SOFTWARE REQUIREMENTS SPECIFICATION

PREPARED FOR

Prof. Salman Abdul Moiz

PREPARED BY

Yasaswini Tiramdas

18MCME22

Table of Contents

1. Introduction

1.1. Purpose

The purpose of this document is to give a detailed description of the requirements for a Voice prescription application, enabling doctors and patients to create e-prescriptions. This application is necessary as it eases the paperwork for doctors and helps them focus more on the patient's needs and requirements. Because of the global pandemic, medical consultations are being done virtually either via phone or video calls. This application makes it easy for a doctor to take note of symptoms, diagnose, prescribe, and advise quickly without any hassle, virtually, and can directly send it to the patient.

The patient can have access to their medical consultations and prescription history. And these e-prescriptions can be easily carried and stored. This is opposite to the current situation where patients have to maintain physical files to store prescriptions and their test reports.

1.2. Scope

The "Voice Prescription" is a web application for doctors and patients. Where they can interact and doctors can diagnose the patient. The main focus is to generate a formatted prescription based on dictation from the doctor using keywords. The doctor would be able to edit/delete any entry that has been made in case there are any mistakes. The doctor will be able to preview the prescription and pre-sign the prescription digitally. After the diagnosis, the patient receives the e-prescription. The patient can also book an appointment before consultation using the application.

1.2.1. In-Scope

- Login system for Doctors.
- Login system for patients.
- Patients can book an appointment with the doctor.
- Doctors provide a formatted prescription.
- Send the prescription to the patient.
- Facility to pre-sign the prescription digitally.
- Storing the medical history of a patient.
- Patients have access to their medical history.

1.2.2 Out-Scope

- Mailing the prescription to the nearby pharmacy in case of an emergency.
- Suggestions for various pharmacies depending on the availability of the drugs prescribed.
- Provide suggestions for diagnosis to the doctor based on the input symptoms.
- Provide information to the patient about the availability of the doctor for surgery or consultation.

1.3. Definitions, Acronyms, & Abbreviations

Table 1 - Definitions

Term	Definition
User	Someone who uses the web application.
Doctor	Someone who writes the prescription and signs it.
Patient	Someone who receives the diagnosis and prescription.
Web-Portal	A web application where the prescription is written by doctors and read by patients.
E-Prescription	An Electronic prescription that can be opened/viewed in mobiles or Computers
DESC	Description.
RAT	Rational
DEP	Dependency.
GIST	A short, simple description of the concept containing a programming language statement.
SCALE	The scale of measure used by the requirement contained in a language statement.
METER	The process or device used to establish a location on a SCALE contained in a language statement.
MUST	The minimum level required to avoid failure contained in a programming language statement.

DEFINED	The official definition of a term contained in a programming language statement.
PLAN	The level at which good success can be claimed contained in a language statement
WISH	A desirable level of achievement that may not be attainable through available means contained in a language statement

1.4. References

1. IEEE Recommended Practice for Software Requirements Specifications. IEEE Std 830™-1998(R2009). (Revision of IEEE Std 830-1993).
2. https://pypi.org/project/SpeechRecognition/

1.5. Overview

The remainder of this document includes three chapters and an appendix. The second one provides an overview of the system functionality and system interaction with other systems. This chapter also mentions the system constraints and assumptions about the product.

The third chapter provides the functional requirements specification in detailed terms and a description of the different system interfaces. Performance requirements along with design constraints and software system attributes are also discussed.

The Appendix at the end of the document includes the Use Case Specifications in detail.

2. Overall Description

This section gives you an overview of the whole application including the constraints and assumptions. The application will be explained with the basic roles and functionalities of each role.

2.1. Product Perspective

This application will basically have two roles: doctor and patient. The doctor can write prescriptions based on the diagnosis s/he made. The patient can access his/her prescriptions and book an appointment with the doctor.

The doctor will first create an account by inputting his/her details such as name, specialization, designation, qualification, contact details, etc. After successfully creating an account they can confirm the appointments booked by a patient and schedule a consultation. During the consultation, the doctor will be able to write a prescription based on the keywords they say and sign it digitally. Before the prescription is submitted the doctor has a chance to edit the errors if any. If the patient had a consultation before this, the doctor will be able to access his older prescriptions.

The patient would also create an account by inputting their details such as name, email, age, etc. After the successful creation of an account, they can book an appointment with the concerned doctor. After the consultation, the patients will get a digital prescription. The patient will be able to access all their old prescriptions.

2.2. Product functions

The functionalities of the application/web portal are as follows:

- Manage Account
  Both user roles Doctor and Patient can manage their accounts, that is, create an account and login to their account.
- Manage Prescriptions
  The Doctor can create a prescription, and it cannot be edited once saved.
- Get Patient's History
  The Doctor can view the patient's older prescriptions.
- Book Appointment
  The Patient can book an appointment with the doctor.

- Get Prescription
  The Patient will receive the prescription once the doctor saves it.

2.3. User characteristics

There are two types of users for this application: Doctors and Patients. Each of these users has different uses of the application so each of them has its own requirements.

Doctors must be able to know how to use the internet and a web portal. They should be able to log in to their accounts and schedule appointments with the patients. During an appointment, they should be able to understand the functionality of the voice typing, edit and saving before ending the consultation.

Patients should be able to know how to use the internet and a web browser. They should be able to log in to their account and view the correct prescriptions they want.

2.4. User constraints

The Internet connection is a constraint for the application. Since the application gets and saves the prescriptions over the Internet, it is crucial that there is an Internet connection for the application to function.

Doctors must have a valid practising certificate and qualification.

2.5. Assumptions & dependencies

No specific assumptions and dependencies.

2.6. Apportioning requirements

If the application is widely accepted, we can incorporate the out-scope features into the existing application.

3. Specific Requirements

This section contains all of the functional and quality requirements of the system. It gives a detailed description of the system and all its features.

3.1. Interface requirements

3.1.1. User Interface

As soon as the user logs in, he/she will be able to see the login page. If the user has not registered, he/she should be able to do that on the log-in page. Every user should have a profile page where they can edit their name and password.

The doctor will be able to confirm appointments and schedule them on one screen, they will also be able to write, edit and save the prescriptions. They will be able to see the patient's history if it's available.

The patient will be able to book an appointment with their desired doctor. They will also be able to view all their prescriptions on one page.

The screen captures are added in Appendix 1.

3.1.2. Hardware Interface

Since the web portal does not have any designated hardware, it does not have any hardware interfaces.

3.1.3. Software Interface

Since the web portal does not have any other software usage, it does not have any software interfaces.

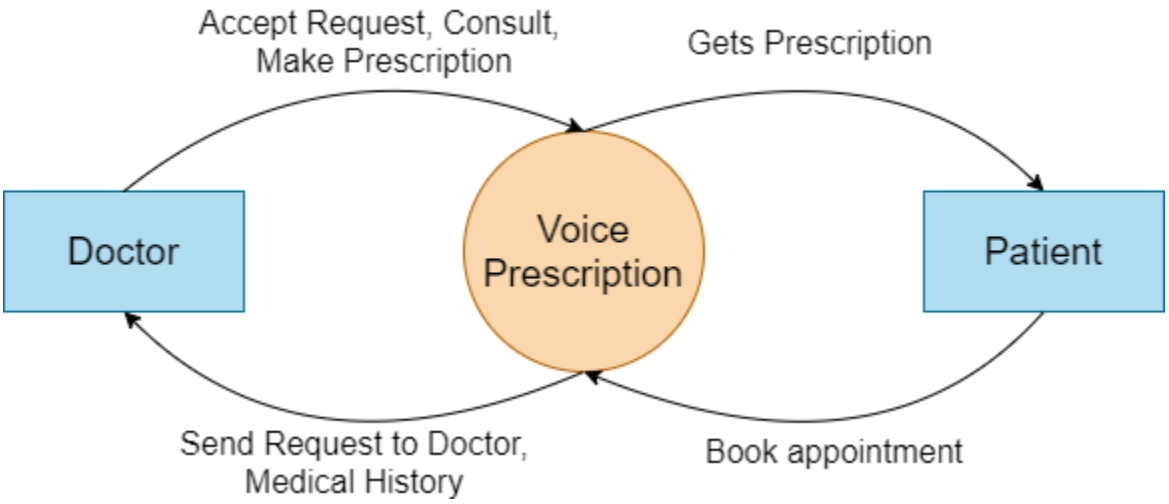
3.1.4. Communication Interface

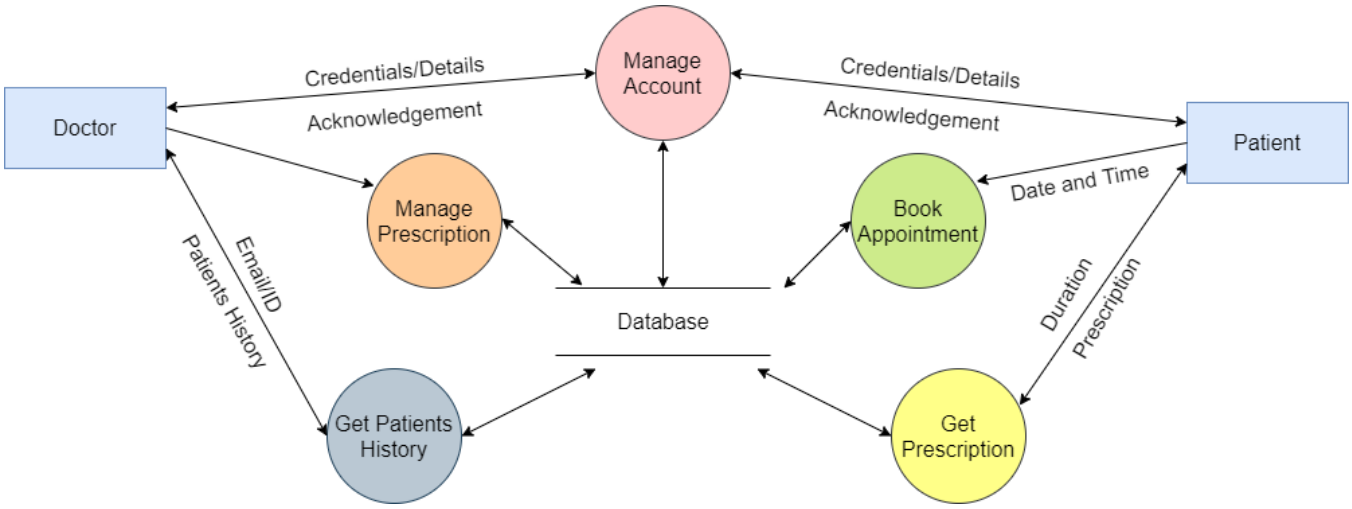
Communication between the different parts of the system is important since they depend on each other. However, in what way the communication is achieved is not important for the system and is therefore handled by the underlying operating systems for the web portal.

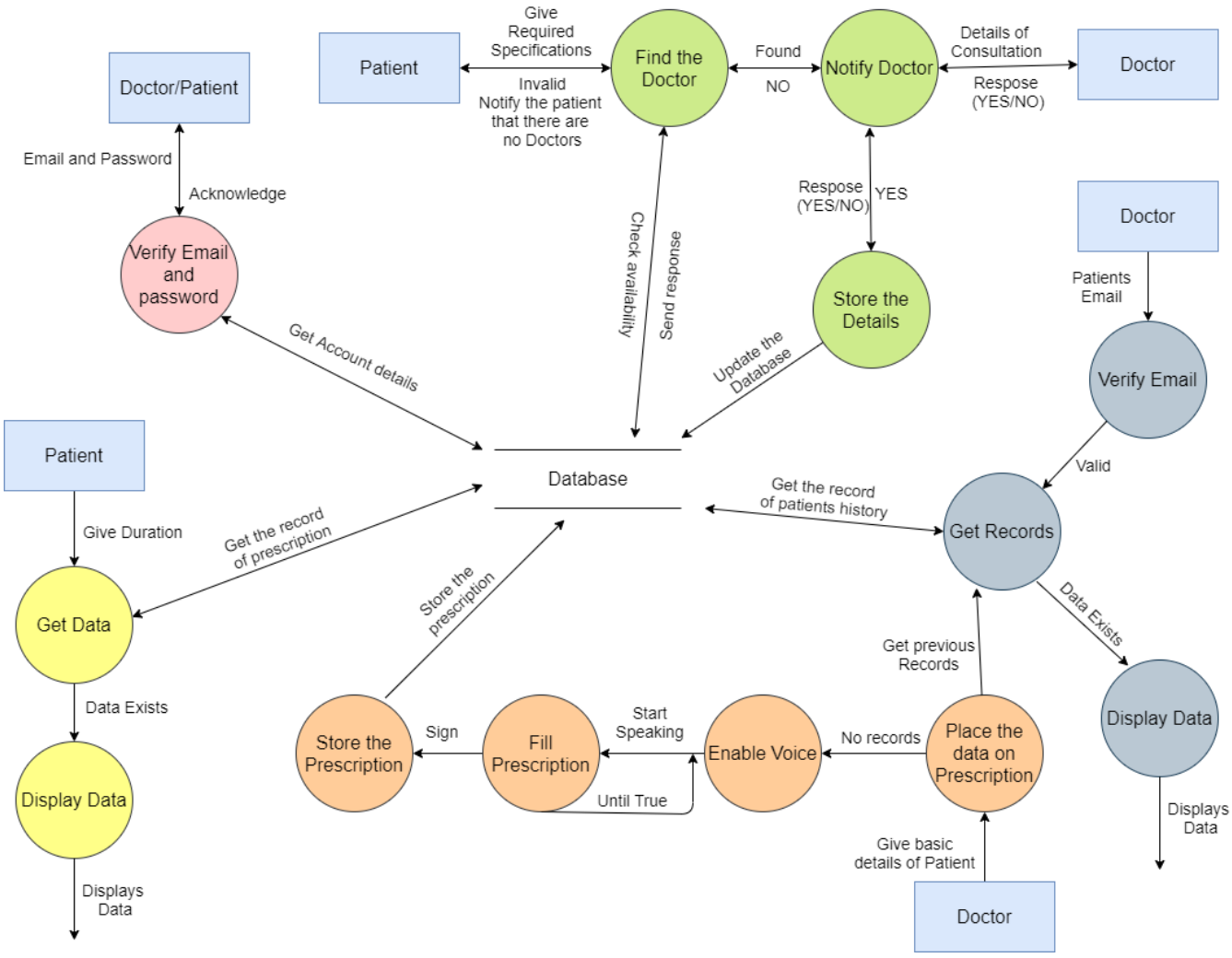
3.2. Functional Requirements

3.2.1. Information flows

Data flow diagrams for this application is as follows:







3.2.2. Process description

Function: Login

Input: Email, Password

Logic: get the data from the database matching the email and verify the password given as inputs with that of the records.

Output: True or False.

Function: Register

Input: Email, Password, Confirm Password, Phone, address.

Logic: Validate all the data and show alerts if any. And check if the given email already exists in the database or not. If not then insert new data records.

Output: True or False.

Function: Find Doctor

Input: Specifications and specialisations of doctor

Logic: Check the database if there are any records of doctors that are free and matching to our requirements, then call notify doctor function. If notification sent to the doctor is accepted the request is then stored as the details of the appointment in the database and reflect the changes to the patient.

Output: Doctor is assigned or Notified if there is no availability.

Function: Notify Doctor

Input: Patient Id, Timing of appointment, Doctor Id

Logic: Show the details of the request to the doctor of the given Id. If the doctor accepts the appointment then call the store details function. Else return false which means we'll find the doctor once again.

Output: Accept or Reject

Function: Store appointment details

Input: Patient Id, Timing of appointment, Doctor Id

Logic: Validate the information and store the data along with the flag value which determines whether the appointment is taken or not. It can be found out by the prescriptions list, where prescriptions will be added when they have appointments.

Output: Data will be stored

Function: Get Prescriptions of given Duration

Input: Duration

Logic: Get data from the database the list of prescriptions that are in given duration and sort them. If no data exists return null else send the data to the display function.

Output: Prescriptions form database

Function: Display Patient's History

Input: Data from 'Get Prescriptions of given Duration' function.

Logic: The data is displayed with proper formatting.

Output: Data is displayed.

Function: Init Prescription

Input: NULL

Logic: This function will call the enable voice function which enables the voice of our system and calls the fill prescription function.

Output: stores the prescription.

Function: Fill Prescription

Input: Voice input

Logic: The voice is converted to text and then by using regular expressions and default libraries the data will be extracted and placed in the prescription data structure or class. This continuously happens until the whole doc is filled.

Output: Returns the prescription.

Function: Store Prescription

Input: Prescription

Logic: The data is validated and stored in database

Output: True or False

Function: Verify Email

Input: Email

Logic: Checks if the email is correct and exists in the users data from the database

Output: True or False

Function: Get records of Patients

Input: Email

Logic: The latest 10 prescriptions of the patient will be taken from the database and calls display history function. In case there are no prescriptions, null is returned. And if there are less than 10 prescriptions all the number of prescriptions that exist will be taken and displayed.

Output: The data stored in Prescription data structure.

Function: display History

Input: Prescriptions

Logic: The data is displayed in a formatted way

Output: Data is displayed.

3.2.3. Data Dictionary

Name	Type	Description
email	string	The email is the unique identifier of user
password	string	Password is a secret key for login.
user_type	char	doctor/patient
Doctor ID	int	Unique identification for doctors
Patient ID	int	Unique identification for patients
Name	String	User name
Address	String	User's residential

		address
Specialization	String	Doctor’s field of specialization
Designation	String	Doctor’s designation at the hospital
Experience	int	Doctor’s experience in years
Licence number	int	Proof that the doctor is a real doctor
age	int	Patient’s age
diabetic	Boolean	1 if patient is diabatic else 0
hypertension	Boolean	1 if patient have hypertension else 0
Appointment Date	Date	The date of appointment
symptoms	Array of String	All the symptoms of patient
Medicine	Array of String	Doctor’s prescription for the diagnosis made
Dosage	Array of String	The amount of prescribed medicine to be taken by the patient

Description of the Data Types:

- User[Email, Password, type(doctor/patient)]
- Doctor[Doctor ID, Name, Specialization, designation, Experience, Licence number, Digital signature]
- Patient[Patient ID, age, Is Diabetic, Has Hypertension]
- Prescription[Patient ID, Doctor ID, Date, Symptoms, Medicines, Dosage]
- Appointment[Patient ID, Doctor ID, Date, Time, Approved]
- User is the superclass of Doctor and Patient.

3.3. Performance Requirements

The requirements in this section provide a detailed specification of the user interaction with the software and measurements placed on the system performance.

There should be a proper list view of all the details as the results displayed in the list view should be user friendly and easy to understand. Selecting an element in the result list should only take one click.

The response time of data flowing should be no more than 1 sec. System dependability should be 100% of the time.

Appointments will contain data such as patient Id and doctor ID along with the time and date of the appointment.

By considering the security issues all the data insertions and deletions are done only through the application. The application takes data from the users and by the server, the data will be stored in the database and vice versa.

3.4. Logical database requirements

We create Databases using MySQL, based on the Use Class requirements.

There are 2 types of users, they are Doctor and Patient. Both the users are generalised and have the attributes UserId, email, password and type. UserId is a unique entity that identifies the user. Type value determines whether the user is a doctor or a patient.

Doctors have extra attributes like his qualification details, his specialisation, digital signature and contact details. Whereas for a patient we have details such as age, his general medical details like diabatic, hypertension etc.

Prescriptions have various attributes like doctor id, patients id, symptoms, medicine name, dosage. Doctor id and patient id specify that this particular doctor has consulted the patient with that ID.

3.5. Design Constraints

This section includes the design constraints on the software caused by the hardware.

TITLE: Application Memory Usage

GIST: The amount of Operating System memory occupied by the application.

SCALE: MB.

METER: Observations done from the performance log during testing

MUST: No more than 1000 MB.

PLAN: No more than 500 MB

WISH: No more than 400 MB

Operating System: DEFINED: The mobile Operating System on which the application is running on.

MB: DEFINED: Megabyte.

3.6. Software System attributes

The requirements in this section specify the required reliability, availability, security and maintainability of the software system.

3.6.1. Reliability

TITLE: The reliability of the system.

SCALE: The reliability that the system gives the right result on login.

METER: Measurements obtained from 1000 searches during testing.

MUST: More than 98% of the searches.

PLAN: More than 99% of the searches.

WISH: 100% of the searches.

3.6.2. Availability

TITLE: The availability of the system.

GIST: The availability of the system when it is used.

SCALE: The average system availability (not considering network failing).

METER: Measurements obtained from 1000 hours of usage during testing.

MUST: More than 98% of the time.

PLAN: More than 99% of the time.

WISH: 100% of the time.

TITLE: Internet Connection

DESC: The application should be connected to the Internet.

RAT: In order for the application to communicate with the database.

DEP: None

3.6.3. Security

TITLE: Communication Security

GIST: Security of the communication between the system and server.

SCALE: The messages should be encrypted for log-in communications, so others cannot get email and password from those messages.

METER: Attempts to get email and password through obtained messages on 1000 log-in sessions during testing.

MUST: 100% of the Communication Messages in the communication of a log-in session should be encrypted.

Communication Messages: DEFINED: Every exchange of information between client and server.

TITLE: User login System.

GIST: Security of accounts.

SCALE: If a patient/Doctor tries to log in to the web portal with a non-existing account then the restaurant owner should not be logged in. The user should be notified about log-in failure.

METER: 1000 attempts to log-in with a non-existing user account during testing.

MUST: 100% of the time

TITLE: User account - create security.

GIST: The security of creating an account for users of the system.

SCALE: If a user wants to create an account and the desired email is occupied, the user should be asked to choose a different email.

METER: Measurements obtained on 1000 hours of usage during testing.

MUST: 100% of the time.

3.6.4. Maintainability

TITLE: Application extendability

DESC: The application should be easy to extend. The code should be written in a way that it favors implementation of new functions.

RAT: In order for future functions to be implemented easily to the application.

DEP: none

TITLE: Application testability

DESC: Test environments should be built for the application to allow testing of the application's different functions.

RAT: In order to test the application.

DEP: none

3.6.5. Portability

TITLE: Application portability

DESC: The application should be portable with any browser in either PC or System.

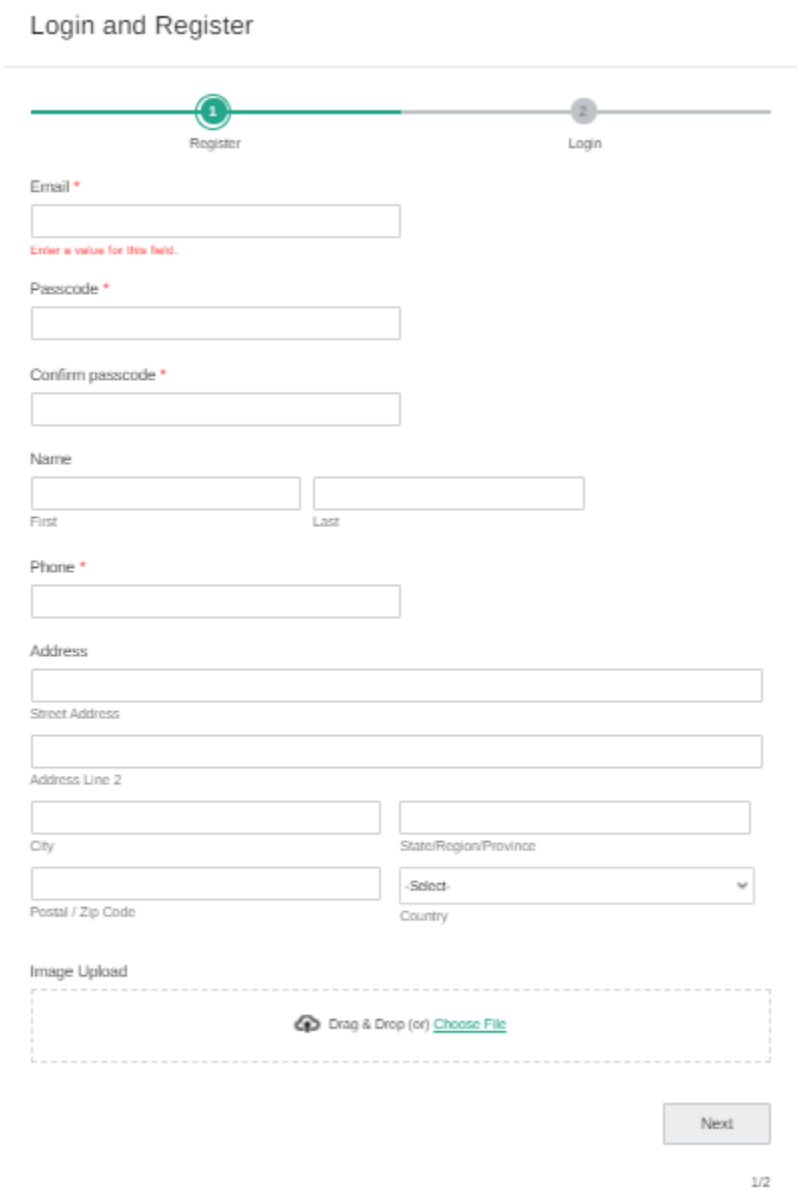
RAT: The adaptable platform for the application to run on.

DEP: None

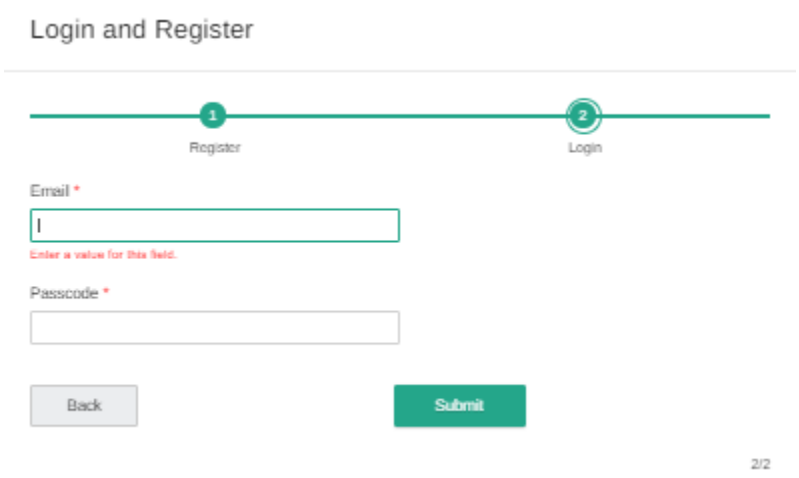
Appendix 1

User Interfaces:

Registration [Generalized]



Login [Generalized]



Prescription