SOFTWARE REQUIREMENTS SPECIFICATION

for

PATIENT PORTAL (2mg)

Version 1.3

Prepared by:
Kshitiz Bhargava (21BCE2067)

Date: 25 March 2024

Contents

Document Version Control:

Full Name	Date	Change Comments	New Version	Previous Version
Kshitiz Bhargava	16.01.2024	Initial Version	V1.0	NA
Kshitiz Bhargava	23.01.2024	Work breakdown structure was added	V1.1	V1.0
Kshitiz Bhargava	28.02.2024	Use Case Diagram, Class Diagram and Entity Relation Diagram was added	V1.2	V1.1
Kshitiz Bhargava	25.03.2024	Data Flow Diagram of Level 0, 1 and 2, Sequence Diagram, State Chart Diagram, Collaboration Diagram and User Interface Diagram has been added	V1.3	V1.2

Requirements Version Control:

Full Name	Date	Change Comments	New Version	Previous Version
Kshitiz Bhargava	16.01.2024	Initial specification of requirements	V1.0	NA
Kshitiz Bhargava	23.01.2024	Refining initial requirements according to criteria of good requirements	V1.1	V1.0
Kshitiz Bhargava	28.02.2024	Functional Requirements were added	V1.2	V1.1
Kshitiz Bhargava	25.03.2024	Solution oriented requirement has been added	V1.3	V1.2

1 Introduction

1.1 Purpose of the SRS

This Software Requirements Specification is used to specify a web application called PATIENT PORTAL. The current version of PATIENT PORTAL is 1.1. This SRS not only includes the basic requirements of the PATIENT PORTAL to meet the clinic daily usage and but also deal with the alternative situations such as password reset; activate, deactivate, and delete patient and employee accounts.

1.2 Scope

The PATIENT PORTAL is a web application that manages electronic patient records to provide rapid and efficient means to handle medical information of a clinic. This SRS has specified 22 use cases and including actors such as patient, nurse, physician, and receptionist, etc. It is the first document for PATIENT PORTAL. No other references are available right now.

1.3 Acronyms, abbreviations, notational conventions

None.

1.4 Overview

This is a partial SRS. It includes a UML use case diagram showing all identified use cases with a brief description of the use case diagram; A UML class diagram for the system showing all identified classes and their attributes and a brief description of the class diagram; Three state diagrams of Appointment, Bill Payment and Doctor Calendar; Sequence diagrams for all the use cases; Non-functional requirements; UI descriptions; One backlog with story points and priority; And brief descriptions of all stories.

1.5 References

- Bhat, Mohammad Waqar & ST, Veerabhadrappa & Bhushan, Himanshu & Shrivastava, Kartik & Sahoo, Ashish. (2020). Secure Online Medicine Delivery System. Review of Computer Engineering Studies. 7. 74-78. 10.18280/rces.070305. [https://www.researchgate.net/publication/344609028_Secure_Online_Medicine_Delivery_System]
- Onuiri, Ernest & Sam-David, Chukwujioke. (2016). Online Pharmaceutical Management System. European Scientific Journal. 12. 1857-7881. 10.19044/esj.2016.v12n12p139. [https://www.researchgate.net/publication/305426561_Online_Pharmaceutical_Management_System]

2 General description

2.1 Product perspective: the environment

The overall goal of the PATIENT PORTAL is to provide physicians, nurses, and clinical staff with a powerful, easy-to-use tool that securely assists them in gathering, storing, and manipulating patients' information. The PATIENT PORTAL to-be-built will facilitate a clinic to provide a dependable and effective health care to the patients at reduced cost while respecting the privacy of the patients.

2.2 Product functions

- The PATIENT PORTAL should handle medical and patient's information of a clinic efficiently and securely.
- PATIENT PORTAL should be able to handle medical records in a variety of forms, such as images, scanned scripts, and audio recording.
- The PATIENT PORTAL shall enable patients to schedule appointments with the doctors at the clinic.
- Using the PATIENT PORTAL, patients can request appointments with the doctors by exchanging messages with receptionists. Also, the PATIENT PORTAL shall send a reminder to a patient 24 hours before the appointment time.
- The PATIENT PORTAL shall assist a patient's office visit, enabling a nurse to enter the system the patient's current medical conditions, such as the blood pressure and the weight. A doctor's diagnosis shall be entered into the patient's chart.
- PATIENT PORTAL shall support financial billing applications, which will collect the co-payments and transmit the appropriate information to the patients' insurance agencies.
- The PATIENT PORTAL shall ensure the storage and processing of medical records are regulatory compliant.

2.3 User characteristics

The User like physicians, receptionist and nurses will be trained by us either training sessions or online training videos on how to use the Patient Portal and what are the features for and scope of each role onto the portal. User like Patients should have basic knowledge about accessing websites and use of internet. They will also have the access to questionnaire onto the portal to understand the interfaces, usage, and scope of the product.

2.4 General constraints

The product must conform to requirements as specified to be accepted.

2.5 Assumptions and Dependencies

Since the Patient Portal is only accessible through the Internet, it is assumed that the end user has a connection to the Internet. It is also assumed that the user has a compatible web browser and updated relevant browser plugins to display the website and its contents.

2.6 Process Model

Agile methodology emphasizes iterative development, where requirements and solutions evolve through collaboration between cross-functional teams. It advocates adaptive planning, evolutionary development, early delivery, and continual improvement.

Requirements Gathering and Analysis:

- Collaborative Requirements Identification: In Agile, requirements are gathered in a collaborative manner with stakeholders, including customers and end-users.
- User Stories: Requirements are often captured as 'user stories' that focus on user needs.
- Backlog Creation: These stories are then added to a product backlog.
- Prioritization: The backlog items are prioritized based on business value, technical feasibility, or stakeholder urgency.

System Design:

- Iterative Design: Design is not a one-time phase but happens iteratively as the project progresses.
- Design Meetings: Regular meetings with the development team and stakeholders help refine the design.
- Continuous Feedback: Design evolves based on continuous feedback from iterations (sprints).

Implementation (Coding):

- Sprints: Work is divided into iterations called 'sprints', typically lasting 2-4 weeks.
- Daily Stand-ups: Short daily meetings help track progress and address any issues.
- Pair Programming: This technique may be used for enhanced quality and knowledge sharing.

Testing:

- Continuous Testing: Testing is integrated throughout the development cycle.
- Test-Driven Development (TDD): Writing tests before code ensures adherence to requirements.
- Automated Testing: Automation is often used to efficiently handle repetitive tests.

Deployment:

- Frequent Releases: Agile aims for frequent and incremental releases of the product.
- Continuous Integration/Continuous Deployment (CI/CD): These practices are commonly used for rapid and reliable software release.

Maintenance:

- Ongoing Development: Maintenance is a continuous process as the system evolves.
- Adaptability: Agile allows for easy adaptation to changes even in the maintenance phase.

Characteristics of the Agile Model:

- Adaptive and Flexible: Agile is well-suited for projects where requirements are expected to change.
- Customer-Centric: Focuses on customer needs and satisfaction.
- Incremental Delivery: Products are built incrementally, providing functional components from early in the development process.
- Collaborative Approach: Emphasizes teamwork and stakeholder involvement.
- Transparency: Regular updates and demonstrations ensure high visibility of project progress.

Advantages of the Agile Model:

- Flexibility: Easily accommodates changes in requirements.
- Risk Management: Early and frequent releases help in identifying and addressing risks promptly.
- Customer Satisfaction: Continuous engagement with the customer ensures that the final product meets their needs.
- Improved Quality: Frequent testing and reviews lead to higher quality products.
- Efficient Feedback Loop: Allows for immediate feedback and continuous improvement.

Disadvantages of the Agile Model:

- Less Predictability: Due to its iterative nature, the final product and delivery date may not be clear at the beginning.
- Scope Creep: Frequent changes can lead to scope creep if not properly managed.
- Requires Active Customer Involvement: Success depends heavily on the customer's

involvement and feedback.
- Not Suitable for All Types of Projects: Agile may not be well-suited for projects with fixed requirements or those that are heavily regulatory.
- Resource Intensive: Requires more time and effort from developers and stakeholders.

2.7 Project Scheduling

This section introduces the significance of project scheduling in the successful implementation of 2mg, ensuring timely delivery and effective resource management.

2.7.1 Timeline Overview

- Project Duration: Estimated 4 months

Phase-wise Breakdown:

1. Requirements Analysis
- Requirement gathering
- Stakeholder reviews and approval

2. System Design
- Initial design development
- Iterative design improvements based on feedback

3. Implementation
- Coding phases
- Internal code reviews and iterations

4. Testing
- Unit Testing
- Integration Testing
- System Testing
- User Acceptance Testing (UAT)

5. Deployment
- Beta Testing
- Full Deployment

6. Maintenance
- Regular updates
- Ongoing support

2.7.2 Conclusion:

Adherence to this schedule is critical for the timely and successful completion of 2mg. This project scheduling adds a structured approach to the timeline and resource management for the 2mg project, aligning with the requirements and goals outlined in the SRS.

2.7.3 Gantt Chart

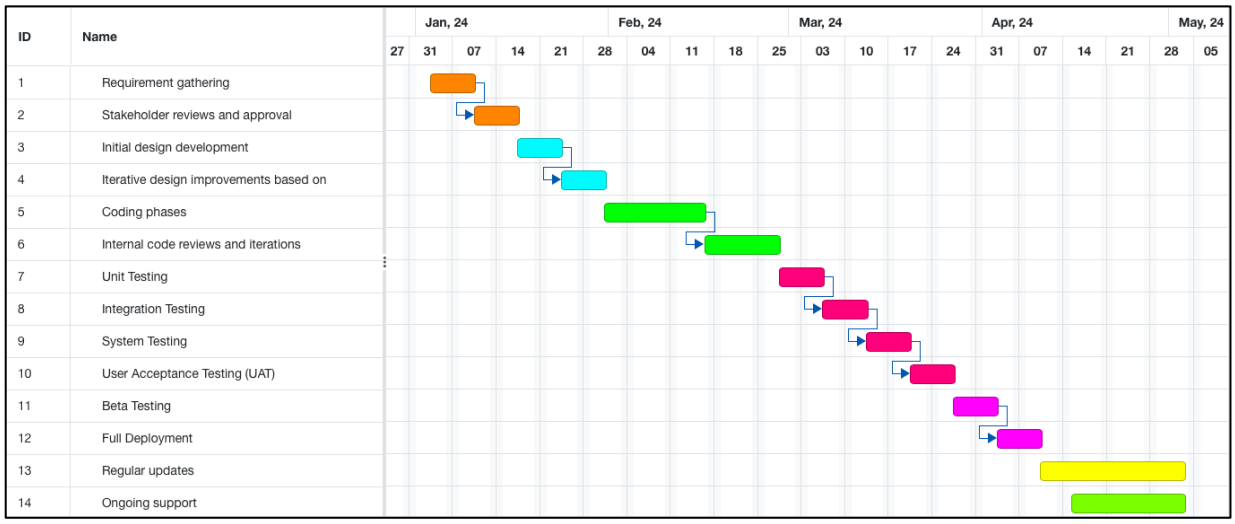

Figure 2.1: Gantt Chart

2.8 Work Breakdown Structure

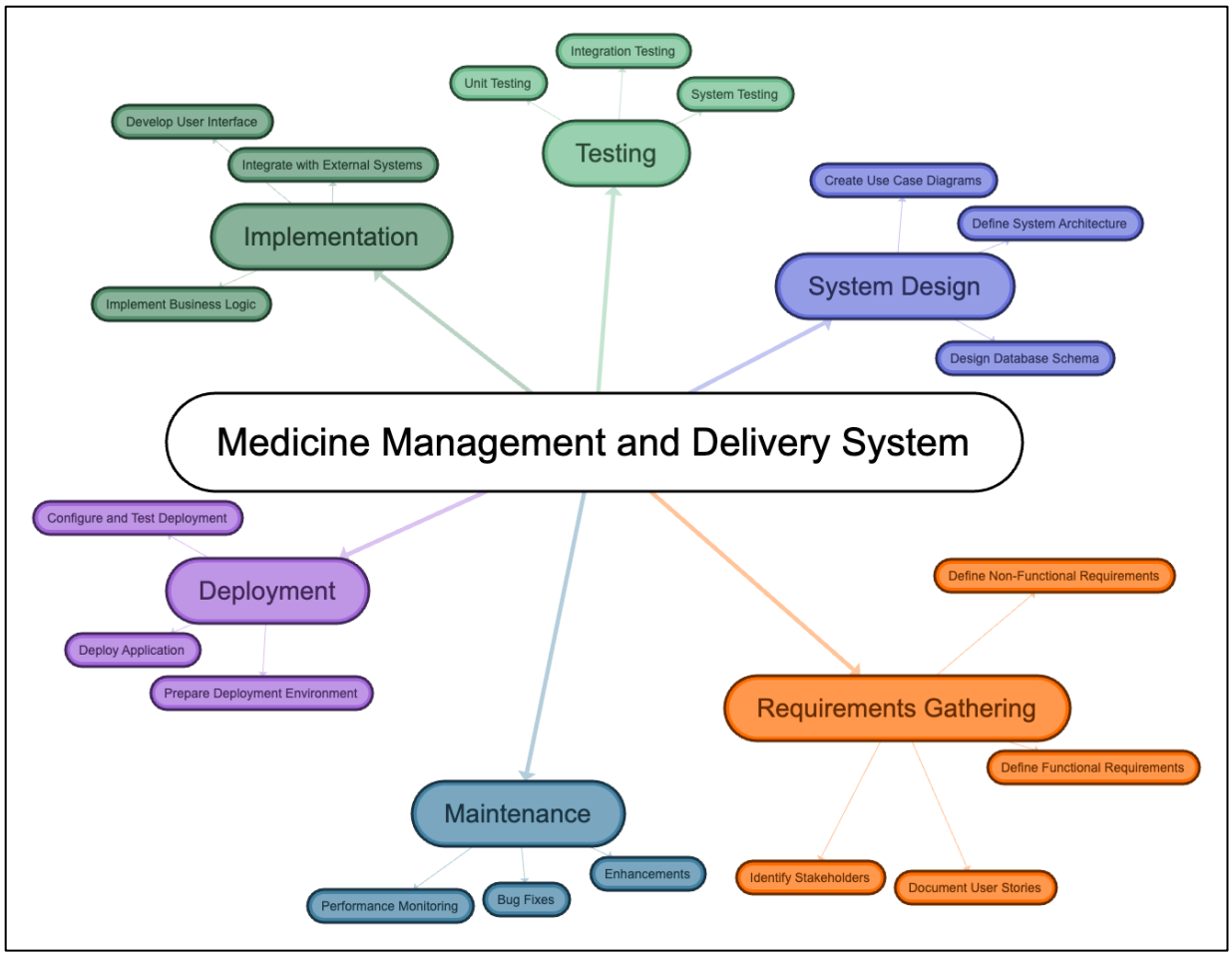

Figure 2.2: Patient Portal Work breakdown structure

3 Specific Requirements

3.1 Functional requirements

3.1.1 Use Case diagram

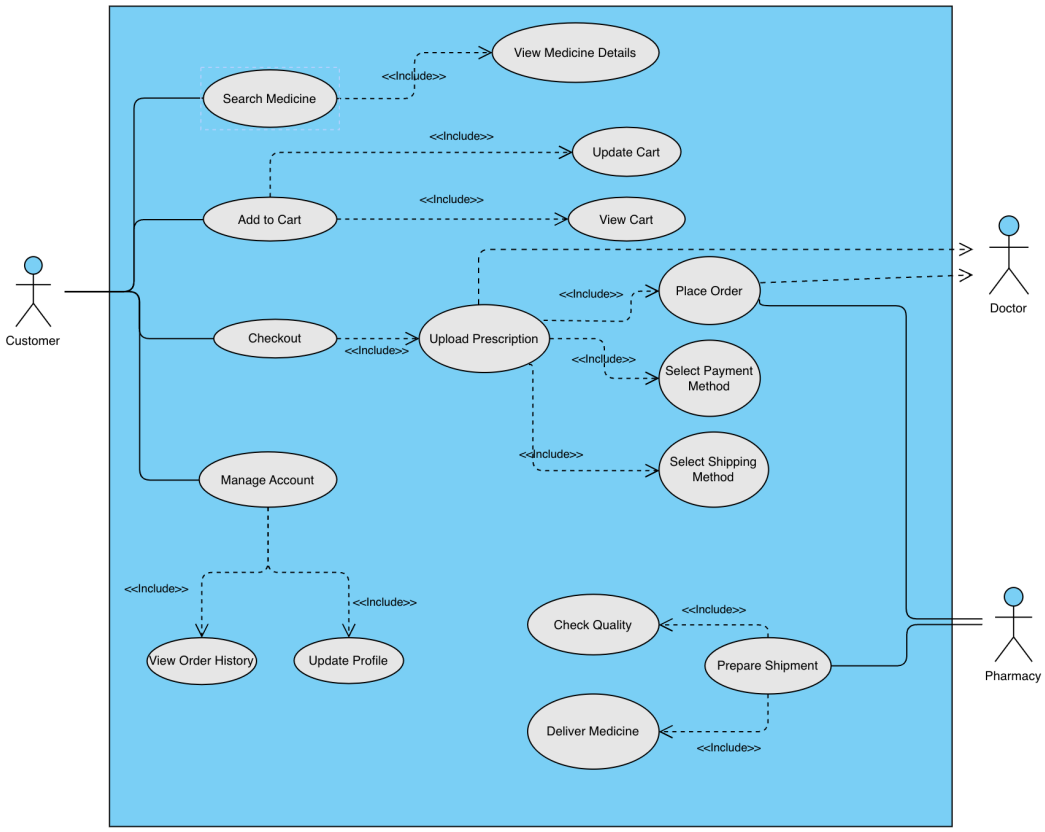

Figure 3.1: Use Case Diagram of 2mg

Figure 3.1 shows the Use Case diagram of 2mg web application. The main actors in this diagram are Customer, Doctor, and Pharmacy.

The main functions or use cases are:

1. Search Medicine - where the customer can look for medicine they need.
2. Add to Cart - allowing the customer to select medicines and add them to a shopping cart.
3. View Medicine Details - where details of a selected medicine can be viewed.
4. Update Cart - to change the quantity or remove items from the cart.
5. Checkout - the process of finalizing the order, which includes several sub-processes like uploading a prescription, viewing the cart, placing an order, selecting payment and shipping methods.
6. Manage Account - where the customer can view their order history, update their profile, etc.
7. Check Quality - a process likely performed by the pharmacy to ensure the quality of medicines before shipment.
8. Prepare Shipment - the process of preparing the ordered medicine for delivery.
9. Deliver Medicine - the final step where the medicine is delivered to the customer or the prescribing doctor.

3.1.2 Class Diagrams

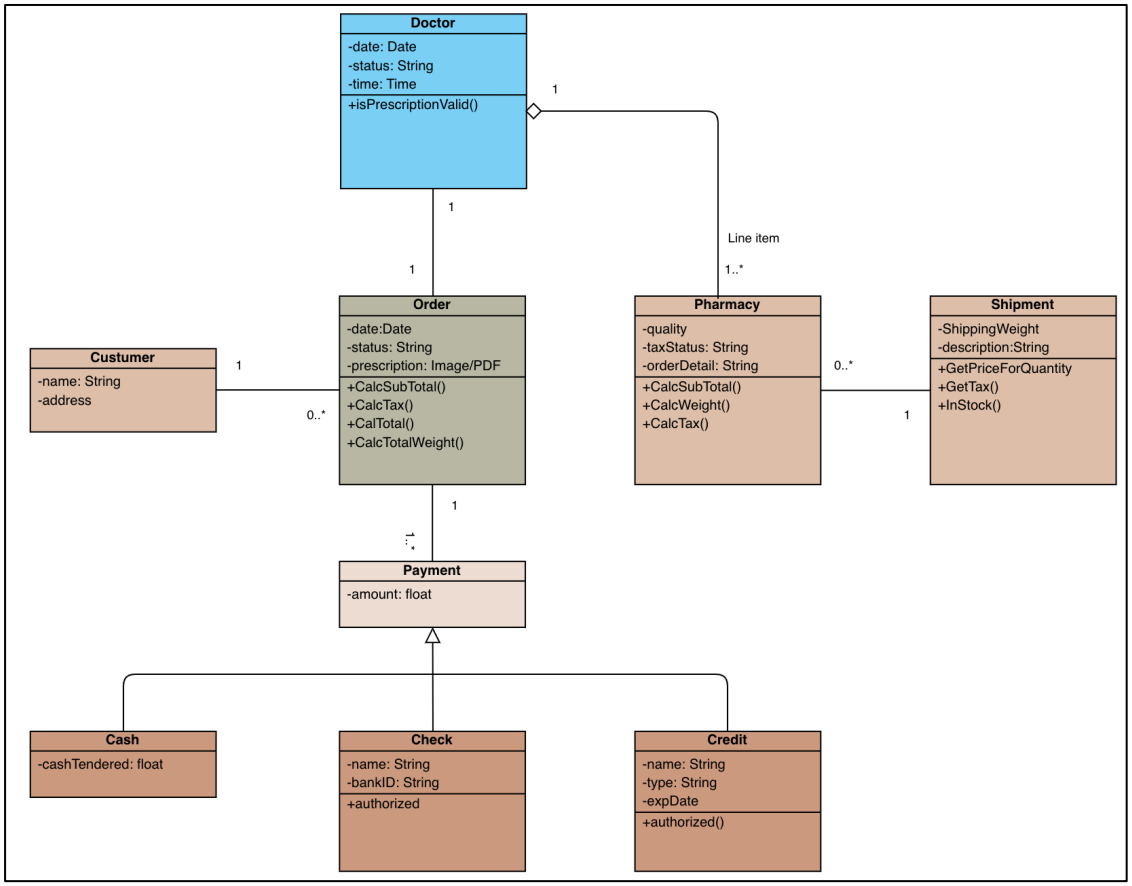

Figure 3.2: Class Diagram of 2mg

Figure 3.2 shows the class diagram of 2mg web application.

Brief Description of each class:

1. Doctor: Has attributes like date, status, time, and a method to check if a prescription is valid.
2. Customer: Stores the customer's name and address.
3. Order: Contains details such as date, status, prescription, and methods to calculate subtotal, tax, total, and total weight.
4. Payment: An abstract class with an attribute for the amount. This seems to be a parent class for different payment types.
5. Cash: Inherits from Payment and has an attribute for the cash tendered.
6. Check: Inherits from Payment and has attributes for name, bank ID, and a method to check authorization.
7. Credit: Inherits from Payment and includes attributes for name, type, expiration date, and an authorization method.
8. Pharmacy: Contains attributes to store the quality of the medicine, tax status, order details, and methods for calculations like the Order class.
9. Shipment: Deals with shipping, having attributes for weight and description, and methods to get price for a quantity, tax, and check stock availability.
10. Line item: No attributes or methods are displayed for this class, but it's related to Shipment, indicating it could represent individual items in a shipment.

3.1.3 Entity Relation Diagram

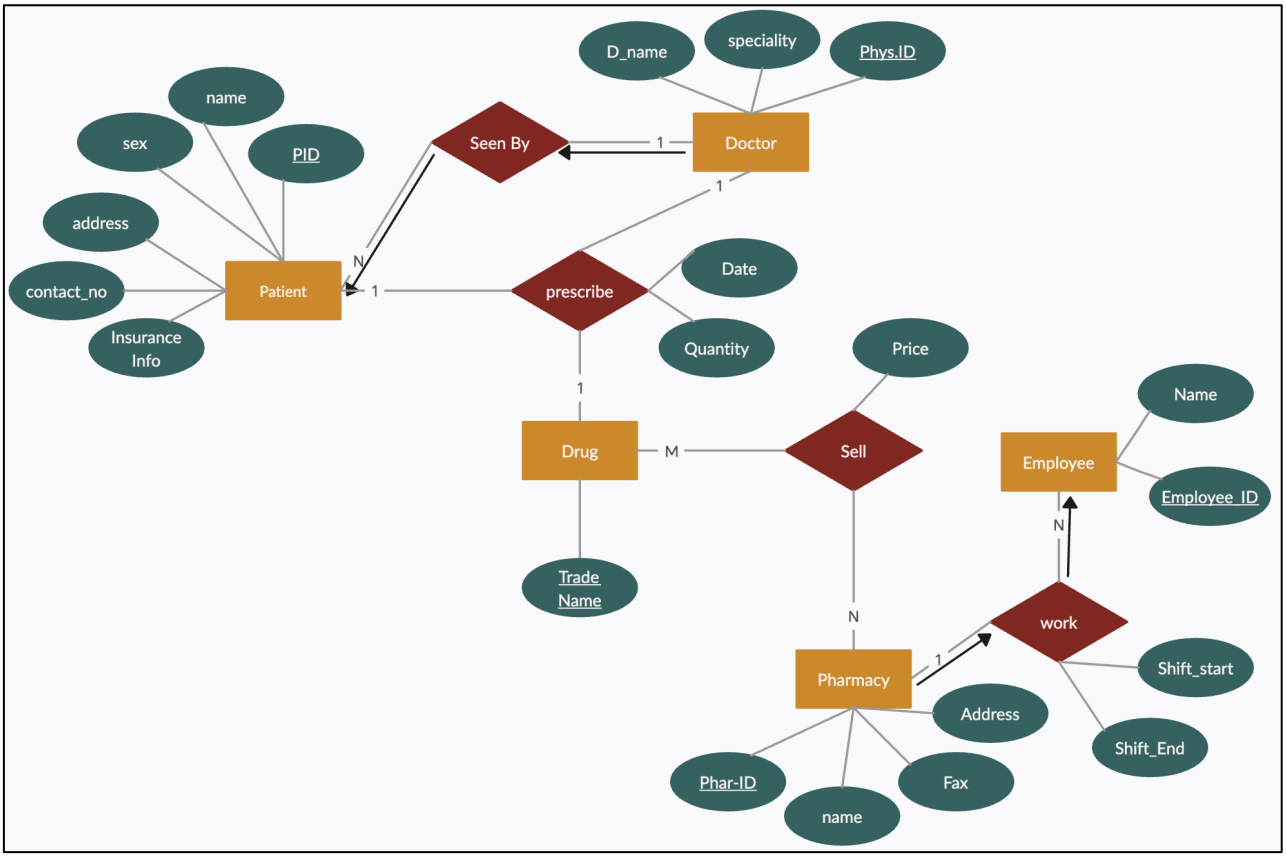

Figure 3.3: ER Diagram of 2mg

Figure 3.3 shows the ER Diagram of 2mg web application.

1. Patient: This entity has attributes like name, sex, address, contact number, insurance information, and a patient ID (PID). It is related to the "Doctor" entity through a "Seen By" relationship, which suggests that a patient is seen by one or more doctors.
2. Doctor: The attributes include doctor name (D_name), specialty, and physician ID (Phys_ID). The "Doctor" entity has a one-to-one relationship with "prescribe," indicating that a doctor can prescribe a drug.
3. Prescribe: This is a relationship rather than an entity, and it contains attributes like date, quantity, and price. It connects the "Doctor" entity to the "Drug" entity and suggests that a doctor prescribes drugs with specific quantities and prices on certain dates.
4. Drug: This entity holds information about drugs with attributes like trade name. It has a many-to-many relationship ("M") with "Sell," indicating that drugs can be sold many times.
5. Sell: Like "Prescribe," "Sell" is a relationship and not an entity. It shows that a drug can be sold by a pharmacy, possibly to many patients or by many employees.
6. Employee: This entity has attributes such as name and employee ID. It is connected to "Pharmacy" through a "work" relationship, indicating an employee works in a pharmacy and has details like shift start and shift end times.
7. Pharmacy: Contains details like pharmacy ID (Phar-ID), address, fax number, and name. It is related to both "Sell" and "Employee," indicating that it sells drugs and employs staff.

3.1.4 Activity Diagram

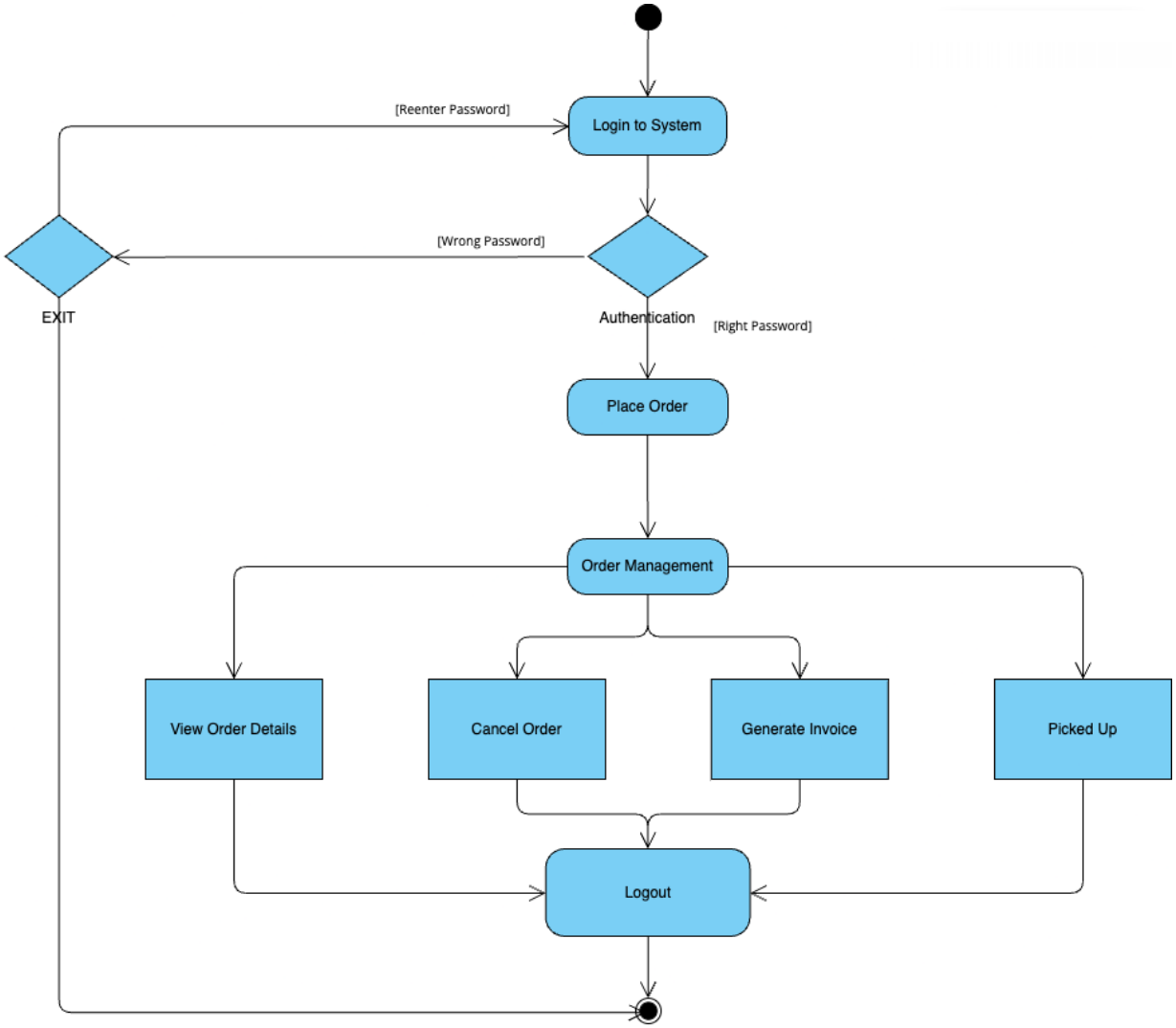

Figure 3.4: Activity Diagram

Breakdown:

1. Login to System: The process begins with an attempt to log into the system.
2. Authentication: There is a decision point after the login attempt. If the wrong password is entered, there is an option to re-enter the password. If the right password is entered, the process moves forward to "Place Order".
3. Place Order: After successful authentication, the user can place an order.
4. Order Management: Once an order is placed, the user enters the order management phase with several options:
5. View Order Details: The user can view details about the order.
6. Cancel Order: The user has the option to cancel the order.
7. Generate Invoice: The system can generate an invoice for the order.
8. Picked Up: This indicates that the order has been picked up, completing the order process.
9. Logout: After managing the order, the user can log out of the system.

1. **System Boundaries**: The system boundaries define the scope of the process. In the provided diagram, the system boundary is not explicitly drawn, but we can infer that it includes everything from "Login to System" to "Logout" including all the processes that happen in between. The boundary would exclude any external processes that are not shown within the flow, like what happens before "Login to System" and after "Logout".
2. **Activities**: These are the steps or functions that occur within the system. In your diagram, they are represented by rectangles with rounded corners (also known as action states in UML). Here are the activities:
    - **Login to System**: The starting point where a user attempts to log in.
    - **Place Order**: Once authenticated, the user can place an order.
    - **Order Management**: A central node that seems to manage various aspects of an order after it has been placed.
    - **Logout**: The end of the session, where the user exits the system.
3. **Sub-Activities**: These are more granular actions within the system, which are part of the main activities. In your diagram, sub-activities under "Order Management" are:
    - **View Order Details**: Checking the specifics of the order placed.
    - **Cancel Order**: Terminating the order before it is finalized or delivered.
    - **Generate Invoice**: Creating a bill for the order placed.
    - **Picked Up**: Indicates the completion of the order retrieval process.
4. **Relationships and Notations**:
    - **Diamonds (Rhombus)**: These represent decision points. In your diagram, there is one for "Authentication" which leads to different paths depending on whether the password is right or wrong.
    - **Arrows**: They show the flow of the process from one activity or decision to another.
    - **Cylinders**: These typically represent databases or data storage, but none are shown in your diagram.
    - **Annotations**:
        - **[Right Password]**: Indicates the condition under which the flow will proceed to "Place Order".
        - **[Wrong Password]**: Indicates the condition under which the flow will return to "Login to System".
        - **[Re-enter Password]**: This annotation is a bit unclear, but it suggests there may be a loop to re-enter the password if the first attempt is incorrect.

3.2 Requirements and Goal Modelling

3.2.1 Data Flow Diagram Level 0

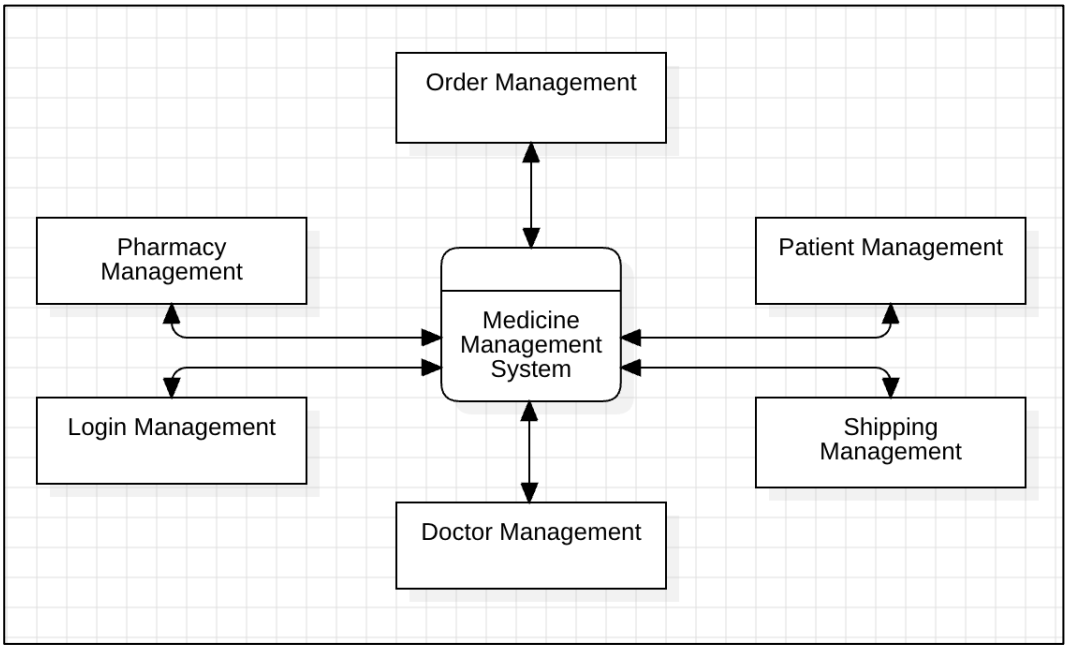

Figure 3.5: Data Flow Diagram Level 0

3.2.2 Data Flow Diagram Level 1

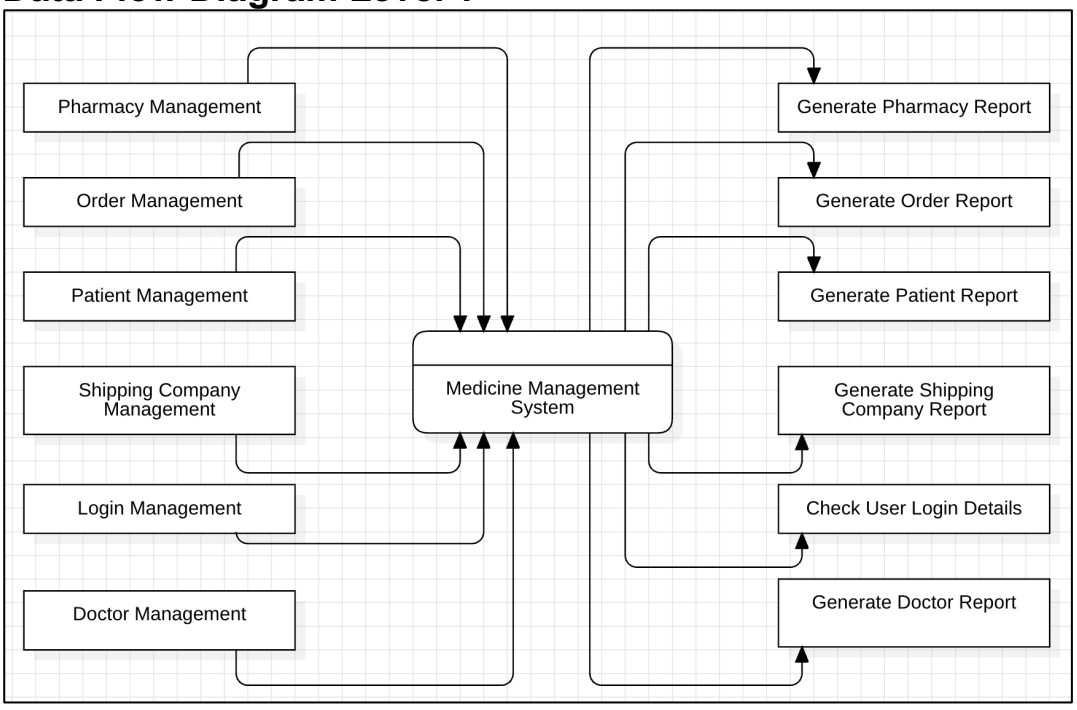

Figure 3.6: Data Flow Diagram Level 1

3.2.3 Data Flow Diagram Level 2

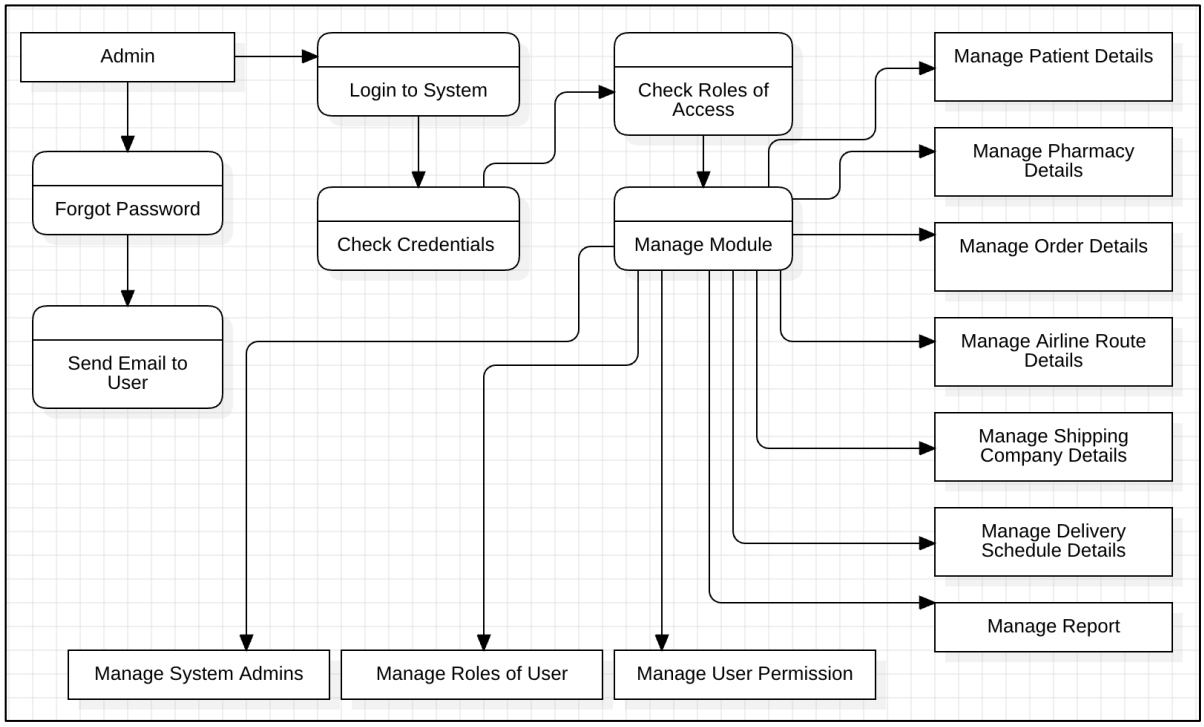

Figure 1: Data Flow Diagram Level 2

3.2.4 Sequence Diagram

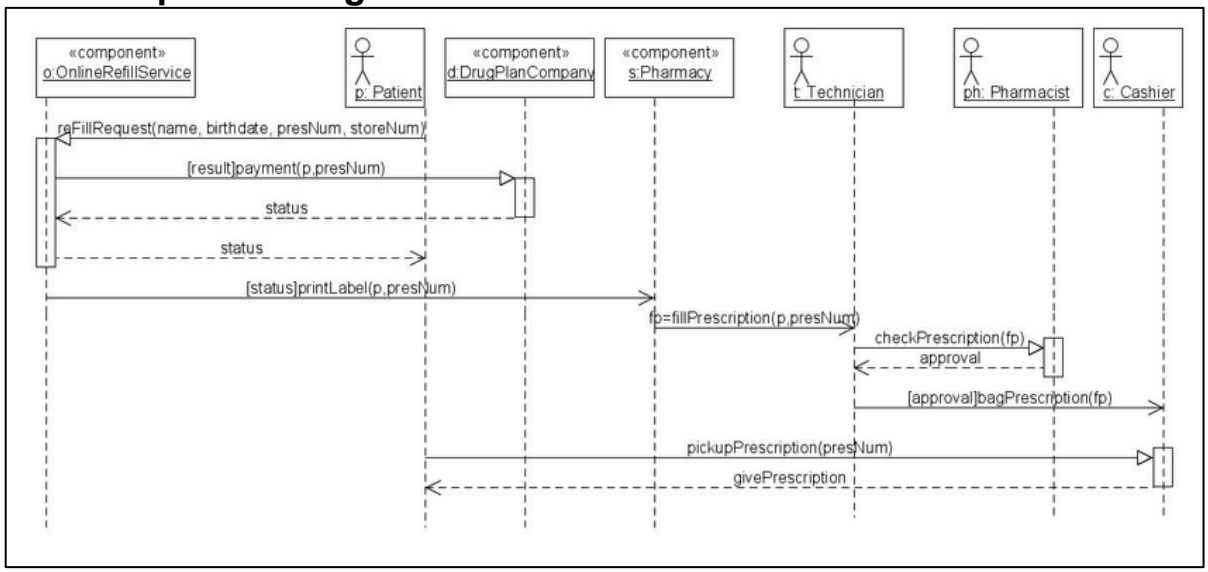

Figure 3.8: Sequence Diagram

3.2.5 State Chart Diagram

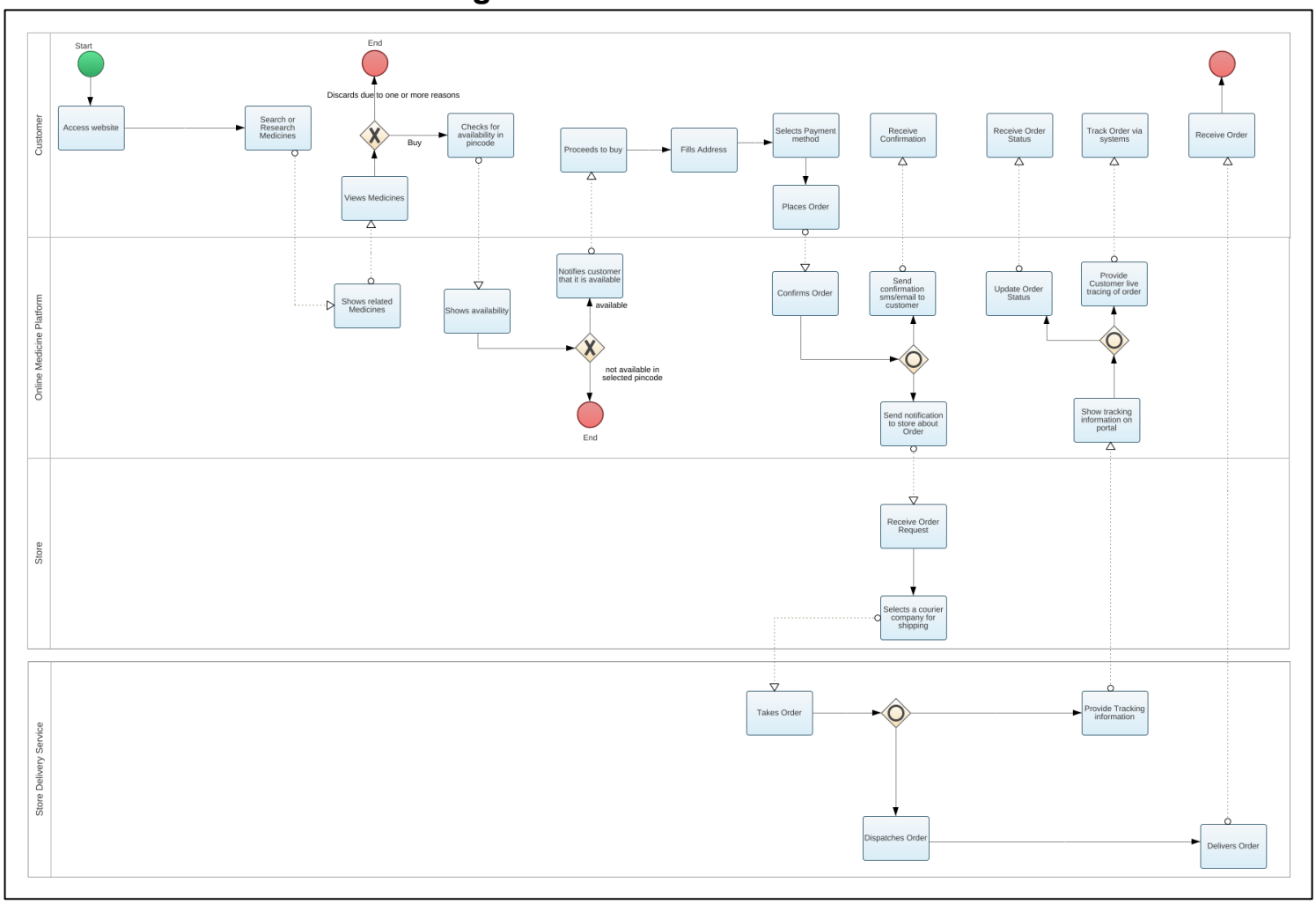

Figure 3.9: State Chart Diagram

3.2.6 Collaboration Diagram

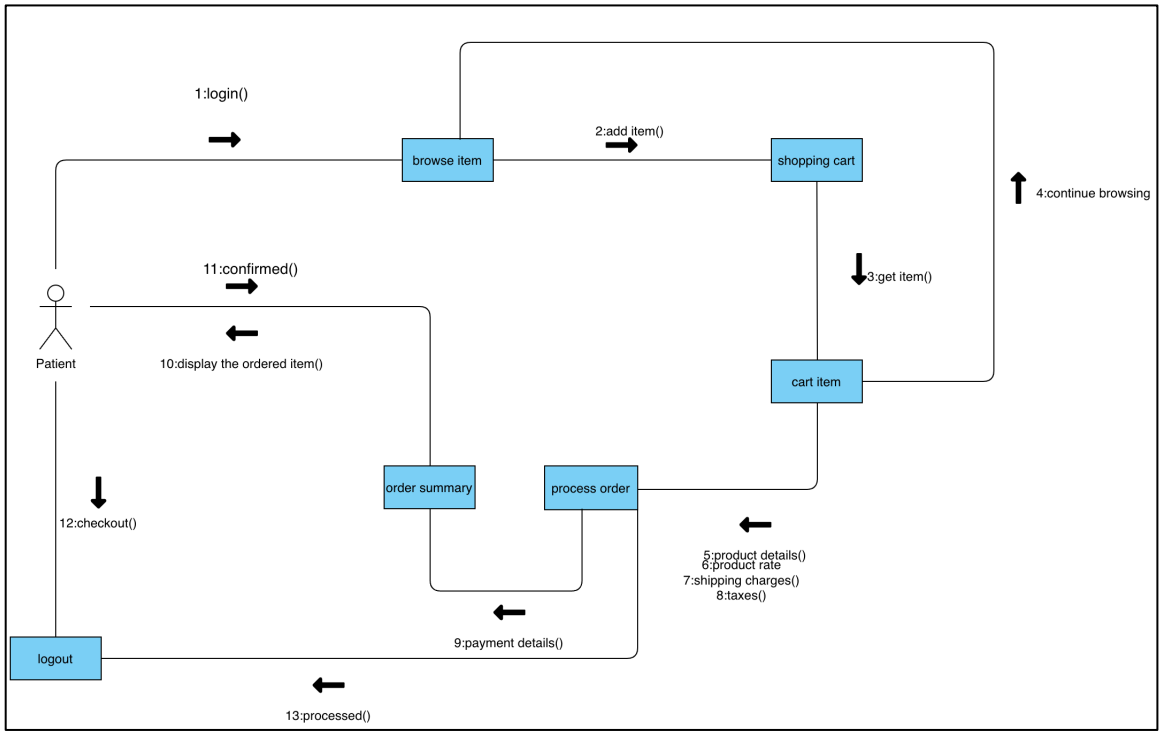

Figure 3.10: Collaboration Diagram for Ordering Medicine

3.2.7 User Interface Diagrams

Figure 3.11: Login Page

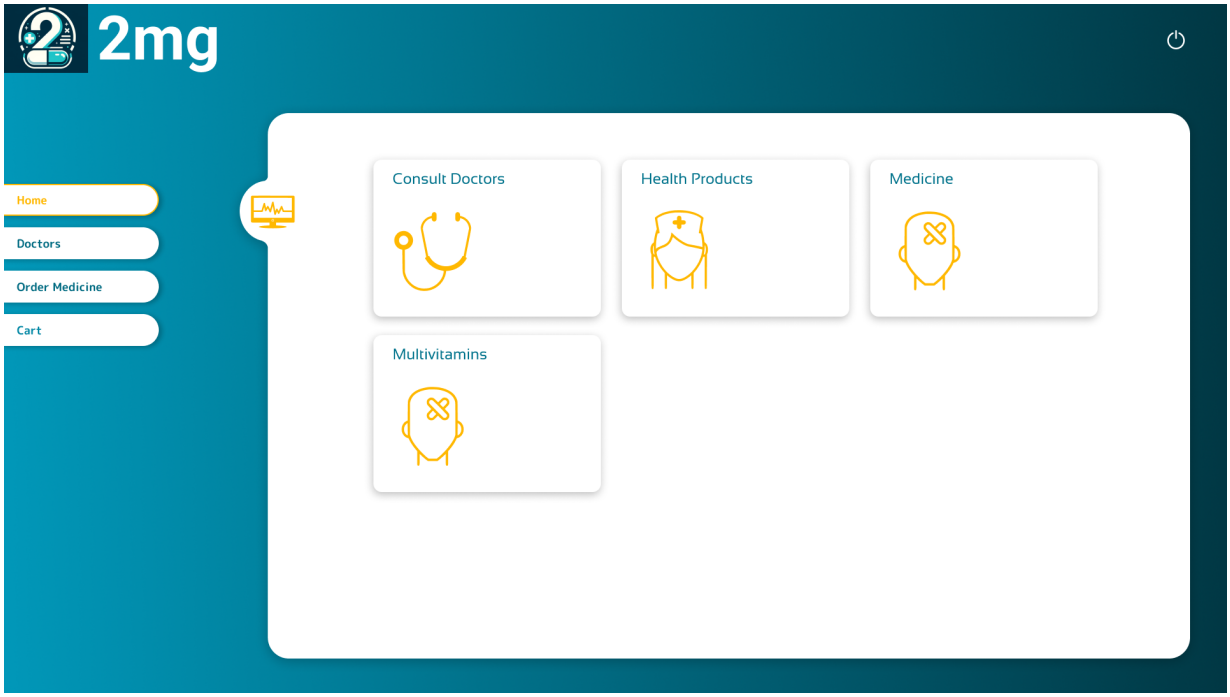

Figure 3.12: Patient Dashboard

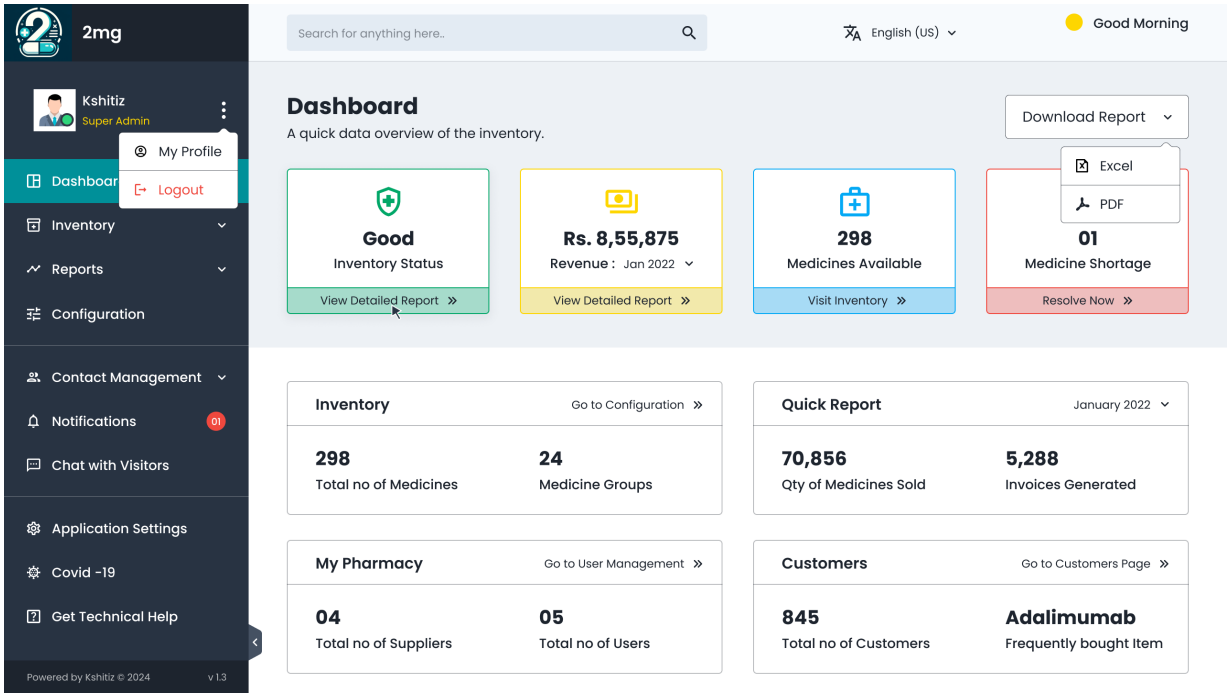

Figure 3.13: Admin Dashboard