Software Engineering
Software Requirements Specification (SRS) Document

MedCheck

9/12/2023

0.0.1

By: Dev Patel, Noah Phillips, Andy Salinas

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

Table of Contents

1. Introduction

1.1. Purpose

[The goal of your project and the objectives it wishes to accomplish]

This management system app will be an easy way for pharmacies to manage distribution of medications. If a medication is prescribed that has a conflict with another, our app will automatically notify the doctor and pharmacist and give options for a better suited medication. Our app also allows for doctors to prescribe medications and pharmacists to fill those medications.

1.2. Document Conventions

[Full description of the main objectives of this document in the context of your project.
Here's how you should begin this section:
"The purpose of this Software Requirements Document (SRD) is to..."
"In it, we will . . ., . . ., and . . .."]

The purpose of this Software Requirements Document (SRD) is to outline the key objectives and specifications of the MedCheck Medication Management System. In this section, we will provide a comprehensive overview of the system's client-view and developer-view requirements, highlighting its functionalities, security measures, and data access restrictions.

The MedCheck Medication Management System is a comprehensive solution designed to streamline medication distribution while prioritizing patient safety, user-friendly experience, and data security. This SRD provides a clear roadmap for the development and implementation of the system, addressing both client and developer requirements.

1.3. Definitions, Acronyms, and Abbreviations

[Include any specialized terminology dictated by the application area or the product area.
For example:]

Java	A programming language originally developed by James Gosling at Sun Microsystems. We will be using this language to build the Restaurant Manager.
MySQL	Open-source relational database management system.
.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.
NetBeans	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 function within the software where the current date and time is displayed on the homepage.
Indication	A valid reason to use a certain test, medication, procedure, or surgery.
Contraindication	A reason to withhold a certain medical treatment because the risks of treatment clearly outweigh the benefits.

1.4. Intended Audience

Stakeholders for the MedCheck Medication Management System include individuals or groups with a vested interest in the system's development, implementation, or outcomes. Here is a list of key stakeholders:

MedCheck Company Leadership: This includes company executives, founders, and decision-makers who are responsible for the overall strategic direction and success of the Medication Management System.

Pharmacists: Pharmacists are primary users of the system. They rely on it for medication dispensing, managing inventory, and ensuring patient safety. Their input is critical for system design and usability.

Doctors and Healthcare Providers: Physicians and healthcare providers use the system to prescribe medications. They play a crucial role in the system's functionality and ensuring patient care.

Patients: Patients benefit from the system's features, such as medication safety checks and access to their own information. Their satisfaction and trust in the system are essential.

Investors and Shareholders: If MedCheck is a publicly traded company or has investors, they have a financial stake in the success of the Medication Management System.

1.5. Project Scope

Aligning the software goals with the overarching business objectives of MedCheck's Medication Management System is pivotal for the success of the project and the company as a whole. This alignment serves as the bridge between technology and strategic vision, allowing the software to contribute significantly to the realization of business goals.

One of the primary software goals is to enhance medication management within pharmacies. By achieving this goal, MedCheck directly supports its core business objective of providing efficient and reliable pharmacy services. Streamlining the medication management process reduces errors, ensures prompt service, and enhances customer satisfaction. As a result, MedCheck can establish itself as a preferred choice for pharmacies, ultimately driving revenue growth and market share expansion.

Another critical software goal is to resolve medication conflicts effectively. This aligns seamlessly with MedCheck's overarching business goal of prioritizing patient safety and complying with healthcare regulations. Resolving medication conflicts through the system minimizes the risk of adverse events, reduces liability, and ensures compliance with stringent regulatory requirements. Consequently, the project bolsters the company's reputation and reinforces its commitment to patient well-being.

1.6. Technology Challenges

The MedCheck Medication Management System project may encounter various technological constraints and require the adoption of new technologies to effectively meet its goals. Here are some potential technological constraints and the new technologies that may need to be incorporated:

Constraint: The healthcare industry imposes strict regulations (e.g., HIPAA) regarding data security and patient privacy. Compliance is essential but can be challenging.
New Technology: Advanced encryption methods, secure authentication protocols, and robust access control mechanisms will be needed to protect sensitive data.

1.7. References

[Mention books, articles, web sites, worksheets, people who are sources of information about the application domain, etc. Use proper and complete reference notation. Give links to documents as appropriate. You should use the APA Documentation model (Alred, 2003, p. 144).]

OpenFDA. (n.d.). Home - OpenFDA. Retrieved from https://open.fda.gov

2. General Description

2.1. Product Perspective

[Describe the context and origin of the product.]

MedCheck Medication Management System emerged from a need to address critical issues in the healthcare and pharmacy management domain, including patient safety, regulatory compliance, and operational efficiency. Its origin is rooted in the desire to improve medication management processes and ultimately enhance the quality of care provided to patients.

2.2. Product Features

[A high-level summary of the functions the software would perform and the features to be included.]

At its core, MedCheck offers a robust Medication Management function, allowing pharmacies to seamlessly handle medication distribution, monitor inventory levels, and process prescriptions efficiently. This includes prescription management, medication dispensing, and real-time inventory updates, ensuring that pharmacies can operate smoothly and meet the needs of patients and healthcare providers effectively.

Ensuring patient safety is a top priority, and MedCheck addresses this through its Medication Conflict Resolution capabilities. The software employs advanced algorithms to identify potential medication conflicts promptly. When conflicts arise, healthcare providers and pharmacists receive instant notifications, allowing for rapid resolution and alternative medication recommendations. This feature not only enhances patient safety but also reduces liability and enhances the company's reputation.

To safeguard sensitive data and maintain compliance with healthcare regulations, MedCheck implements a sophisticated User Roles and Access Control system. User authentication, role-based permissions, and stringent data access restrictions ensure that only authorized personnel can access specific functionalities and patient information, reinforcing data privacy and compliance.

MedCheck is committed to providing a User-Friendly Experience for all system users. With a responsive design, intuitive navigation, and user-specific dashboards, the software aims to simplify interactions, reducing training costs and promoting widespread user adoption.

2.3. User Class and Characteristics

[A categorization and profiling of the users the software is intended for and their classification into different user classes]

Our website application does not expect our users to have any prior knowledge of a computer, apart from using a web browser.

2.4. Operating Environment

[Specification of the environment the software is being designed to operate in.]

The application is designed to operate on the web on a PC.

2.5. Constraints

[Any limiting factors that would pose a challenge to the development of the software. These include both design as well as implementation constraints.]

- The healthcare industry, including pharmacy management, is heavily regulated. Compliance with regulations like HIPAA (Health Insurance Portability and Accountability Act) and FDA (Food and Drug Administration) requirements is mandatory. Designing and implementing the system while adhering to these regulations can be complex and time-consuming.
- Protecting patient data is paramount. The system must ensure the security and privacy of sensitive medical information. Implementing robust encryption, access controls, and auditing mechanisms can be challenging.

2.6. Assumptions and Dependencies

[A list of all assumptions that you have made regarding the software product and the environment along with any external dependencies which may affect the project]

The application will use the OpenFDA API (https://open.fda.gov/) that will display the medication information.

3. Functional Requirements

[Statements of services the system should provide, how the system should react to particular inputs and how the system should behave in particular situations.]

3.1. Primary

[All the requirements within the system or subsystem in order to determine the output that the software is expected to give in relation to the given input. These consist of the design requirements, graphics requirements, operating system requirements and constraints if any.]

- FR0: Detailed specifications of the system's functionalities, such as prescription management, conflict resolution algorithms, and user authentication.electronic and paper form.
- FR1: Thorough description of data handling, including data storage, retrieval, and security measures to protect sensitive information.
- FR2: Performance requirements including system response times, scalability, and data processing speed to ensure efficient operation.
- FR3: Strict security measures to maintain data confidentiality and access control. User roles are defined to limit access to specific functions and data.
- FR4: Secure user authentication mechanisms to ensure that only authorized users can access the system.

3.2. Secondary

[Some functions that are used to support the primary requirements.]

- Password protection for information only accessible to employees, managers, and each individual.
- Authorization scheme so that customers can only alter and see their prescriptions and no other customers' prescriptions.

4. Technical Requirements

4.1. Operating System and Compatibility

[The environments that will be needed to operate the system]

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

4.2. Interface Requirements

4.2.1. User Interfaces

[The logic behind the interactions between the users and the software. This includes the sample screen layout, buttons and functions that would appear on every screen, messages to be displayed on each screen and the style guides to be used.]

The web application will have a function to login and signup for our service.

4.2.2. Hardware Interfaces

[All the hardware-software interactions with the list of supported devices on which the software is intended to run on, the network requirements along with the list of communication protocols to be used.]

The web application will run on any hardware device that has access to the internet, the ability to display webpages, 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

[Determination of all the communication standards to be utilized by the software as a part of the project]

It must be able to connect to the internet. The communication protocol, HTTP, must be able to connect to the OpenFDA API and return the product information.

4.2.4. Software Interfaces

[The interaction of the software to be developed with other software components such as frontend and the backend framework to be used, the database management system and libraries describing the need and the purpose behind each of them.]

We will use Spring Boot with Java to connect the frontend to the backend.

5. Non-Functional Requirements

[Constraints on the services or functions offered by the system (e.g., timing constraints, constraints on the development process, standards, etc.). Often apply to the system as a whole rather than individual features or services.]

5.1. Performance Requirements

[The performance requirements need to be specified for all the functional requirements.]

- NFR0 (R): The system shall maintain a medication database that occupies less than 500 MB of storage space.
- NFR1 (R): The system shall ensure that 95% of medication conflict alerts are generated and delivered to doctors and pharmacists within 3 seconds of identification.
- NFR2 (R): The system shall ensure that patient medication history is retrieved from the database and displayed to doctors and pharmacists within 2 seconds for any given patient.
- NFR3 (R): The system shall be capable of handling a minimum of 1,000 medication prescriptions per hour during periods of high demand.

5.2. Safety Requirements

[List out any safeguards that need to be incorporated as a measure against any possible harm the use of the software application may cause.]

- User Authentication and Authorization:
  Implement robust user authentication to ensure that only authorized personnel, such as healthcare providers and pharmacists, can access the system.
  Use role-based access control to restrict user access to specific functions and data, preventing unauthorized actions.
- HIPAA Compliance:
  Adhere to the Health Insurance Portability and Accountability Act (HIPAA) guidelines to ensure the confidentiality, integrity, and availability of patient health information.

5.3. Security Requirements

[Privacy and data protection regulations that need to be adhered to while designing the product.]

- NFR4(R): The system will only be usable by authorized users.

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

- Load Balancing: Use load balancing to distribute user requests evenly across multiple servers to prevent overload and ensure consistent performance.

5.4.2. Correctness

- Data Validation: Implement rigorous data validation checks to ensure that medication information is accurate and consistent.

5.4.3. Maintainability

- Documentation: Maintain comprehensive documentation, including code comments, user manuals, and system architecture diagrams.

5.4.4. Reusability

- Standardization: Follow industry standards and best practices to ensure compatibility and ease of integration with external systems.

5.4.5. Portability

- Browser Compatibility: Ensure that the system's web-based components are compatible with multiple web browsers.

5.5. Process Requirements

5.5.1. Development Process Used

[Software Process Model]

Sprints

5.5.2. Time Constraints

FA Semester 2023

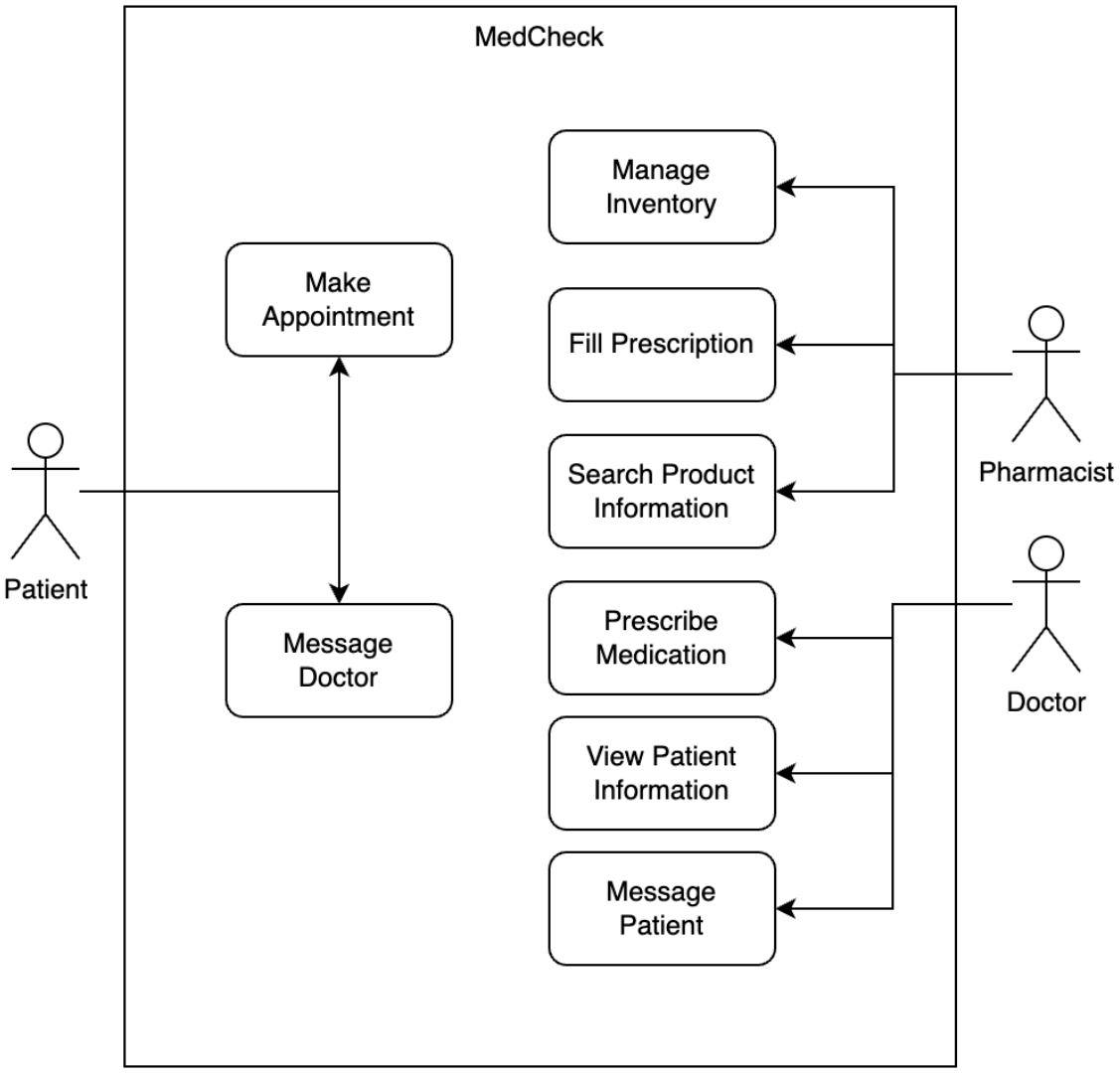
5.5.3. Cost and Delivery Date

Delivery: Dec 5, 2023

5.6. Other Requirements

None

5.7. Use-Case Model Diagram



5.8. Use-Case Model Descriptions

5.8.1. Actor: Patient (Noah Phillips)

- **Make Appointment**: In this Use-Case, Patient can schedule an appointment with a Doctor. This involves selecting a suitable date and time, specifying the reason for the visit, and possibly choosing a preferred doctor if options are available. The appointment request is then submitted for confirmation.
- **Message Doctor**: This use case enables the Patient to send a message to a Doctor. It allows for secure communication between the patient and their healthcare provider. Patients can use this feature to ask medical questions, share updates on their condition, or request information related to their treatment.
- **View Patient Information:** This use case allows the Patient to view their own patient information, such as medical history, prescription records, test results, and personal details. Access to this information helps them stay informed about his health status and treatment.

- **View Product Information:** This use case allows Patient to view information about healthcare products, medications, or medical devices. They can access details such as product descriptions, usage instructions, potential side effects, and availability.

5.8.2. Actor: Doctor (Dev Patel)

- **View Patient Information:** This use case allows the Doctor to view their patients' information, such as medical history, prescription records, test results, and personal details. Access to this information helps them stay informed about his health status and treatment.
- **View Product Information:** This use case allows the Doctor to view information about healthcare products, medications, or medical devices that they can prescribe to the patients. They can access details such as product descriptions, usage instructions, potential side effects, and availability.
- **Message Patient**: This use case enables the Doctor to send a message to a Patient. It allows for secure communication between the doctor and their Patients. Doctors can use this feature to ask medical questions, share updates on their diagnosis, or request information related to their health.
- **Prescribe Medication:** This use case enables the Doctor to prescribe medication to patients.
- **View Conflict Resolution**: By accessing this information, doctors can prevent adverse drug reactions, identify potential drug misuse or abuse, and provide appropriate counseling to patients.

5.8.3. Actor: Pharmacist (Andy Salinas)

- **Manage Medication Stock:** Enables the pharmacist to stay up-to-date with medications. This involves ensuring the availability of medications while minimizing waste and cost. By managing stock efficiently, pharmacists can ensure that patients receive their prescribed medications promptly and maintain a well-organized pharmacy environment.
- **Fill Prescription**: Once the prescription is filled, the pharmacist must label it correctly with the patient's name, dosage instructions, and other relevant information. They must also provide counseling to the patient on how to take the medication safely and effectively.
- **View Product Information**: Allows pharmacists to access comprehensive details about medications, including their composition, dosage, indications, contraindications, side effects, and interactions
- **View Conflict Resolution**: By accessing this information, pharmacists can prevent adverse drug reactions, identify potential drug misuse or abuse, and provide appropriate counseling to patients.

5.9. Use-Case Model Scenarios

5.9.1. Actor: Patient (Noah Phillips)

- **Use-Case Name**: N/A
    - **Initial Assumption**: N/A
    - **Normal**: N/A
    - **What Can Go Wrong:** N/A
    - **Other Activities**: N/A
    - **System State on Completion**: N/A
- **Use-Case Name**: N/A
    - **Initial Assumption**: N/A
    - **Normal**: N/A

- **What Can Go Wrong**: N/A
- **Other Activities**: N/A
- **System State on Completion**: N/A

5.9.2. Actor: Doctor (Dev Patel)

- **Use-Case Name**: Message patient
  - **Initial Assumption**: You want to message a patient
  - **Normal**: You can message a patient
  - **What Can Go Wrong**: Patient disconnects
  - **Other Activities**: None
  - **System State on Completion**: Complete
- **Use-Case Name**: View patient information
  - **Initial Assumption**: You want to view a patient's information
  - **Normal**: View patient's information when you click their name
  - **What Can Go Wrong**: Patient deletes account
  - **Other Activities**: None
  - **System State on Completion**: Complete

5.9.3. Actor: Pharmacist (Andy Salinas)

- **Use-Case Name**: Manage Inventory
  - **Initial Assumption**: You want to manage your medication inventory
  - **Normal**: Add, edit, and remove stocks of medication
  - **What Can Go Wrong**: Editing or removing a product not available will result in error.
  - **Other Activities**: None
  - **System State on Completion**: Complete
- **Use-Case Name**: Search Product Information
  - **Initial Assumption**: You want to look up information about a medication
  - **Normal**: View information provided by API
  - **What Can Go Wrong**: Search query is not available
  - **Other Activities**: None
  - **System State on Completion**: Complete

6. Design Documents

6.1. Software Architecture

Presentation Layer:

- Java Classes:
  - PatientUI
  - DoctorUI
  - PharmacistUI
- Functionality: These classes handle the user interfaces for patients, doctors, and pharmacists, allowing them to interact with the system for tasks like making appointments, messaging, and viewing information.

Application Layer:

- Java Classes:
  - PatientManager
  - DoctorManager
  - PharmacistManager
  - MedicationManager
  - AppointmentManager
- Functionality: These classes manage the core application logic, coordinating user requests and interactions with the data layer. Each class handles functionalities specific to its user role, such as medication stock management, prescription filling, medication prescribing, and appointment scheduling.

Business Logic Layer:

- Java Classes:
  - MedicationConflictDetector
  - MedicationInformationProvider
- Functionality: These classes contain the business logic for detecting medication conflicts, providing medication information, and supporting decision-making processes for doctors and pharmacists. They automatically notify relevant parties in case of medication conflicts.

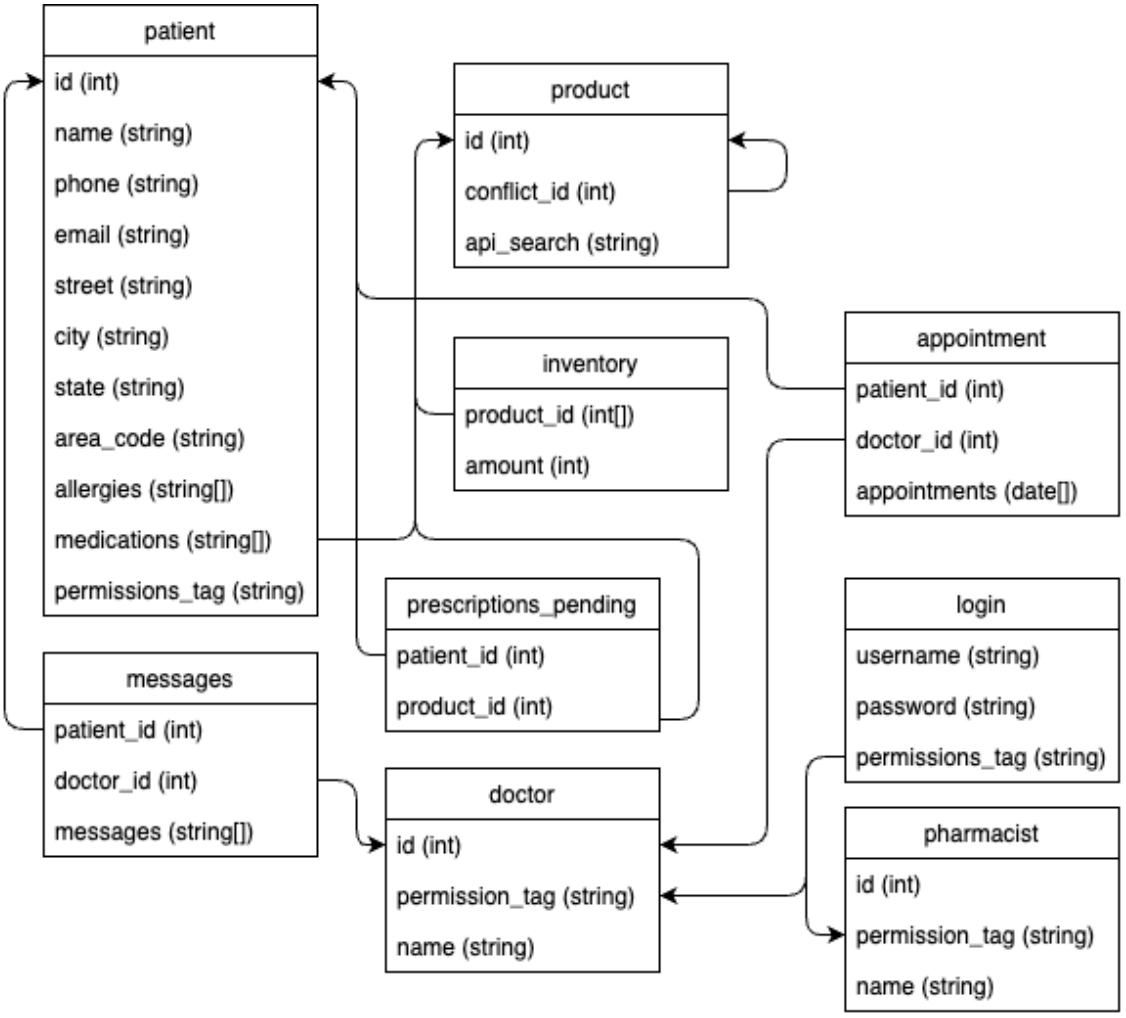
Data Access Layer:

- Java Classes:
  - DatabaseAccess
- Functionality: This class manages data access, including reading and writing data to and from the database. It interacts with the database to store and retrieve patient information, medication data, and more.

Database:

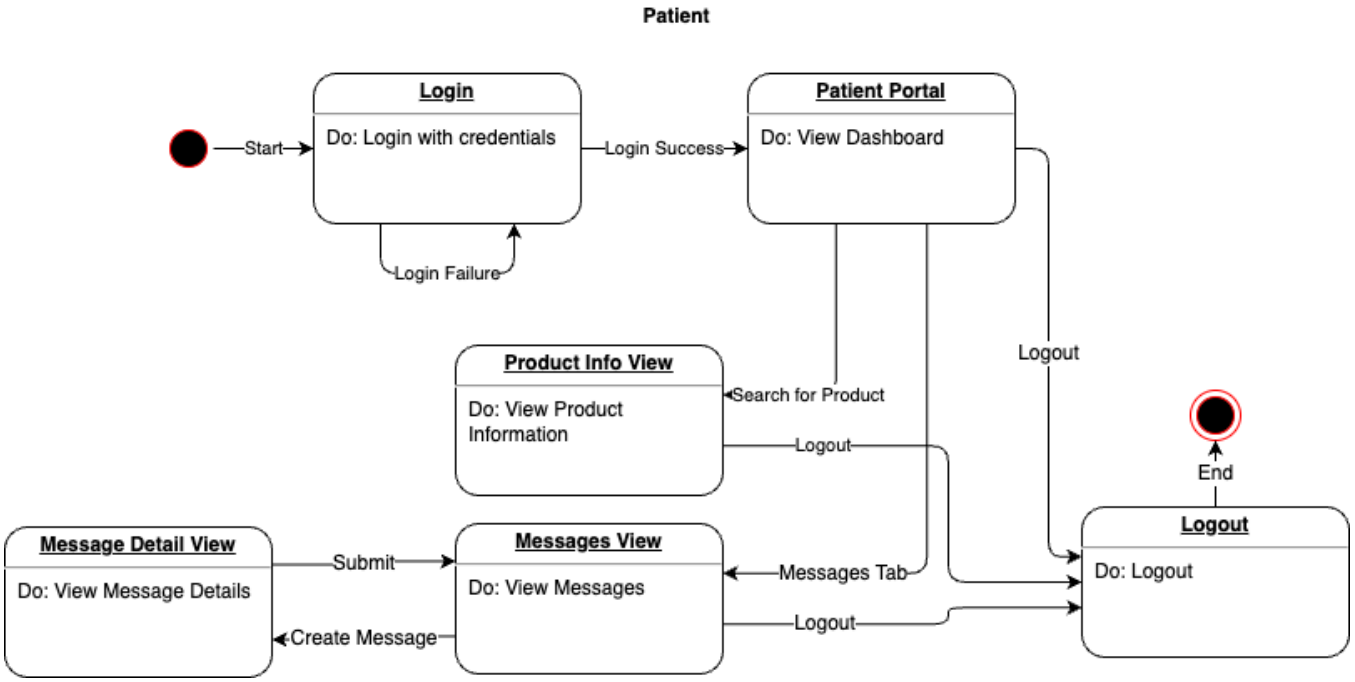
- Database Management System (e.g., SQL Server, MySQL)
- Functionality: The database stores data related to patients, medications, appointments, doctors, pharmacists, and medication conflicts.

6.2. High-Level Database Schema

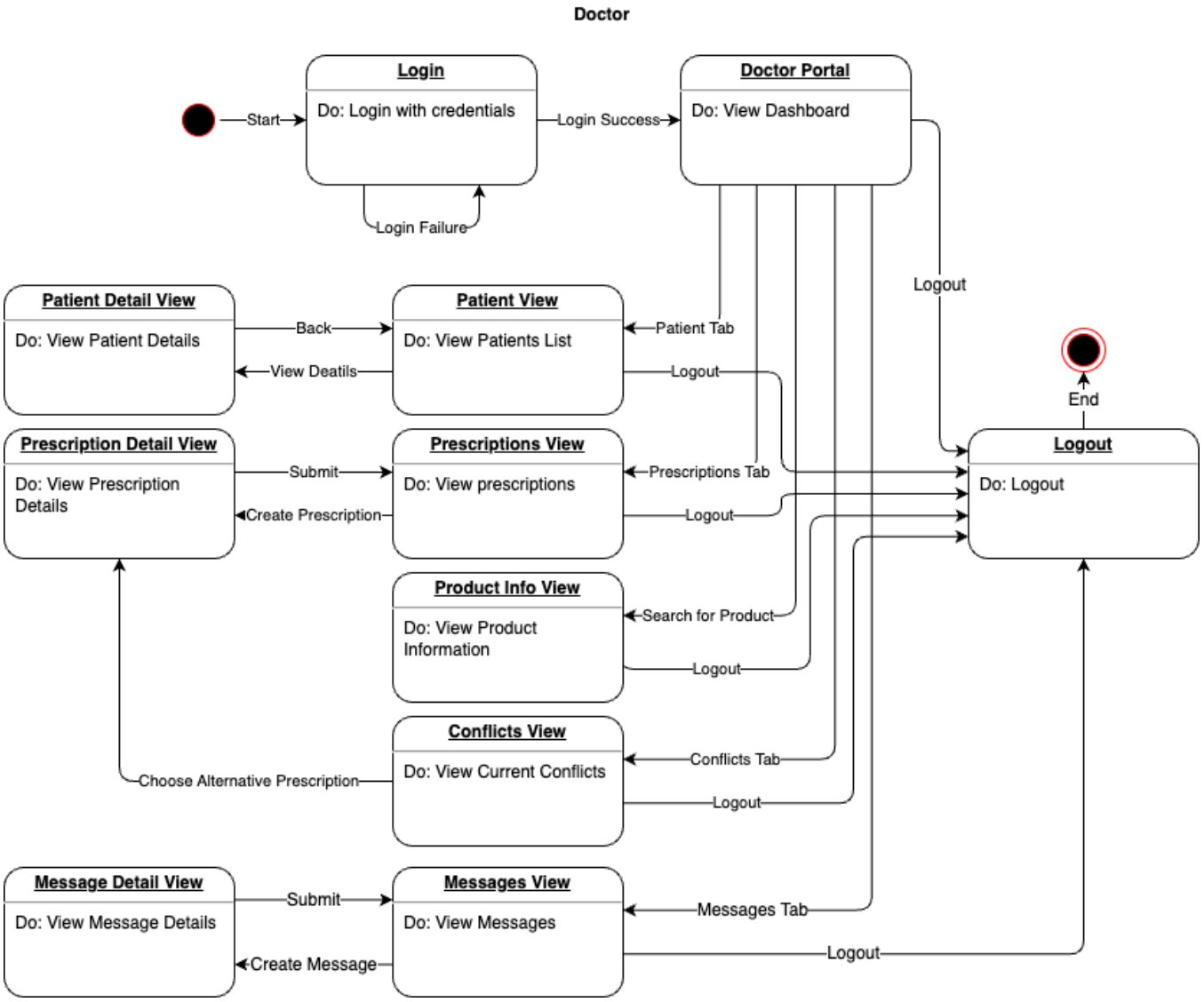


6.3. Software Design

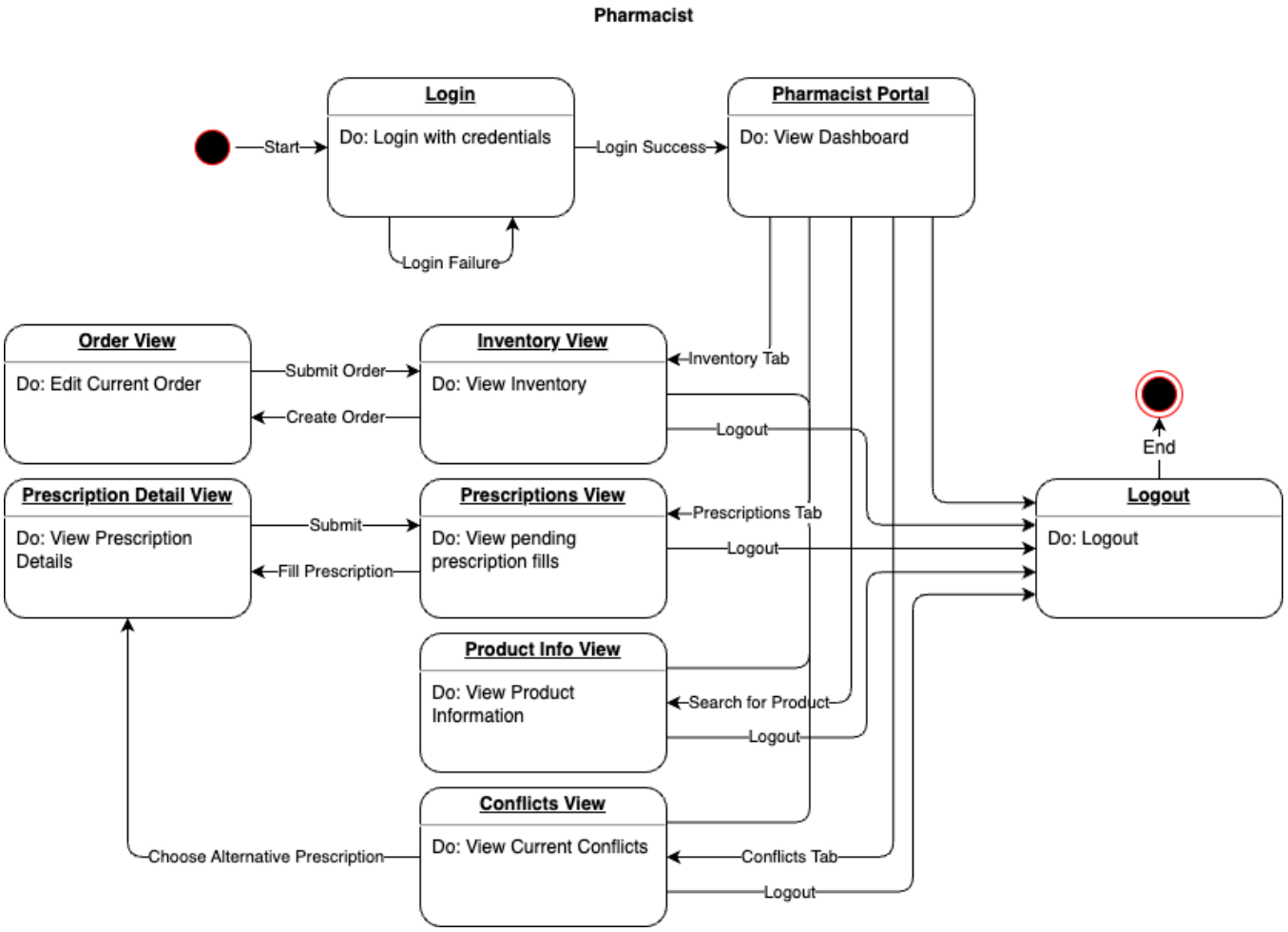
6.3.1. State Machine Diagram: Patient (Responsible Team Member)



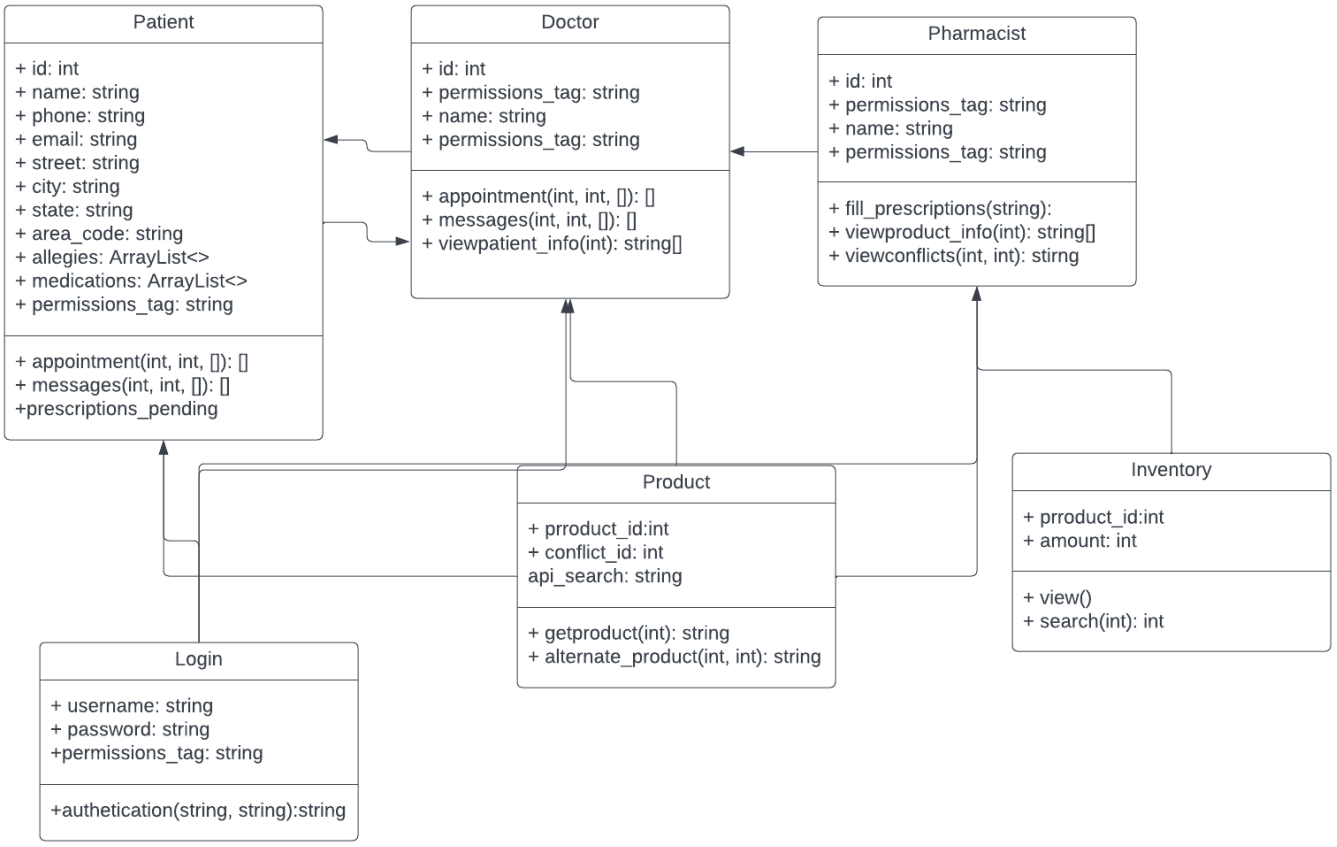
6.3.2. State Machine Diagram: Doctor (Responsible Team Member)



6.3.3. State Machine Diagram: Pharmacist (Andy Salinas)



6.4. UML Class Diagram



7. Scenario

7.1. Brief Written Scenario with Screenshots

A doctor signs up and has a list of patients. They can then message patients, view their information or create prescriptions for them. A pharmacist would also sign up. They can then see the prescriptions made by the doctor and fill them as needed to patients. They can also manage the inventory of medications and view product information.