Software Engineering
Software Requirements Specification (SRS) Document

RxSelect

September 26, 2023

Version 1

By: Kevin Ornelas Ceron, Romario Barahona, Jesse Carter

WE HAVE ABIDED BY THE UNCG ACADEMIC INTEGRITY POLICY ON THIS ASSIGNMENT

Table of Contents

1. Introduction

1.1. Purpose

The goal of the RxSelect application is to enable patients and healthcare workers to manage and keep track of healthcare appointments, prescriptions, and billing information. Patients will be able to see upcoming appointments set by admins. Admins will be allowed to send and update patients bills to the patient portal system.

1.2. Document Conventions

The purpose of this Software Requirements Document (SRD) is to outline the client-view and developer-view requirements for our Medical Center Management System (MCMS). Client-oriented requirements provide an overview of the system's functionality from the user's perspective, encompassing the diverse user roles within the healthcare ecosystem. Developer-oriented requirements, on the other hand, offer a detailed insight into the software's functionality, data management, performance specifications, and other critical aspects, seen through the lens of software developers and system architects.

1.3. Definitions, Acronyms, and Abbreviations

FDA	U.S. Food and Drug Administration, federal regulatory body that provides the drug information used in our system via their openFDA API.
Java	A programming language originally developed by James Gosling at Sun Microsystems. We will be using this language to build RxSelect.
MySQL	Open-source relational database management system that will be used to keep patient records.
.HTML	Hypertext Markup Language. This is the code that will be used to structure and design the web application and its content.
SpringBoot	An open-source Java-based framework used to create a micro Service. This will be used to create and run our application.
MVC	Model-View-Controller. This is the architectural pattern that will be used to implement our system.
Spring Web	Will be used to build our web application by using Spring MVC. This is one of the dependencies of our system.
Thymeleaf	A modern server-side Java template engine for our web environment. This is one of the dependencies of our system.
Visual Studio Code/IntelliJ	An integrated development environment (IDE) for Java. This is where our system will be created.
API	Application Programming Interface. This will be used to implement a search function for healthcare workers to browse medications.

1.4. Intended Audience

Stakeholders for this project include the users of the RxSelect System:

- Administrators, who manage billing information and appointments via the system.
- Healthcare Workers, who assign treatments to patients via the system.
- Patients, whose treatment history is recorded in the system.
- Owners of medical facilities that use RxSelect.
- Developers, who are in charge of developing and maintaining the system.

The following sections of the SRS document are intended for all stakeholders of the RxSelect System, to better understand its general purpose and use from a user perspective:

- Section 1 - Introduction
- Section 2 - General Description
- Section 3 - Functional Requirements
- Section 7 - Scenario

The following sections of the SRS document are intended primarily for the Developers of RxSelect, breaking down implementation details and expected behaviors:

- Section 4 - Technical Requirements
- Section 5 - Non-Functional Requirements
- Section 6 - Design Documents

1.5. Project Scope

The goal of RxSelect is to provide an easy-to-use interface for all patients, employees, and healthcare workers of a medical center, as well as provide patients with ease-of-use to meet their needs. This aligns with the overall goals of a healthcare professional as a medical center requires precise and efficient service in order to fulfill the needs of its patients.

The benefits of the project to medical centers include:

- **Enhancing Employee and Healthcare Worker Efficiency:** The software alleviates stress and work pressure on medical center employees and healthcare professionals by offering an intuitive platform that allows them to efficiently access and manage patient information and tasks as needed.
- **Improving Patient Satisfaction:** Patients benefit from increased control and convenience as the software empowers them to access services and information easily, reducing their reliance on staff intervention. This translates to higher patient satisfaction levels as they have a more active role in managing their healthcare experience.
- **Reducing Wait Times:** By streamlining processes and improving information accessibility, the software significantly reduces patient wait times. This, in turn, allows the medical center to serve more patients efficiently throughout the day, contributing to better overall service quality and patient throughput.

1.6. Technology Challenges

Security and Compliance: Ensuring the highest level of security for patient data is crucial. We will need to employ robust encryption, access controls, and/or authentication mechanisms to protect sensitive medical information. Compliance with healthcare data regulations like HIPAA (in the U.S.) or GDPR (in Europe) is essential.

1.7. References

openFDA API Documentation. (2023, August 14). U.S. Food and Drug Administration. https://open.fda.gov/apis/

2. General Description

2.1. Product Perspective

RxSelect found its origins in an aspiring developer's quest for a more efficient and patient-centric approach to healthcare management. The concept was conceived by a healthcare novice, with the primary goal of serving the healthcare community.

2.2. Product Features

RxSelect encompasses a range of essential features to serve both individual patients and healthcare facilities efficiently. Key functionalities include user registration for patients and healthcare institutions, granting administrators the ability to manage these accounts effectively. Patients can access features such as medical history tracking, appointment scheduling, and secure communication with healthcare providers, tailored to their specific medical needs. For healthcare facilities, the software empowers them to manage patient data, create and manage appointments. Administrators have additional capabilities such as user account oversight and management for optimal system control.

2.3. User Class and Characteristics

RxSelect is a website application which does not expect our users to have any prior knowledge of a computer, apart from using a web browser and basic data entry. Our web-based application has removed the necessity for users to have a background in healthcare, medical terminology, or technical knowledge, allowing them to focus on managing their healthcare needs effortlessly.

2.4. Operating Environment

RxSelect is designed to operate on the web across the many different devices that support a web browser.

2.5. Constraints

Given the complexity of our application features, we have certain limitations that necessitate careful consideration during design and implementation. These include browser compatibility constraints, as our app may require modern web browsers to fully support certain functionalities. Other limitations include designing the application without first hand medical practitioner experience.

2.6. Assumptions and Dependencies

RxSelect will be dependent on Spring Web and Thymeleaf in order to create and execute the MVC architecture that will be developed within Visual Studio Code/IntelliJ. The application will also use the openFDA API (https://open.fda.gov/) to allow healthcare workers to gather up-to-date information from the FDA's database in order to create comprehensive treatment plans for their patients.

3. Functional Requirements

3.1. Primary

- FR0: The system will allow patients to access their medical records to view current and previous medications, along with treatment details from their healthcare provider.
- FR1: The system will allow patients to request appointments with their healthcare provider, which can be approved by administrators.
- FR2: Patients, healthcare workers, and administrators will be able to view upcoming appointments.
- FR3: The system will allow patients to view current billing information, including amount owed, payment history, and cost breakdown by appointment.

- FR4: The system will allow healthcare workers to set prescriptions and treatment details for patients, update them easily, and mark treatment plans as completed/resolved.
- FR5: The system will allow healthcare workers to search the FDA's database for information regarding medications they are considering prescribing.
- FR5: The system will allow administrators to register and delete current patients from the system, as well as update their billing information.
- FR6: The system will allow administrators to confirm or deny appointment requests made by patients.

3.2. Secondary

- Password protection for information only accessible to patients, healthcare workers, and administrative details.
- Authorization scheme so that patients can only view their own medical records, administrators cannot view medical information, and healthcare workers cannot view administrative details.

4. Technical Requirements

4.1. Operating System and Compatibility

The application will be compatible with any operating system that is able to access the internet and view and interact with traditional web pages.

4.2. Interface Requirements

4.2.1. User Interfaces

The user interface should be easily navigable to a new user with limited technical expertise, using large buttons with easy to understand functions, a format that is easy to read, and including the following features:

Screen Layout:

- Dashboard
- Patient Profile
- Appointment Scheduler
- Medical Records
- See Prescription
- Billing and Payments

Buttons / Functions:

- Save/Submit
- Back/Cancel
- Menu Navigation or Sidebar

Messages:

- Success/Error
- Payment Confirmations
- Appointment Notifications

4.2.2. Hardware Interfaces

The web application will run on any hardware device that has access to the internet, the ability to display web pages via a web browser, and the ability to interact with web pages. This includes, but is not limited to, smartphones, tablets, desktop computers, and laptops.

4.2.3. Communications Interfaces

It must be able to connect to the internet as well as the local database on phpMyAdmin. The communication protocol, HTTP, must be able to connect to the openFDA API and return drug data based on symptoms.

4.2.4. Software Interfaces

We will use React and Spring Boot ThymeLeaf to help build the frontend, as well as MySQL for the backend database functionality. We will also use Spring Boot with Java to connect the frontend to the backend.

5. Non-Functional Requirements

5.1. Performance Requirements

- NFR0(R): The local cache of patient records for a doctor will consume less than 50 MB of memory.
- NFR1(R): The system (including the local caches for doctors and administrators) will consume less than 200 MB of memory.
- NFR2(R): A doctor should be able to access a patient's medical history and update their records in less than 10 seconds.
- NFR3(R): An administrator should be able to process a patient admission or discharge in less than 3 minutes.
- NFR4(R): A patient should be able to view their medical records, including appointment history and prescriptions, within 5 seconds of logging into the portal.
- NFR5(R): Patient data uploads(e.g., medical images or documents) should take less than 30 seconds to process and store.
- NFR6(R): The portal's search functionality should return results for patient queries within 5 seconds.
- NFR7(R): The system should generate and send appointment reminders to patients via email or SMS within 5 minutes of appointment creation.
- NFR8(R): Data retrieval and rendering for medical images (e.g., X-rays or MRIs) should occur in less than 5 seconds for doctors reviewing patient records.
- NFR9(R): The portal should be responsive and compatible with modern web browsers, with page load time of less than 3 seconds for all pages.

5.2. Safety Requirements

- NFR10(R): The healthcare portal must implement robust data encryption for all data at rest and in transit to protect patient confidentiality.
- NFR11(R): Access to patient data must be strictly controlled and follow the principle of least privilege, ensuring that only authorized users can view and modify patient records.
- NFR12(R): The system must regularly back up patient data to prevent data loss in the event of hardware failure or other disasters. Backup data must be stored securely and be easily recoverable.
- NFR13(R): The healthcare portal must implement an intrusion detection system (IDS) and intrusion prevention system (IPS) to detect and mitigate potential security breaches.
- NFR14(R): Emergency access procedures should be in place to allow authorized personnel to access patient records in critical situations while ensuring proper logging and auditing.

- NFR15(R): The system should have mechanisms to automatically log users out after a period of inactivity to prevent unauthorized access when a user leaves their session unattended.
- NFR16(R): The healthcare portal must adhere to industry standards and best practices for security, such as those outlined by HIPAA and GDPR.

5.3. Security Requirements

- NFR17(R): Users must authenticate using strong methods, such as multi-factor authentication (MFA), to access the portal.
- NFR18(R): Role-based access control (RBAC) must be enforced to ensure that users only have access to the information and functions appropriate to their roles.
- NFR19(R): All data transmitted between users and the portal must be encrypted using strong encryption protocols (HTTPS).
- NFR20(R): Sensitive patient data, including medical records and personal information, must be encrypted when stored in the database.
- NFR21(R): Ensure strict compliance with healthcare regulations, including HIPAA, GDPR, and other applicable laws and standards.

5.4. Software Quality Attributes

[Detailing on the additional qualities that need to be incorporated within the software like maintainability, adaptability, flexibility, usability, reliability, portability etc.]

5.4.1. Availability

Due to the important medical use of the software, it should have very high availability with almost constant uptime, and very short downtime should occur from maintenance or updates. The search functionality availability will be determined by the availability of the openFDA API.

5.4.2. Correctness

The software should behave as expected in all scenarios. API downtime or unstable connections should not result in unexpected behavior, and should be handled appropriately by letting the user know of the issue and allowing them to continue their work within the program.

5.4.3. Maintainability

The software should be built in such a way that it is easy to add, modify, or remove features in the future based on the needs of medical staff, healthcare workers, and patients. It should also be easily adaptable to allow for quick updates to conform to any changes in the use of the openFDA API, or allow for a different data source to be used in place of it if desired.

5.4.4. Ease of Use

Healthcare workers, patients, and administrators should be able to be trained to use the software quickly and easily, with minimal previous technical experience. Patients of all backgrounds should be able to understand how to navigate the software to acquire their treatment details.

5.4.5. Portability

The software should be highly portable, allowing patients with a range of devices to be able to access their medical records from their PC, phone, tablet, or any other device with internet access.

5.4.6. Security

Due to the sensitive nature of medical records, the system should employ robust authorization techniques in order to ensure the safety of confidential patient data.

5.5. Process Requirements

5.5.1. Development Process Used

In our project, we have adopted an Agile software development process model to build RxSelect. Agile is best suited to our dynamic and rapidly evolving healthcare environment. It enables us to develop and refine the application, accommodating changing requirements and ensuring close collaboration with stakeholders.

5.5.2. Time Constraints

The project is on a set timeline, with a UI Prototype to be developed by Oct 3, design documentation to be completed by Nov 2, and all functionality must be implemented by Dec 5.

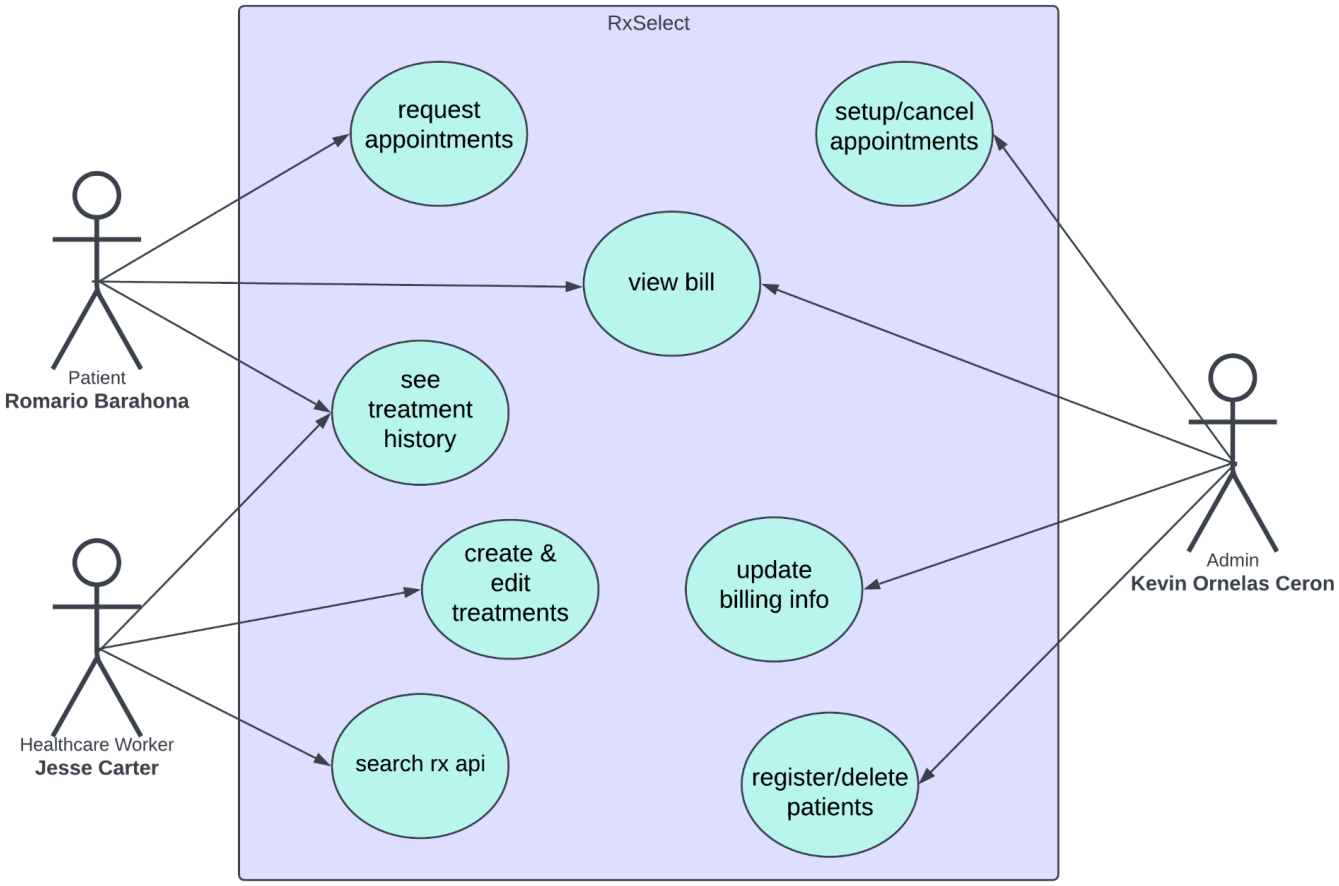
5.5.3. Cost and Delivery Date

The system is to be presented on December 5th, 2023, and must be in a deliverable state with all functionality implemented by this date.

5.6. Other Requirements

TBD

5.7. Use-Case Model Diagram



5.8. Use-Case Model Descriptions

5.8.1. Actor: Patient/User (Romario Barahona)

- **Request Appointments**: Patient can select preferred date and times for appointment, specify the reason for the appointment (e.g., routine check-up or specific health concern). Patients should receive confirmation and appointment details through the portal. Users may also receive notifications or reminders about upcoming appointments.
- **View Bills**: Users can access a summary of their healthcare bills, associated costs, insurance coverage, and payment history. The patient can download the bill statement for each medical visit. The portal can provide options for making payments or setting up payment plans.
- **See Prescriptions**: Users can review a list of their prescribed medications, including medication name, dosage, frequency, and the prescribing healthcare provider's name. Additionally they can access details about each medication, such as usage instructions, potential side effects, and refill information. The portal may also provide options for requesting prescription refills, scheduling medication reminders, or sending messages to healthcare providers for medication-related questions or concerns.

5.8.2. Actor: Admin (Kevin Ornelas Ceron)

- **Add / Modify / Delete Patient Data:** Admin Worker / Receptionist should be able to input patient data such as name, DOB, and other personally identifiable information (excluding actual health data).
- **Set Appointments**: Admin workers should be able to set, modify, and delete appointments for each patient.
- **Patient Billing:** Admin workers should be able to add/change/view billing amounts for patients based on service performed by healthcare workers and amounts paid by the patients if they have any outstanding balances. Billing portal should also have the option to take online payments using a credit card for ease of use.

5.8.3. Actor: Healthcare Worker (Jesse Carter)

- **See Prescriptions**: The healthcare worker should be able to view current prescriptions and treatment notes for any patient. This can include information about any medications the patient is taking, details such as dosage instructions, and the patient's history of previously prescribed medications.
- **Set Prescriptions**: The healthcare worker should be able to assign new prescriptions to the patient, either by selecting them from the result of a search or by adding them manually. They can also give specific dosage instructions along with any other personalized notes for treatment.
- **Search Medications**: The healthcare worker should be able to query the system for medications based on patient symptoms. By inputting a symptom, the healthcare worker should receive a list of medications that can be used to treat that symptom, along with other important information about the drug, such as possible side-effects and risks.

5.9. Use-Case Model Scenarios

5.9.1. Actor: Patient/User (Romario Barahona)

- **Request Appointment**:
    - **Initial Assumption**: Patient has successfully logged into the healthcare portal with the appropriate permissions.

- **Normal**: The patient logs into the healthcare portal. They can navigate to the "Appointment" section of the portal and then select the "Request Appointment" option. They can choose a preferred date and time for the appointment that is available. The patient can provide a reason for the appointment, which could be "following up on a recent medication. They then can select his primary care physician. The patient reviews the appointment details and confirms the request. The system will then display a confirmation message, stating the appointment has been submitted successfully. The patient will then receive an email notification with the appointment details and may opt-in for texts reminding them of their future appointments.
- **What Can Go Wrong:** The user could encounter technical issues, such as a system error during the submission and may need to retry the request. The patient's preferred healthcare provider could be booked for that date and time and the system will suggest an alternative time and date. If the provider is fully booked it will ask the user to choose another healthcare provider.
- **Other Activities**: The user can view and manage his upcoming appointments through the portal. The admin will review and confirm the appointment request. If approved, the user will receive a confirmation message.
- **System State on Completion**: The system records a user's appointment request and receives a confirmation with appointment details. The appointment request is pending confirmation from the admins.

- **View Bills**:
  - **Initial Assumption**: The patient has logged into the healthcare portal with the correct permissions. The patient also has received medical services, and the bills associated with the medical services are available in the system.
  - **Normal**: The patient logs into the healthcare portal. They navigate to the "Billing" section of the portal. Then they select the "View Bills" option. The system then displays a summary of his recent healthcare bills, showing information such as the total amount due, due dates, and a list of bills. The user clicks on a specific bill from the list to view detailed information. In the detailed bill, the patient can see the itemized charges for the service, date of service, insurance coverage, and any payments made. If needed, the patient can download the detailed bill as a PDF for his record.
  - **What Can Go Wrong**: Technical issues may prevent the user from accessing the bill details. They might receive an error message and should retry accessing the information later. The billing information may not be up-to-date or may contain errors. In this case the user should contact the admin/billing department through the portal for corrections.
  - **Other Activities**: The user can perform a variety of financial activities related to the bills, such as making payments online, setting up a payment plan, or inquiring about billing-related issues through the portal.
  - **System State on Completion**:
- **See Prescription**:
  - **Initial Assumption**: Patient has logged in to the portal and has active prescriptions from their healthcare provider.
  - **Normal**: Patient is logged in and selects the "See Prescription" option and will take him to a web page that lists his prescription. Patients can select a medication to view

more details about the prescription such as, usage instructions, potential side effects, and refill information.

- **What Can Go Wrong**: Issues can arise during the process of seeing a prescription. The user can experience network issues or the website could be experiencing a downtime where the prescription can't be accessed. If this is the issue the user will have to try again later.
- **Other Activities**: The Patient can request prescription refills, set medication reminders, look at past prescriptions, or send messages to healthcare providers for questions about the medication.
- **System State on Completion**: The user can access and review information about the prescribed medications, including usage, instructions, and refill details. The prescription information is accurate and up-to-date in the system.

5.9.2. Actor: Admin (Kevin Ornelas Ceron)

- **Use-Case Name**: Manage Patient Appointments
  - **Initial Assumption:** The admin worker has successfully logged into the system with the appropriate permissions. It also assumes that patient appointment data is available and up to date in the system.
  - **Normal:** The admin accesses the appointment management interface, where they can view a list of scheduled appointments. They have the ability to create new appointments for patients, selecting the date, time, and healthcare provider. Additionally, the admin can modify existing appointments, either rescheduling or canceling them as needed. The system allows the admin to search for specific appointments and filter them based on criteria like date, patient name, or healthcare provider. Furthermore, the admin can generate reports or summaries of appointments for analysis or record-keeping.
  - **What Can Go Wrong:** Several potential issues may arise during this process. The admin may encounter network problems, hindering their access to appointment data. Incorrect data entry could lead to scheduling conflicts or inaccurate appointment information. The admin may face difficulties while searching for specific appointments if the search functionality is not intuitive or encounters technical issues. System downtime or technical problems might also temporarily disrupt appointment management.
  - **Other Activities:** Apart from appointment management, the admin may need to engage in other activities. This could include communicating with patients or healthcare providers regarding appointment changes or updates. Additionally, the admin might need to coordinate with the billing department in cases where appointments impact patient billing or insurance claims.
  - **System State on Completion:** Upon successful completion of this use case, the system will accurately reflect any changes made to the appointment schedule. The admin will retain the ability to view, create, modify, and search for appointments as needed. Patient and healthcare provider schedules will be kept up to date and accurate, ensuring efficient appointment management within the healthcare setting.
- **Use-Case Name**: Patient Billing
  - **Initial Assumption:** Assumes the Admin is logged in with the necessary permissions, and patient billing and service data is up-to-date.

- **Normal:** Admin accesses the billing interface, records/updates billing details, calculates billing amounts, generates detailed billing records, handles payments, and collaborates with patients/insurance for inquiries and disputes.
- **What Can Go Wrong:** Network issues, data entry errors, technical glitches, payment processing issues, patient/insurance inquiries or disputes.
- **Other Activities:** Admin may communicate with patients and insurance providers for billing clarification, dispute resolution, and insurance claims. Collaboration with the finance department for reporting and compliance**.**
- **System State on Completion:** System accurately reflects billing records, payment statuses, and outstanding balances. Billing inquiries and disputes are addressed efficiently.

- **Use Case Name:** Add/Manage Patient Information
  - **Initial Assumption:** The scenario assumes that the Admin has successfully logged into the system with the necessary permissions, and the admin dashboard is readily accessible for patient management.
  - **Normal:** In the usual course of action, the Admin navigates through the system effortlessly. They can add new patients by entering their information, edit existing patient records as needed, efficiently search for specific patients, and delete patient records when required. The system is designed for data accuracy and security, ensuring a smooth and streamlined process for managing patient information.
  - **What Can Go Wrong:** Potential issues that may occur include network interruptions, data entry errors, or technical glitches that could temporarily disrupt the patient management process. Additionally, accidental deletions or unintended edits may require corrective actions.
  - **Other Activities:** Apart from adding and managing patient information, the Admin may also perform other administrative tasks within the system. These activities could include managing appointments, overseeing billing, or coordinating with healthcare providers and patients for various administrative needs.
  - **System State on Completion:** Upon successful completion of the scenario, the system maintains accurate and up-to-date patient records. Patient information is securely stored and easily accessible. Any changes or updates made by the Admin are reflected accurately, ensuring efficient patient management within the healthcare system.

5.9.3. Actor: Healthcare Worker (Jesse Carter)

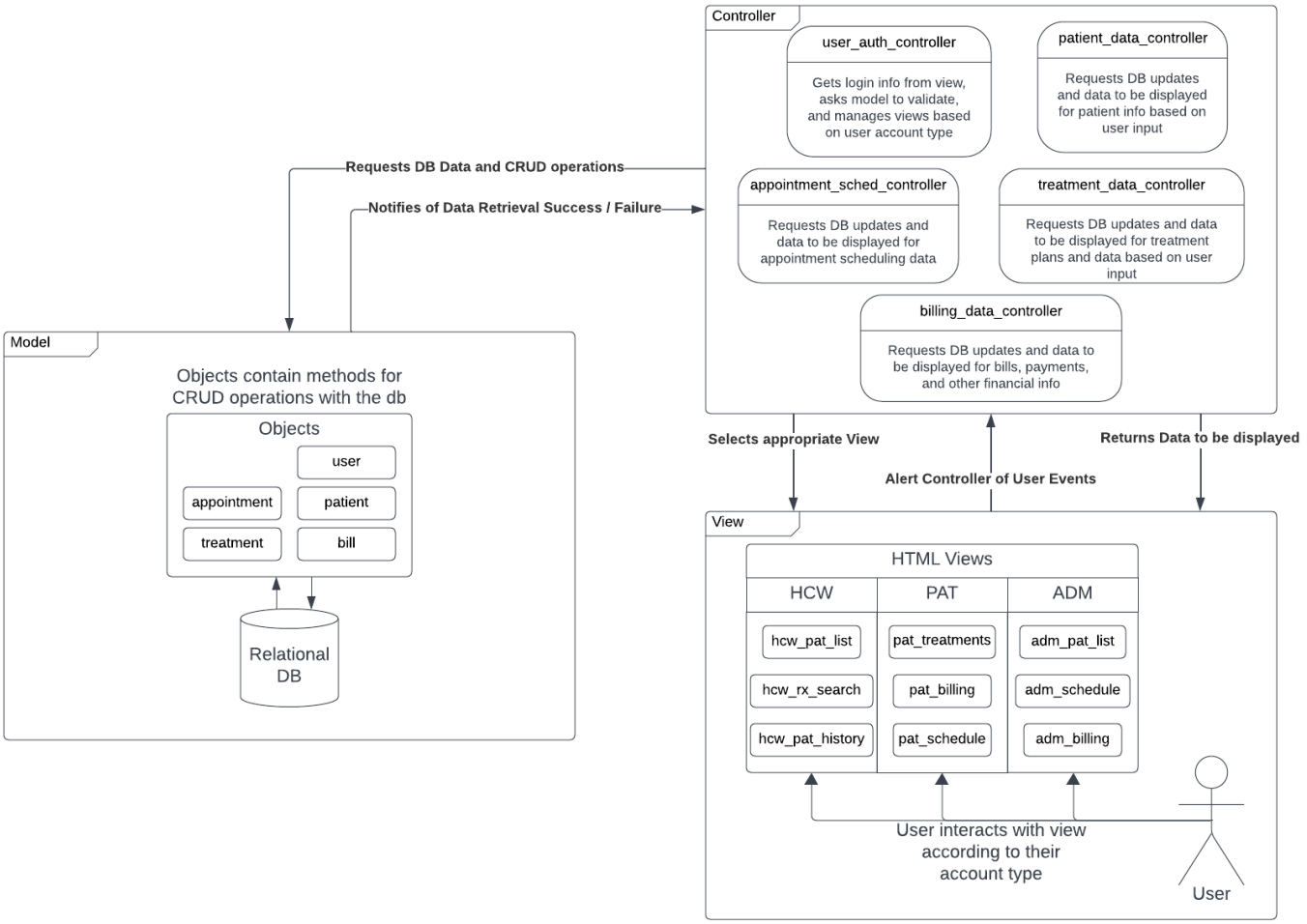
- **See Treatments**:
  - **Initial Assumption**: The user is logged into the system as a healthcare worker.
  - **Normal**: The user will navigate to “Patient List,” (this is the default starting page after logging in as a healthcare worker) and be shown a list of all patients. They can filter this list by patient information such as name or date of birth to find the desired patient. Once the desired patient is selected, the system should display the patient’s current set of prescriptions and treatment notes (if any), as well as a list of the patient’s previous prescription entries sorted by date prescribed. A prescription entry will include a brief treatment title, a list of medications prescribed (if any), and a section for detailed treatment notes. When finished, the user can click “discard changes and return to patient list” or use the navbar to navigate to another page.

    - **What Can Go Wrong**: If the search criteria do not match any patients in the database, the system should present an empty list. If the patient is a new patient with no current prescription and no prescription history, the patient information screen should show an empty treatment history.
    - **Other Activities**: The user can click on any previous prescription entry to see a more detailed view, including the dosage and treatment notes, for that prescription. Note: A prescription entry in this system does not necessarily have to include a medication, and could include only a set of treatment notes.
    - **System State on Completion**: The system will return to the list of patients. From here, the user can either search for another patient, or use the navbar to navigate to another page.

- **Set Treatments**:
    - **Initial Assumption**: The user is logged into the system as a healthcare worker and has selected a patient from the system's "Patient List" screen.
    - **Normal**: Once the desired patient is selected, the system should display the patient's current and previous prescriptions and treatment details. The user can click "Add New Rx" to add a new prescription entry. The user will then be able to enter a medication name or brief treatment title, as well as treatment details. Once finished, the user can click "Save Changes and Update Rx" to add the new prescription to the patient's treatment history, and can use the navbar to navigate away from the patient information screen.
    - **What Can Go Wrong**: If the patient is a new patient with no current prescription and no prescription history, the patient information screen should show an empty treatment history. If the database is not accessible for some reason and changes cannot be saved, the system should show a warning to indicate this to the user.
    - **Other Activities**: The user can mark previous prescription entries as "completed" to indicate that the treatment is not ongoing. The user can also delete prescriptions in the list, in case an error was made in their creation.
    - **System State on Completion**: The system will return to the list of patients. From here, the user can either search for another patient, or use the navbar to navigate to another page.
- **Search Medications**:
    - **Initial Assumption**: The user is logged into the system as a healthcare worker and has navigated to "Rx Search" on the system's navbar.
    - **Normal**: The user will enter the symptom they are trying to treat or the name of the medication they are interested in. The system will display a list containing up to 3 results, displaying the medication name and basic information. This includes Usage information, Dosage and Administration details, and information regarding possible Adverse Reactions.
    - **What Can Go Wrong**: If the search function yields no results or the API is unavailable, the system should indicate this issue.
    - **Other Activities**: The user can choose to view more search results, which will expand the list to include more entries.
    - **System State on Completion**: The user stays on the Rx Search page, where the user can continue to search and browse other Rx information, or they can use the navbar to navigate to another page.

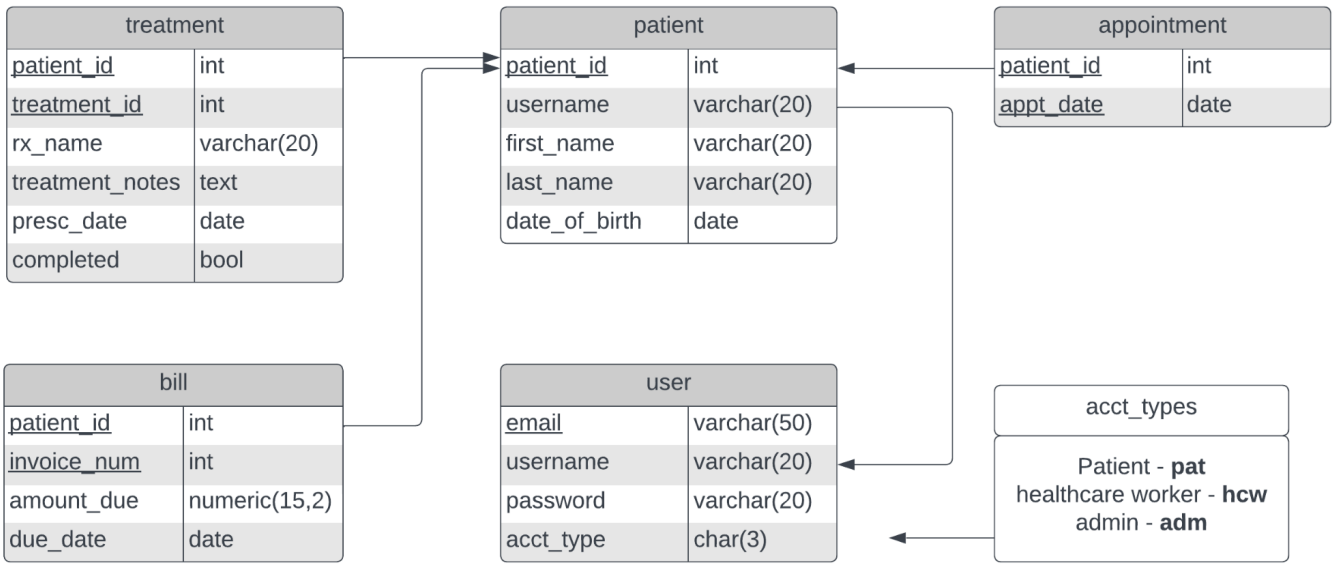
6. Design Documents

6.1. Software Architecture



Our software will use the Model-View-Controller (MVC) Architecture. The model will consist of our object classes, and will handle CRUD operations with the relational database that stores our persistent data. Our controllers will handle the task of fetching data from the model and feeding it to the view that the user will interact with. The type of view will be determined by the user's account type, which will be verified by the user_auth_controller.

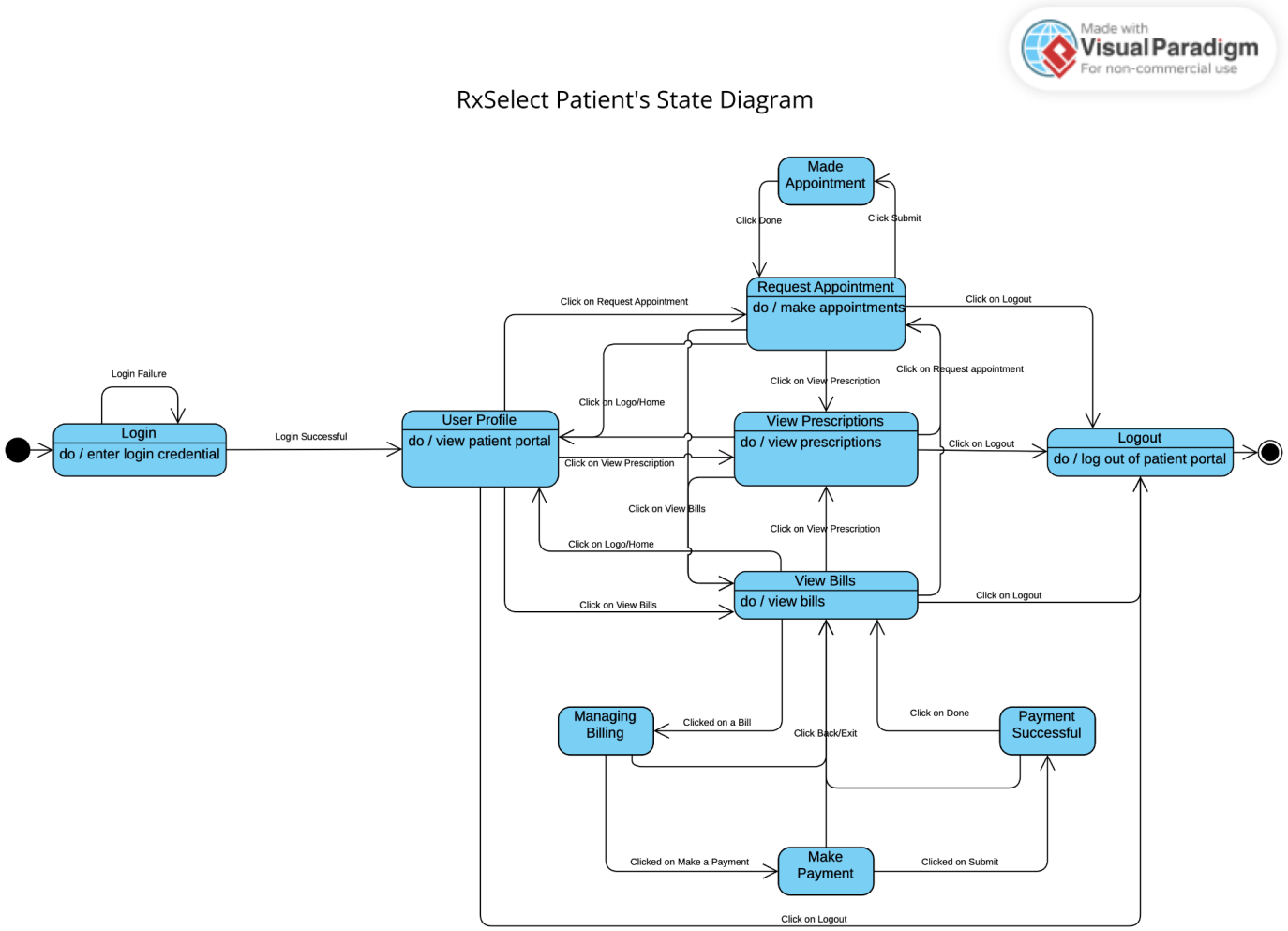
6.2. High-Level Database Schema



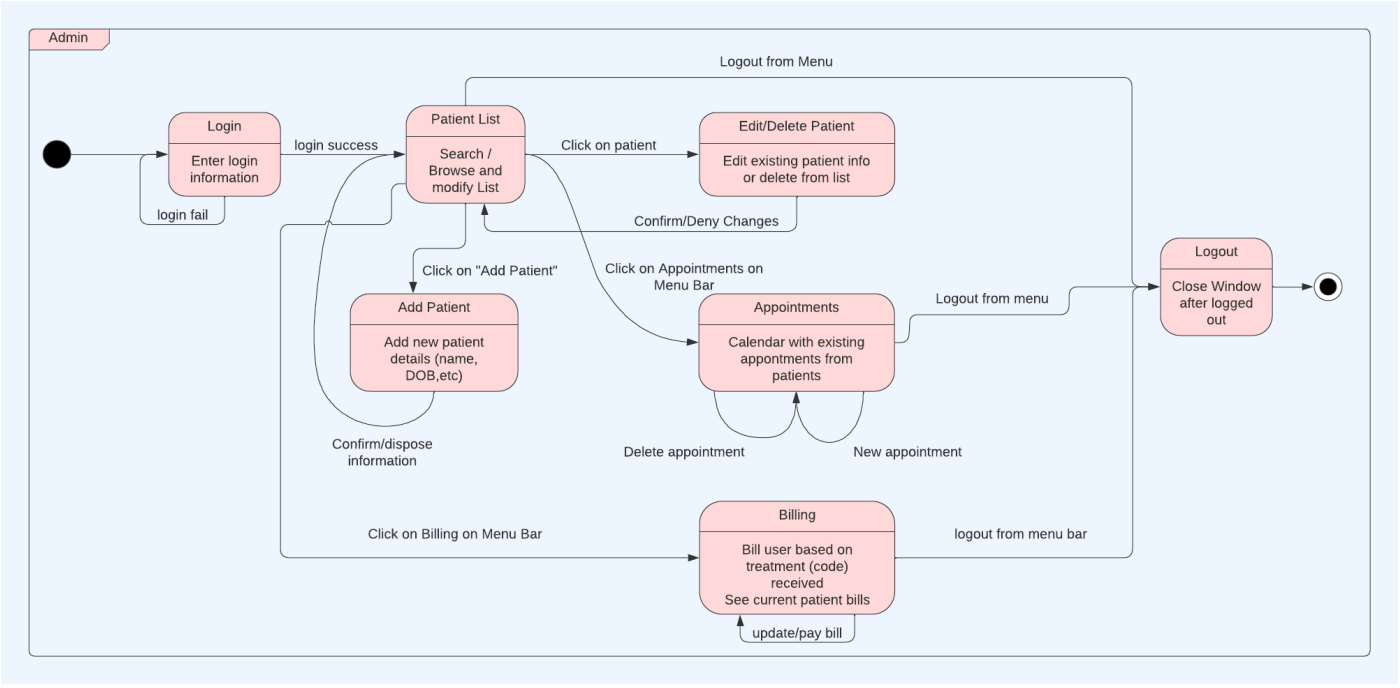
Our data will be stored in a relational database with the schema shown above. We will store data about **users**, including their login information, in the user table. The user table will also store information about what type of the account the user has - patient, healthcare worker, or administrator. The **patient** table is a specialization of a user, which also stores the patient's patient ID, first and last name, and date of birth. The patient ID is used to identify which **treatments**, **bills**, and **appointments** are associated with that patient. Email is used for the user to log in, and username is used to determine what name to display in the navbar for the currently logged in user. The acct_type will be used to determine what view is displayed to the current user.

6.3. Software Design

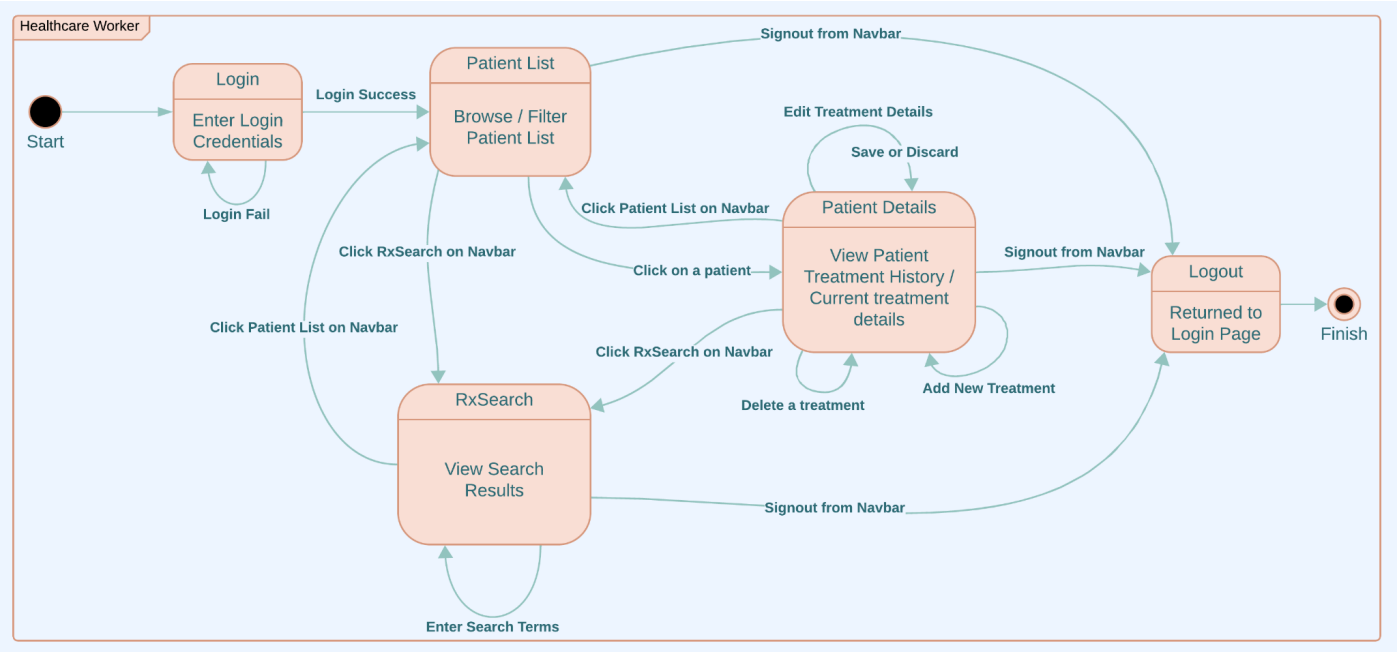
6.3.1. State Machine Diagram: User/Patient (Romario Barahona)



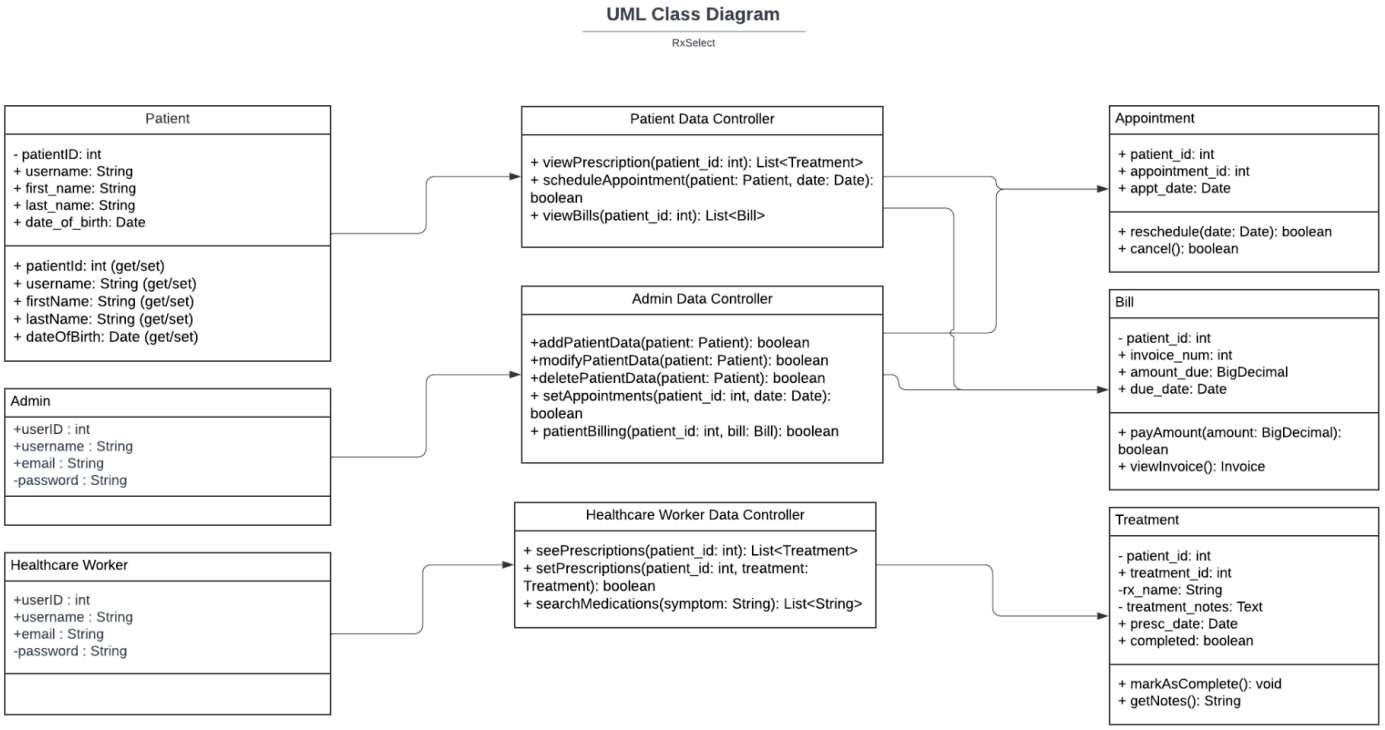
6.3.2. State Machine Diagram: Admin (Kevin Ornelas Ceron)



6.3.3. State Machine Diagram: Healthcare Worker (Jesse Carter)



6.4. UML Class Diagram



7. Scenario

7.1. Brief Written Scenario with Screenshots