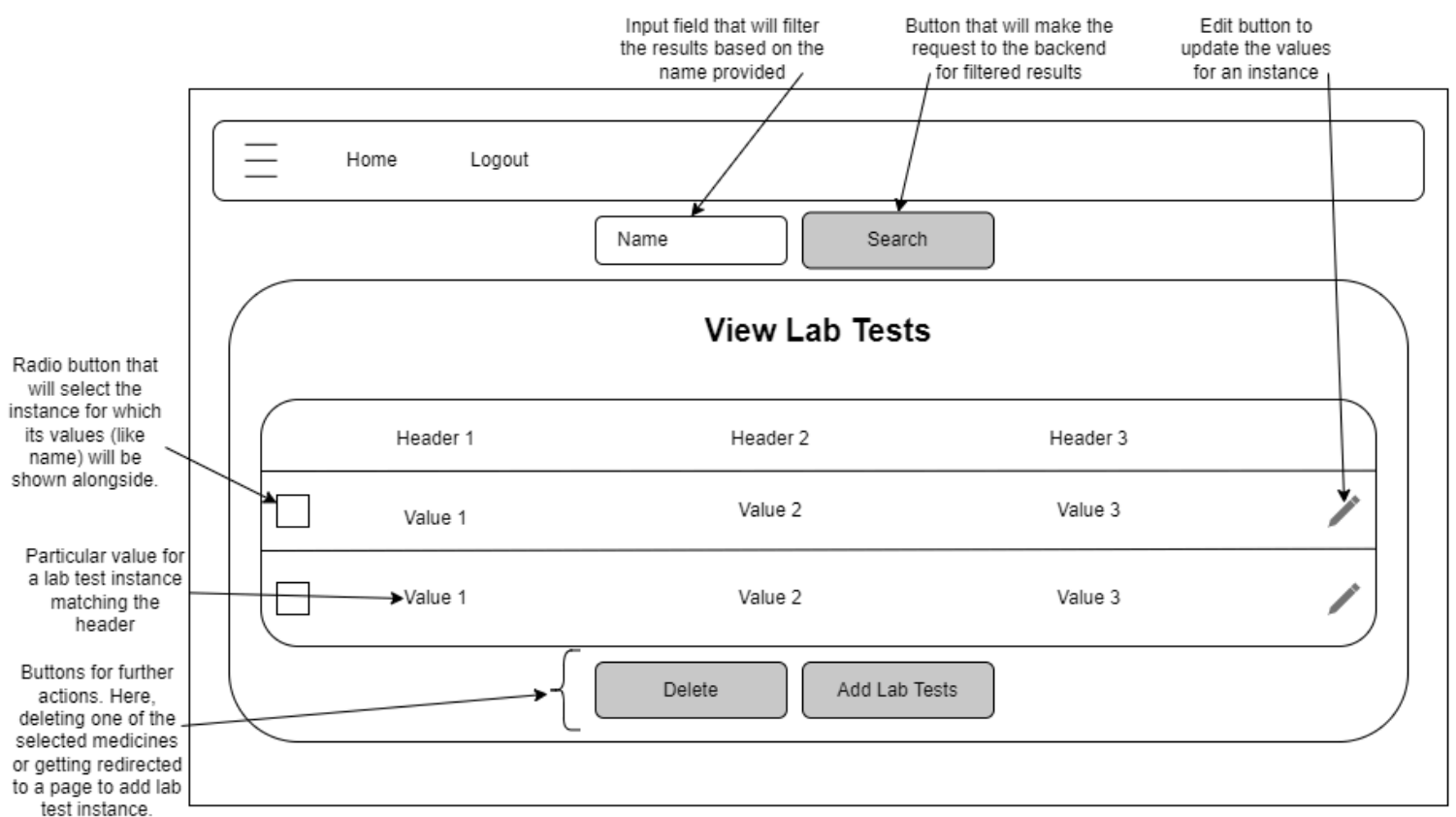


Figure 20: Page for viewing lab tests

All View pages have a search feature that can filter the results obtained from the database, as shown in Figure 20. Additionally, lab tests and specializations pages have edit features that can edit the details for a particular instance. Additional related pages are indicated at the bottom of the page, and the user can be redirected by clicking on those buttons.

Add New Medicines

Home Logout

Add Medicines

Ingredients

Ingredient 1

Add Ingredient

Allergens

Allergens 1*

Add Allergen

Categories

Category 1*

Add Category

Side Effects

Side Effects 1*

Add Side Effect

Submit

Different Input fields for receiving data from the user to create another medicine instance

When the submit button is clicked, the data is validated to make sure all the required fields are filled, and then sends the request to the backend.

Button that will create another Input Field for another entry of allergen.

The same logic is followed for the other buttons as well

Figure 21: Page for adding new medicines

Figure 21 shows the GUI layout for the page which can add a medicine instance to the database. In all the pages, the required fields are marked by an asterisk. To make sure all the required fields are filled, all the values from the required fields are checked for validity. If a required field is not filled, the user is informed accordingly. All 'Add' pages follow largely the same layout and structure to ensure a consistent user experience.

Add Existing Medicines

Input field that will filter the results based on the name provided

Button that will make the request to the backend for filtered results

Home Logout

Name Search

Add Existing Medicine

Select Medicines	Header 1	Header 2	Header 3
<input checked="" type="radio"/>	Value 1	Value 2	Value 3
<input type="radio"/>	Value 1	Value 2	Value 3

Stock * Expiration Date * Submit

Radio button from which only one can be selected

Details required along with the selected medicine to successfully add stock

Makes the necessary validation checks and sends the data for the backend on being successful

Figure 22: Page for adding stock of existing medicines

Figure 22 shows the ‘Add Existing Medicine’ page, which is used to add stock for a medicine already existing in the database. This reduces repetition as the details of the medicine need not be repeated again. The user has to select the medicine for which stock has to be added by using the radio button on the extreme left side of the data table. The user needs to add the number of units in the stock and their expiration date to be able to submit the request to the backend to create a new **MedicineStock** instance. Similar layout and logic can be seen in other parts of the application such as when creating an appointment for an already existing patient.

Doctor Profile

The **Doctor Profile** grants doctors exclusive access to make prescriptions for new prescriptions and view past prescriptions to ensure accuracy and continuity of patient care. With the ability to review comprehensive prescription histories, doctors can make informed decisions, reference prior treatments, and update medications or lab tests as needed. Any prescription instance can also only be accessed by the Doctor profile and Pharmacy profile for a limited time. The doctor profile can also only access prescriptions that was created using that profile i.e. a doctor cannot access prescriptions from other doctors. By combining functionality with strict access controls, the Doctor Profile supports enhanced patient care while maintaining the integrity and confidentiality of medical records. All the features granted to the Doctor Profile can be seen in Figure 23.

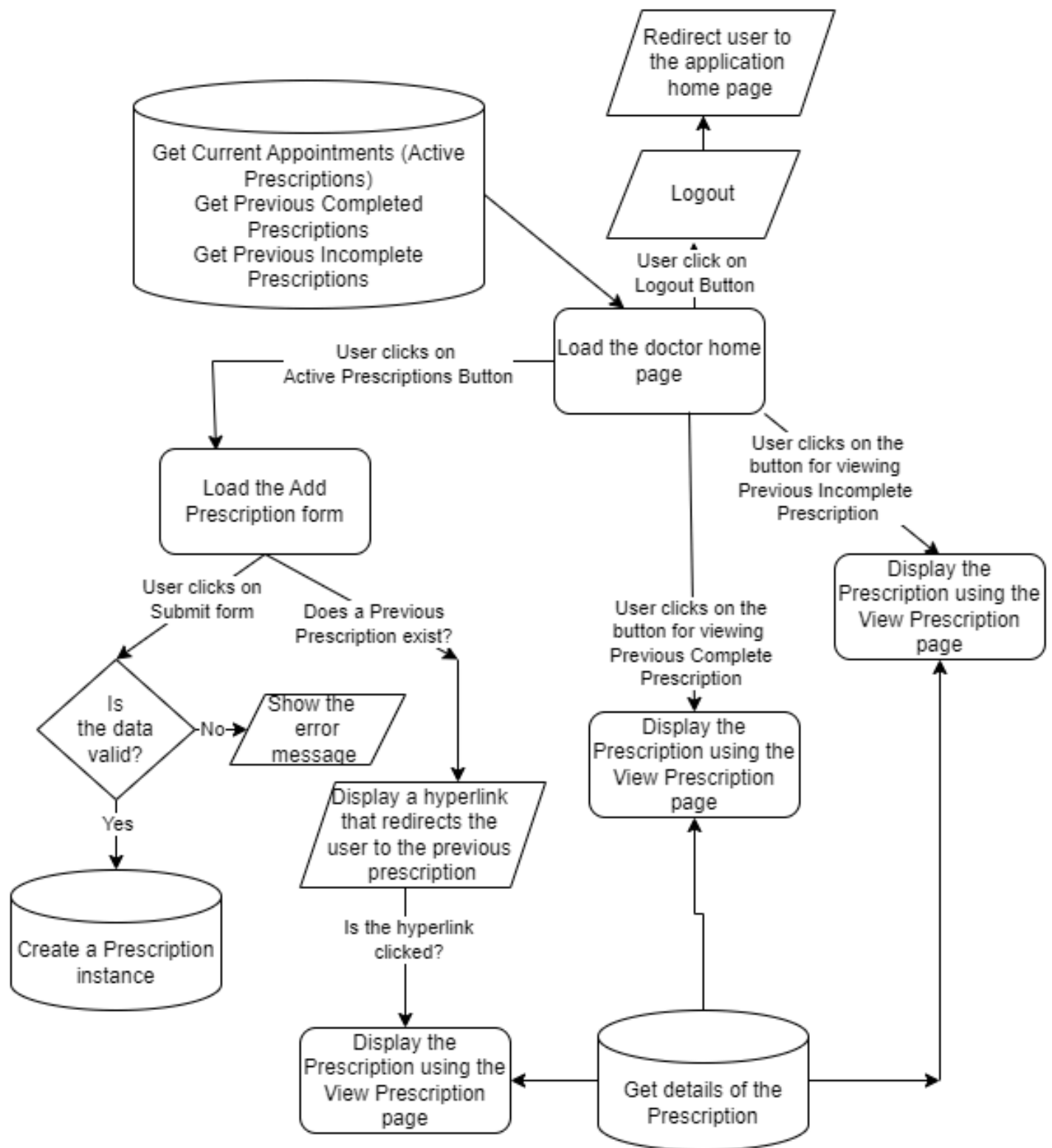


Figure 23: Doctor Profile Functionalities (The login functionalities largely remain the same as administrator except with its own separate page).

Front Desk Profile

The Front Desk Profile can be used to add details for new patients and create an appointment for them. It can also create appointments for existing patients. For already created appointments, it can rebook or cancel them or rebook no-show appointments. Additionally, this profile can collect payment for appointments that have pending payment.

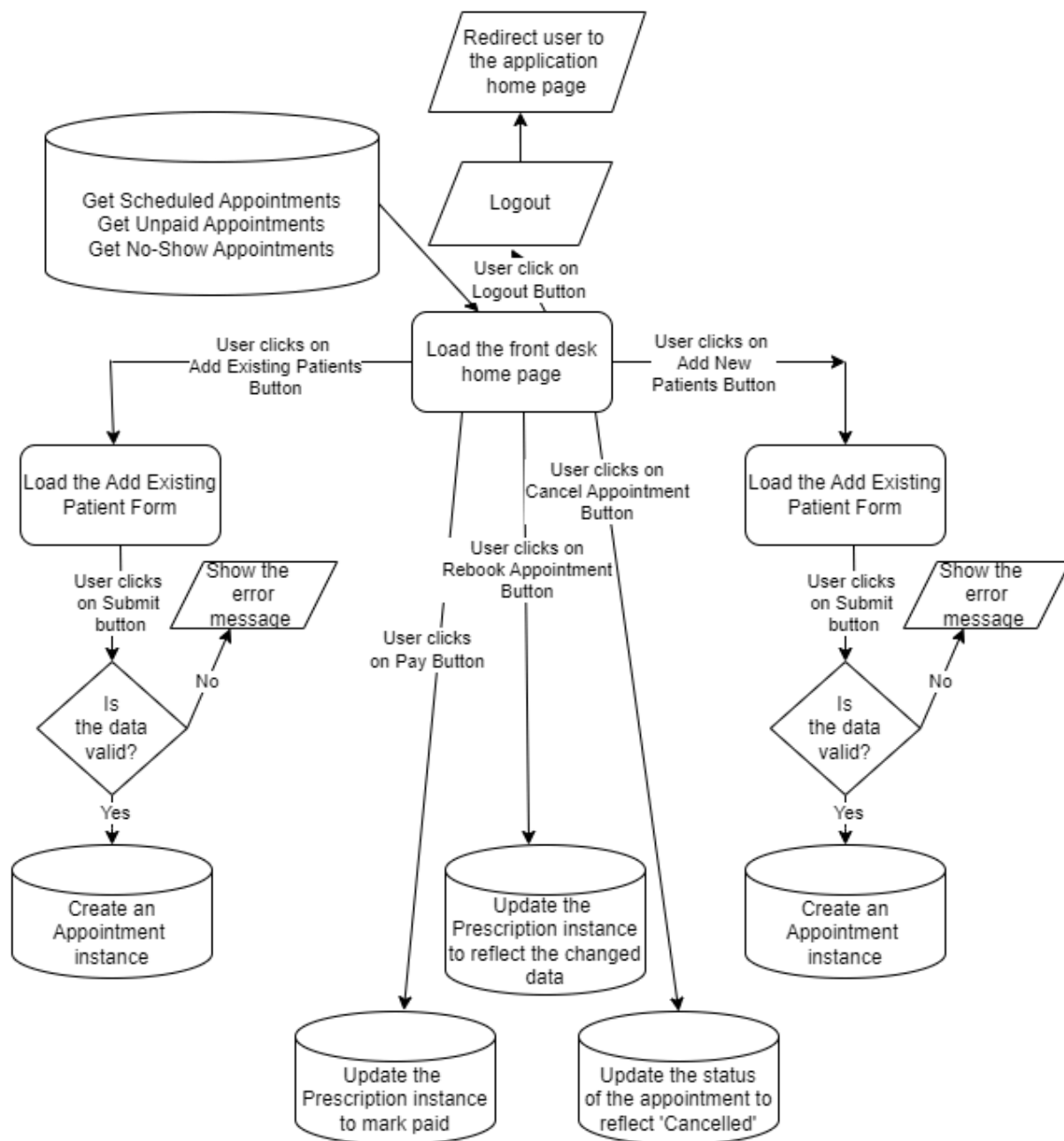
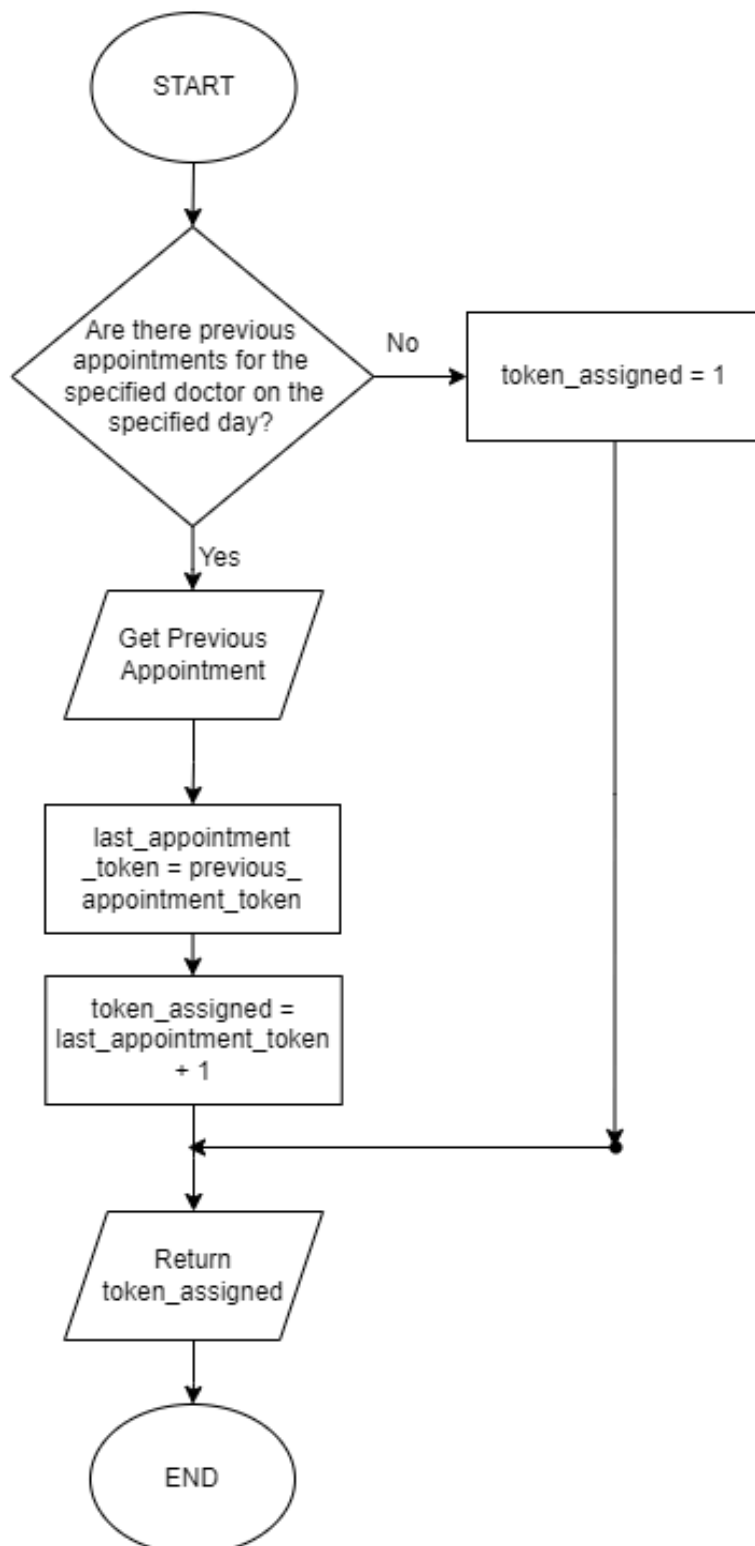


Figure 24: *Front Desk Functionalities (The login functionalities largely remain the same as administrator except with its separate page).*

Whenever a new appointment is booked, a token is assigned to that patient to establish an order. The token is generated from the back end and sent to the front end. The process behind generating the token is elaborated in the next section.



Token Creation Logic for Patient

The data that will be sent to the backend will have the information from the doctor, as the token number is independent for each doctor. Figure 25 alongside showcases the logic of creating tokens in more detail.

Figure 25: Token Creation Logic when creating an appointment

Pharmacy Profile

The Pharmacy Profile can be used to dispense medicines for the prescriptions written by the doctor. It can also collect data for prescribed lab tests. As soon as a new prescription is written, it is updated on the pharmacy application, so that appropriate actions can be taken. Additionally, the profile has the permission to edit the inventory or the medicines in the system to reflect accurate stock levels, ensuring that the inventory is up to date and preventing discrepancies in medication availability. Similarly, the profile can edit details for lab tests.

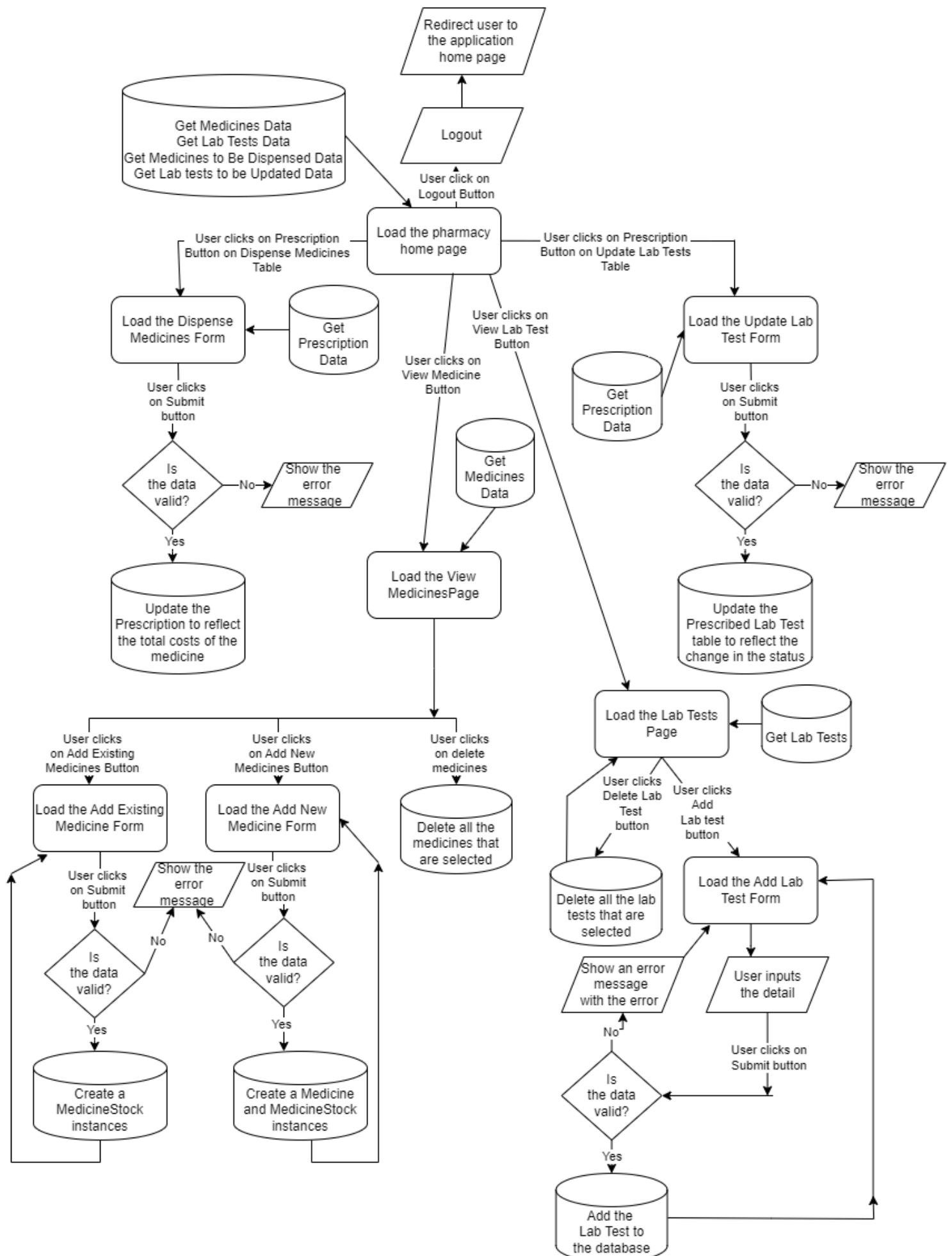


Figure 26: Pharmacy Functionalities (The login functionalities largely remain the same as administrator except with its separate page).

Test plan

Criteria Serial	Success Criteria	Test Serial	Testing	Expected Outcome
1	Profiles for the Front Desk, Pharmacists, Doctors, and the Admin	1	Each of the four user profiles should have dedicated pages with unique access methods. The features available to a user will be determined by their specific profile.	All profiles will have different URLs to access them separately.
2	Secure and distinct signin and login portals for all user profiles.	2	When the application is opened, there should be four options for Login or Sign In for different types of user, and they should be able to navigate to their respective profiles from there.	All the buttons required will be visible and usable, and the Login and Sign-in pages will be navigable to access the respective profiles.
2	Only the admin profile can verify and remove employees and edit doctor specializations.	3	The admin should get a message if a new user login is detected, and an option whether to verify or delete the employee from the database.	The employees must first input their details from their dedicated sign-in pages. The admin will be notified and can approve or delete the employee from the system. The verification status for the user in the database will be updated accordingly.
		4	A profile cannot be used for login until it is verified.	The new user will not be able to log in unless the admin verifies it.
		5	A profile can be used for login if it is verified.	The new user will be able to login into his/her profile page if it is verified by the user.
		6	The admin should be able to delete employees from the system.	The button will be visible to delete the respective employee from the system. If deleted, the entry will be deleted from the database and the credentials cannot be used for login.
		7	The admin should be able to add, edit or delete specializations that can be assigned to a doctor.	Appropriate buttons and fields should be visible to add, edit, or delete specializations. Relevant changes will be made to the

				specialization instance in the database according to the action taken.
3	Admin and pharmacy profiles can view and edit medicines and inventory, and lab tests.	8	Both admin and pharmacy profiles should have the option to view medicines and lab tests available in the system.	Both pages will have the option to view all the medicines and lab tests that are registered in the system.
		9	Both admin and pharmacy profiles should be able to add and delete medicines.	Working buttons and fields (if required) will be placed to complete the respective actions. Upon clicking the button and verification from a dialog box, the appropriate changes will be reflected in the database.
		10	Both admin and pharmacy profiles should be able to add stock for a particular medicine.	
		11	Both admin and pharmacy profiles should be able to add, edit, and delete lab tests.	
4	The front desk profile can book appointments for new and existing patients.	12	The front desk profile should be able to enter details for new patients to book their appointment.	Appropriate fields will be placed for the input of details of the patient, along with their preferred doctor. On clicking of the submit button, a new appointment should be created in the database, and should also be visible in the selected doctor's profile.
		13	It should be able to assign appointments to existing appointments	Option to search and select for existing patients in the database, select them and assign a new appointment for them to a doctor on a particular date. The changes will be reflected in the database.
5	The doctor profile can write a prescription for booked appointments.	14	The doctor should have an option for viewing the appointments that is booked for his/her profile, with a button to write prescription	There will be a way to view all the prescriptions appointed. Beside the listed appointment, there will be a button that is going to allow the doctor user to write a prescription for the selected appointment.
		15	The doctor should be able to write prescription for appointed patients.	There will be fields to prescribe medicines, lab tests, and additional information for the patient. Upon submitting, a prescription instance will be created in the database.

6	The doctor's profile allows access to the prescription at all times, and the pharmacy profile allows access for some time, abiding by medical rules and regulations.	16	The doctor will have access to previous completed prescriptions and new uncompleted prescriptions, and can be accessed at all times.	The doctor profile will be able to view prescriptions previously prescribed by him/her, and can write prescriptions for new appointments.
		17	The pharmacist can only access prescriptions for which medicines have to be dispensed or data needs to be collected for the lab tests.	The pharmacist profile will be able to view pending prescriptions for which medicines need to be dispensed, or sample/data need to be collected.
		18	If the patient pays for his/her appointment in the front desk, the pharmacist will not be able to access that prescription anymore.	The pharmacy will not be able to access prescriptions that have been paid through the front end profile. The database should be updated accordingly.
7	The pharmacy profile can dispense medicines and enter the required data for lab tests.	19	There should be an option to dispense medicines from the pharmacy profile. It should redirect to a page for the pharmacy can dispense the required medicines.	Upon dispensing medicines, the database should be updated accordingly to reflect the change is stock for the medicines prescribed and on the status of the prescription.
		20	There should be an option to collect details pending lab tests. It should redirect to a page where the user can enter details for the lab tests.	Upon collecting the details of the lab tests, the database should be updated accordingly to include the new information and changed in the status of the lab tests.
8	Status trackers for prescribed medicines, lab tests and prescriptions.	21	For prescribed medicines and each lab test, there should be status trackers that will keep track of the progress. It should be reflected in front end accordingly.	After each update to status of prescribed medicines/lab test from the pharmacy profile in the front end, the appropriate changes will be made in the database. If all medicines are updated/dispensed and data for lab tests are collected, the prescription can be marked as complete.
9	All required fields must be filled out before submitting; to inform the user	22	All pages should be checked if the data is successfully submitted even when the required fields are not present.	If the required fields are not present, an appropriate error message should be displayed.
		23	All pages should be checked if the data is successfully submitted when the optional fields are not present.	Even if the optional fields are not present, the data will still be submitted successfully.

	otherwise appropriately.			
10	Sessions will automatically expire after a set period, requiring re-login.	24	After a day of login, the user will not be able to access the page anymore.	The data will fail to load from the backend.
		25	If a day has passed since login, the user will be redirected to the login page.	After a day since login, the data will fail to load from the backend, redirecting the user to their respective login page.
11	Structured way of generating tokens at the front desk, for an ordered way of writing prescriptions by the doctor.	26	Sequentially, tokens should be generated for a particular day for a particular doctor.	The first token number generated for a day will start from 1 and then increment by 1 every time a new patient is entered for that doctor. For a different doctor, the token number will again start from 1.
		27	The changes should reflect in the doctor page.	The doctor will be able to see the name and token number of the patient.
12	The system will calculate the total costs for an appointment based on consultations, lab tests, and medicines taken.	28	In the front desk profile, the costs should be accurately calculated and displayed for all unpaid appointments.	There will be a view where the front desk profile can see the total costs to be paid by a patient and have the option to mark it as paid. The appropriate changes should be made in the backend.
13	The system will display clear error messages and success	29	For every action on the database done by any user in the application, the user will be kept informed if the action was a success or a failure.	There will be appropriate messages displayed on the screen every time a user makes a change to the database.

	notifications for user actions to keep them informed.	30	For every action to retrieve data from the backend, the user will be informed if the action was a failure.	If the data could not be collected from the database, an error message will be displayed.
--	---	----	--	---