Patients to Doctors - Final Project Report

Final Report for CSI308 Final Project

Author: Elia Ghazal - 22330018 , Charbel Abou Akl 22232006,
George Khayat 22232005

Instructor: Dr. Aziz Barbar

7 January, 2025

This report was submitted as the final project report for the course
Software Engineering and System Design (CSI308)

Department of Engineering and Computer Science
American University of Science and Technology
Zahle, Lebanon

Abstract

Across the world, there is many regons that lack accessibility to healthcare. Factors such as insufficient number of healthcare professionals, and the challenges of reaching remote areas creates a problem. The ”Connecting Patients to Doctors” project aims to address these problems by developing a web application that uses SMS technology, thereby eliminating the gap between patients and healthcare providers.

Through this playform, patients can submit their symptoms via SMS, which are then sent to appropriate healthcare professionals for diagnosis and treatment. The application supports remote patient monitoring through connected health devices, therefore providing real-time updates.

Developed under an Agile Incremental process model, the project showcases adaptability and ongoing user feedback. Such as multiple testing phases to ensure that the system meets the important functional and non-functional requirements.

Moreover, this project offers a promising path toward building a good healthcare system in undeserved communities. Future expansions may integrate advanced diagnostics, voice-based features, and even AI integration, further enhancing the platform's effectiveness.

CONTENTS

LIST OF FIGURES

Chapter 1

Case Development

This case study models a web app designed to connect patients and healthcare professionals, where its main focus is resolving the challenge of limited access to healthcare in parts of the developing world.

1.1 Patient and Healthcare Professional Registration

Events should start by the patient (1st actor) registering into the system using their name, location and phone number. The System (2nd actor) verifies and stores the patient's information. The System Administrator (3rd actor) reviews and approves registration. The fourth actor (Healthcare professional) 's part is to register their information, name, phone number, what field they are specialized in and the location of their clinic or workplace. Professionals are provided with an option to indicate their work hours or if they are available or not. The System processes and stores their information, and the Administrator approves or rejects the registration. The patient can then send an SMS to a central application number, containing their symptoms. The message should follow a specific layout and include keywords, for example the name of the body part that is in need of medical attention, in order to ensure the connection with the correct healthcare professional.

1.2 System Connection and Prescription

The system connects the patients to professionals depending on the specialization of the professional and the patient's symptoms. Each inquiry by a patient that is filtered and forwarded to a professional should include the patient's details and case. Professionals can then review the patient's case and reply using SMS service through the web app, while all of the messages are anonymized for data privacy of both the patient and healthcare professional. The healthcare professional can then send a prescription of the needed medicine through the system. The patient receives all the medicine details via an SMS.

1.3 Pharmacist, Insurance, IoT, and Maintenance

A Pharmacist (5th actor) now registers and goes through the same registration process as the patients and healthcare professionals before. The pharmacist approves the medicine and dispatches it to the patient. The patient submits insurance details through the system, then the system sends the treatment plan to the insurance company (6th actor) for approval. The insurance company can now decide whether to approve or deny the coverage. The patients are monitored using IoT devices such as glucose monitors, heart rate trackers etc. that send logs daily, or a certain set period of time, back to the professional and can send alerts if there are abnormal signs and readings, so the healthcare professional can review and reply with instructions. In order to establish health trends, the system sends anonymized data to the Health Ministry (7th actor) where it issues public health alerts or interventions as needed. The application should be able to handle a large amount of SMS traffic so redundancy should be considered in case something fails. The IT Team (8th actor) performs regular system updates and bug fixes reported by the system administrator. The IT Team then resolves any reported technical problems.

1.4 System Events

The events that go on are:

1. Patient Registration (actors: Patient, System Administrator):
    - Patient submits their personal details into the web app.
    - System verifies and stores patient information.
    - System administrator reviews and approves registration.
2. Healthcare Professional Registration (actors: Healthcare Professional, System Administrator):
    - Healthcare professional submits credentials and contact details.
    - System verifies the professional's information.
    - System administrator approves or rejects the application.
3. Symptom Reporting via SMS (actors: Patient, System):
    - Patient sends an SMS with symptom details.
    - System receives and stores the information.
    - System then forwards the message to the correct healthcare professional.
4. Diagnosis and medical advice (actors: Patient, Healthcare Professional):
    - The healthcare professional reviews the symptoms.
    - The professional then sends the diagnosis and treatment plan via the system.
    - The patient receives the response through SMS.
5. Prescription Management (actors: health care professional, patient, pharmacist):

- Professional prescribes medication through the system.
- Patient receives the prescription via SMS.
- Pharmacist verifies the prescription and dispatches the medicine.

6. Insurance Verification (actors: Patient, Insurance Provider):
   - Patient submits insurance details to the system.
   - System contacts insurance provider with the treatment plan for approval.
   - Insurance provider verifies and approves or denies coverage.
7. Patient Monitoring (actors: Healthcare professional, Patient, System):
   - Patient's connected devices send data about the patient to the system.
   - System alerts healthcare professional about abnormal readings.
   - Professional reviews the data and sends recommendations and instructions to patient.
8. Public Health Reporting (actors: Government/Health Ministry, System):
   - System aggregates anonymized patient data.
   - Government receives reports on health trends.
   - Health ministry issues public health alerts or interventions as needed.
9. System Maintenance (actors: IT Team, System Admin):
   - IT Team performs regular system updates and bug fixes.
   - System administrator monitors for anomalies and reports issues.
   - IT Team resolves any reported technical problems.

Chapter 2

Software Process Model

This project should be done using the Agile Methodology because it provides flexibility and offers the developers the ability to deliver working features in an incremental way. And since the client is not very rigid on implementation details and open to innovation, the agile method's iterative approach suits this project very well. Phases of the agile process:

1. Concept Phase: In this phase, we should determine the vision of the project, what we want it to do and the objectives behind it.
    - Requirement Gathering: Define the core requirements of the project, such as: SMS communication between patient and healthcare professional, patient and healthcare professional registration, approval of registration, remote monitoring via connected devices.
    - Set a distinction between functional requirements (needed features to function) and non-functional requirements.
2. Planning Sprints: In this phase, we set sprint length (2 weeks) and what to do in each sprint. We prioritize core requirements in the first sprints. Taking scenarios as examples and converting them into tasks, allows for a more structured approach on which requirement is more important.
    - 1st Sprint: Setting up the SMS messaging system.
    - 2nd Sprint: Registration System for patients, healthcare professionals and pharmacists.
    - 3rd Sprint: Setting up communication mean with the patient's insurance company.
    - 4th Sprint: Integrating medical devices (heart rate monitor, glucose monitor etc.) to monitor patients and sending medical data to the system.
    - 5th Sprint: Establishing a connection with the health ministry in order to send anonymized patient data to set health trends.
    - 6th Sprint: Functioning web application to provide healthcare for developing countries.
3. Architecture design and Prototyping: This phase focuses on building a good architecture and appealing UI through prototypes.
    - Create registration forms for people registering, and an admin panel for the sys admin.

- Design an appealing and easy to work with SMS system.
- Design a communication system that connects the system with insurance companies.
- System Architecture: Split it into Front-End and Back-End:
  - Front-End: Web App user interface, easy to navigate and use.
  - Back-End: Data base to store user data, SMS routing.
- Integration: APIs to manage medical devices logs and SMS information.
- Prototyping: Develop a functioning prototype of what the application would look like and interact with the user, to test out the rigidity of the features provided and see where problems reside.

4. Development Stage:
   - This is where the application is built as a functioning system where the core functionality is implemented incrementally using sprints. After every sprint, the system is tested to ensure good functionality and reliability.
5. Testing: In this phase, the system is tested in 4 stages:
   - Unit Testing: Test each component on its own (e.g message routing).
   - Integration Testing: Making sure that all components work together with no issues.
   - User Testing: Testing the system as a whole with real life users (doctors, patients . . .).
   - Testing under load: Stress testing the system to ensure it can handle heavy traffic.
6. Deployment and feedback:
   - Deploy the system in a small region or a small group of people for them to use the system as it should function
   - Collect feedback after a certain amount of usage time, on usability and performance. Use this feedback to fill gaps and add features requested by users to facilitate usage.
7. Maintenance:
   - After all is done, the IT team should be ready to maintain the app and keep it up to date based on user reviews. The project should be scalable in order to ensure covering more areas where professional healthcare is needed, with no major problems.

Chapter 3

Requirements Specifications

This chapter defines the requirements specifications for connecting patients to doctors system. Based on the case in Chapter 1 and the Agile process model in Chapter 2, these specifications help us understand what the system should do (functional requirements) and the qualities it must have (non-functional requirements). User interactions are usually illustrated through use case scenarios.

3.1 Functional Requirements

Functional requirements help us understand the specific behaviors the system must support. Most notably, each requirement have to align with the events, actors, and workflows already outlined in Chapter 1.

(a) User Registration and Management
- The system should let actors (patient, healthcare professionals, pharmacists, insurance provider, system admin, and IT team) to create and manage an account.
- The system has to validate and store user's data, including contact information and symptoms, and specialization details.

(b) SMS-Based Symptom Reporting
- The system has to enable patients to send symptom reports via SMS.
- The system should pass these messages to healthcare professionals, and analyze and identify keywords.

(c) Diagnosis and Prescription Handling
- Healthcare professionals should be able to review patient messages and send back a diagnosis or prescription through the system.
- The system should notify patients via SMS and inform pharmacists of upcoming prescriptin requests.

(d) Pharmacists and Medication Dispensing
- Pharmacists should receive prescription details through the system for verification and dispatch.
- The system should confirm medication dispatch to patients and record relevant transaction data.

(e) Insurance verification

- The system should allow Patients to submit insurance details.
- The system should pass relevant treatment data to the insurance provier for approval or denial of coverage.

(f) Remote Patient Monitoring

- The system should accept health readings from IoT devices.
- The system should generate alerts for healthcare professionals when abnormal data is detected.

(g) Public Health Data Aggregation

- The system should ensure that no identifiable information is included in government-level data.

(h) System Administration and Maintenance

- The System administation should have privileges to review and approve user registrations andmonitor system activities.
- The IT team should perform updates, bug fixes, and system maintenance tasks to ensure smooth operation.

3.2 Non-Functional Requirements

Non-functional requirements are quality attributes the system have to meet. For example:

(a) Performance and Scalability

- The system should handle high volumes of SMS traffic without delay

(b) Security and Privacy

- All personal data should be encrypted.
- Only authorized actors can view or modify data.

(c) Reliability and Availability

- The system should maintain service during hardware or network failures (almost like a plan B).

(d) Usability

- The web application interface should be supporting users with varied techincal skills (making it easier for users to use and navigate.)
- SMS commands must be straightforward so patients can easily follow instructions.

(e) Maintainability

- Clear documentation should be maintained for the IT team to know how to troubleshoot and so on.

3.3 Use Case Scenarios

Use case showcases how each actor interacts with the system. Here are few examples from events dicusessed in Chapter 1:

(a) Use Case: Patient Registration
- Actors: patient, system admin, system
- Preconditions: Patient has internet access and the system is online.
- Main flow:
  i. Patient accesses the registration form
  ii. System prompts for name, location, and phone number.
  iii. Patient submits details.
  iv. System stores the data.
  v. System Admin reviews and approves the registration.
  vi. Patient receives confirmation

(b) Use Case: Report Symptoms via SMS
- Actors: Patient, Healthcare professional, system
- Preconditions: Patient is a registred user, system is online
- Main flow:
  – Patient sends an SMS with symptom details.
  – System passes the message to identify the condition/body part.
  – System forwards the SMS details to an appropriate Healthcare professional based on keywords.
  – Healthcare Professional reviews the symptoms and decides on next steps.

(c) Use Case: Prescription Management
- Actors: Healthcare professional, patient, pharmacist, system
- Preconditions: patient's symptoms have been reviewed and healthcare professional is registered.
- Main flow:
  – Healthcare Professional prescribes medication in the system.
  – System notifies Patient of the prescription via SMS.
  – System sends prescription data to the Pharmacist.
  – Pharmacist verifies the prescription and dispatches medication.
  – Sysstem confirms medication dispatch to Patient

3.4 Assumptions and Constraints

(a) Assumptions
- All Patients have access to a functioning mobile phone with SMS capabilities.
- Healthcare professionals and pharmacists have stable internet connection.

(b) Constraints
- System must operate in remote areas where there might not be internet connection.
- The budget might be high for patients

Chapter 4

Architectural Design

4.1 Context Diagram

4.2 Use Case Diagram

4.3 State Machine Diagram

4.4 Data Flow Diagram (DFD)

4.5 Sequence Diagram

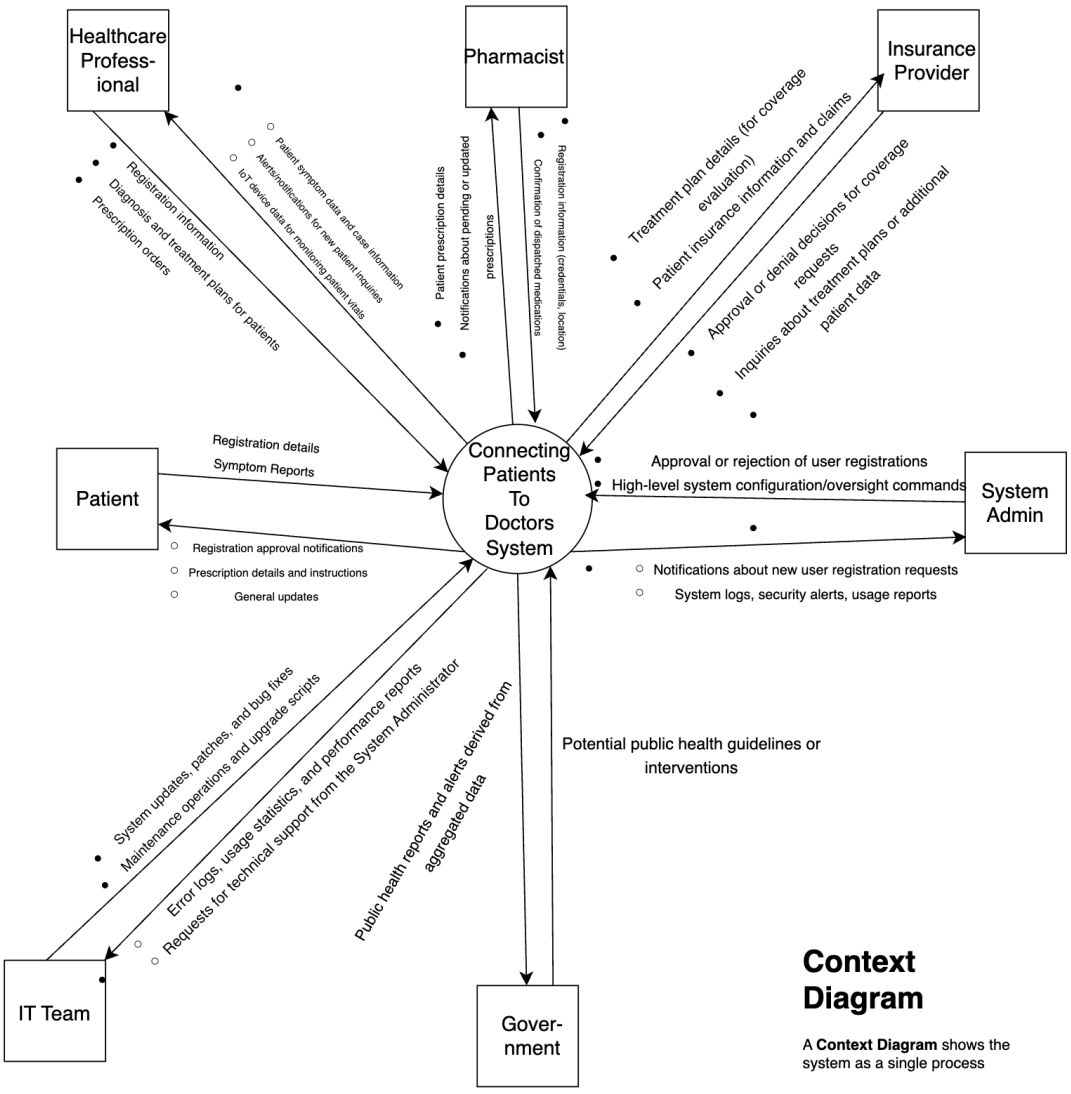

Figure 4.1: Context Diagram

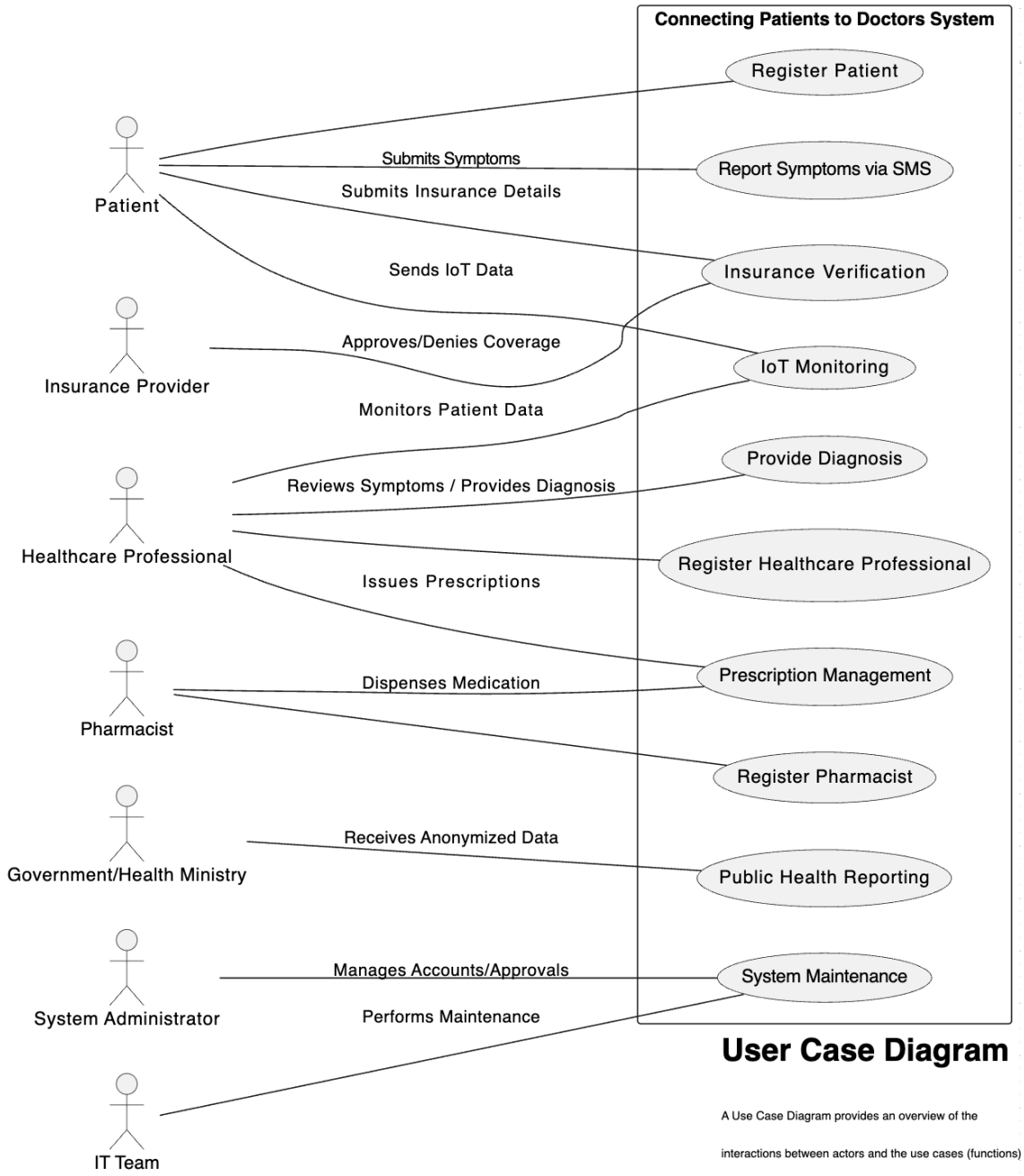

Figure 4.2: Use Case Diagram

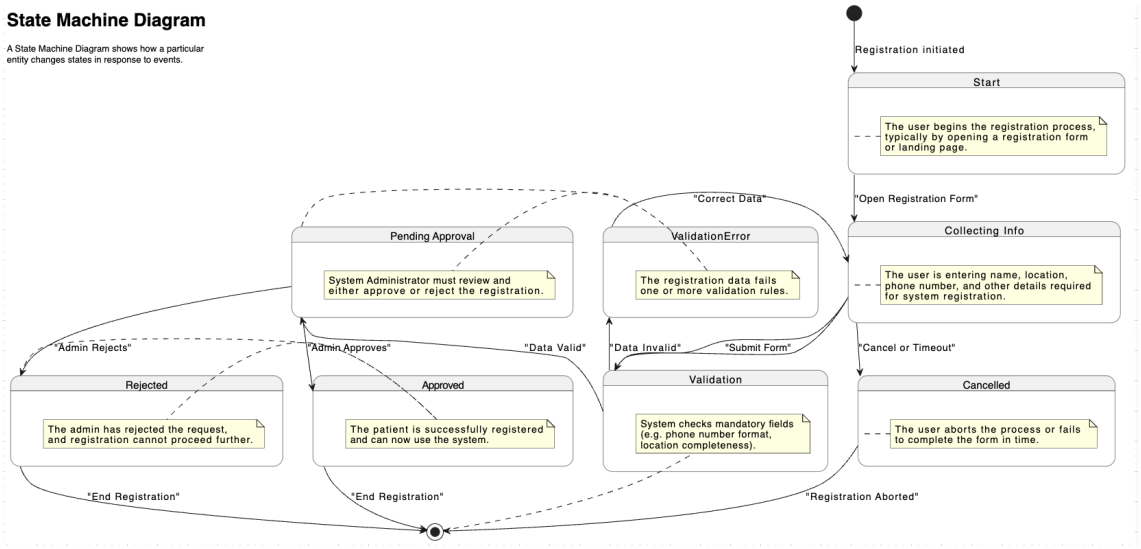

Figure 4.3: State Machine Diagram

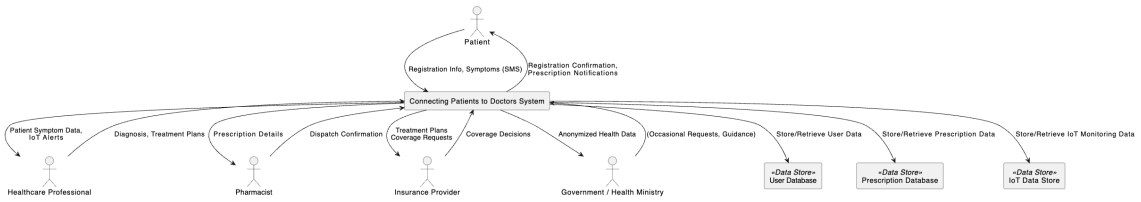

Figure 4.4: Data Flow Diagram (DFD)

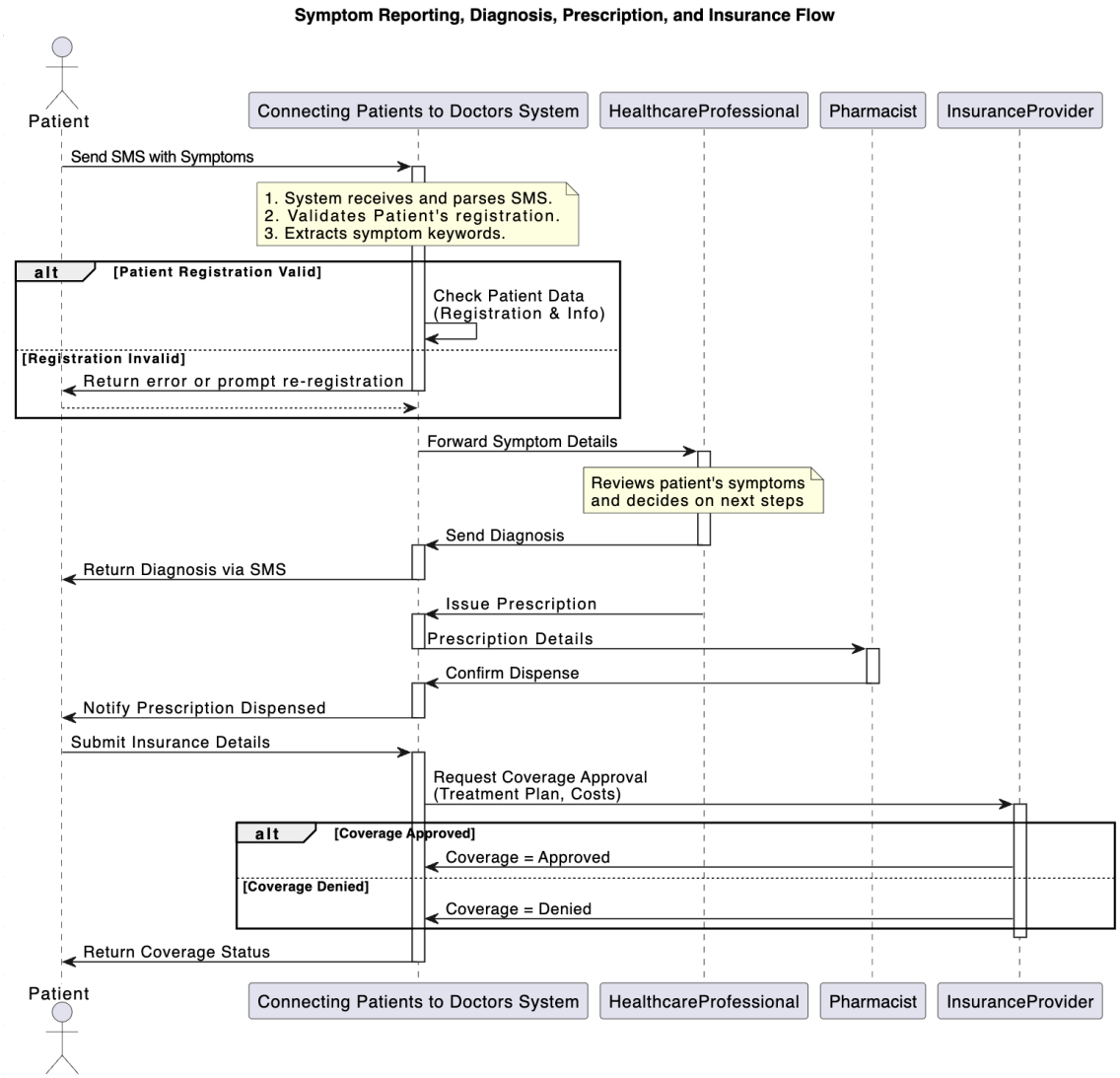

Figure 4.5: Sequence Diagram

Chapter 5

Software Testing

This chapter explains the approach, scope, and outcomes of testing a software to confirm that the system fulfills its requirements effectively.

5.1 Testing Approaches

Multiple approaches are used to verify correctness:

(a) Component-Level Checks (Unit Testing)
- Each module/fuction like SMS or user registration was checked to catch potential bugs early

(b) Inter-Module Evaluations (Integration Testing)
- The team tested the interactions between systems (like front end and back end) to ensure good data flow.

(c) Whole-System Analysis (End to End Testing)
- Conducted in realistic scenarios, it usually covers all user types (patients, insurance...) operating accordingly.

(d) Practical Trials (User Acceptance Testing)
- Healthcare providers and patients offer feedback to identify potential gaps in their experiences using the system.

5.2 Core Test Scenarios

These scenarios represent the most critical operations within the system

(a) SMS Message Relay
- Aim:
  - Confirm accurate delivery of the symptoms submitted by patients to the right healthcare professional
- Procedure:
  - Patient texts symptoms via SMS.

  - System forwards the details to the correct professional.
  - Professional responds with advice or follow-up.
- Expected Result:
  - Message arrive within seconds.

(b) New User Onboarding
- Aim:
  - Ensure the system correctly registers patients and healthcare professionals.
- Procedure:
  - User enters personal information.
  - System validates and stores data.
  - Admin approves the account, notifying the user.
- Expected Result:
  - Valid accounts are activated with minimal delays.

(c) Prescription Workflow
- Aim:
  - Verify that professionals can issue prescriptions seamlessly.
- Procedure:
  - Professional writes a prescripton in the system.
  - Pharmacist receives and confirms medication dispatch.
  - Patient is notified of prescription readiness via SMS.
- Expected Result:
  - Accurate prescription records with minimal processing time.

(d) Remote Device Monitoring
- Aim:
  - Check that patient health data is successfully transmitted and monitored.
- Procedure:
  - IoT device sends measurements to the system.
  - System updates the database and checks for anomalies.
  - Professional receives alerts for abnormal readings.
- Expected Result:
  - Timely data collection and notifications without delay.

5.3 Outcomes and Observations

Key results from the verifications include:

(a) High SMS Delivery Rates.

(b) Seamless Registration.

(c) Efficient Prescription Handling.

(d) Stable IoT Integration.

(e) User Feedback:
- Users confirmed that they are able to use the web application easily due to its friendly interface.

Chapter 6

Conclusion and Future Work

6.1 Summary of Achievements

The Connecting Patients to Doctors project is a good solution for limited healthcare accessibility in undeserved regions using SMS and a web-based platform. Due to its model type, which is incremental, it helped create a stable and resilient system that:

(a) Combines messaging capabilities with remote consultation.

(b) Supports diverse user roles, like patients, healthcare professionals and others.

(c) Integrates IoT devices for real-time health monitoring.

(d) Has proven usability under practical testing conditions.

By incorporating continuous improvements through each version, the final product will align well with the users' satisfaction.

6.2 Potential Enhancements (Future Work)

(a) Voice and video features.

(b) AI diagnostic tools.

(c) Local language customization.

(d) Expanded government collaboration:

- Partnering with public health agencies and NGOs.

(e) Pharmacy and insurance ecosystem:

- Collaborating with local pharmacies and insurance networks.

6.3 Conclusion

The system is a practical and adaptable solution for removing the gap in healthcare delivery. Its incremental model ensures it can grow alongside community needs, integrating new medical devices and communication methods. By engaging with users, the system will work well in digital healthcare.

Looking ahead, strategic enhancements like collaborating with local pharmacies or having partnerships with NGOs can help the system have a huge impact and reach in the future. With dedication and reliability, this project can evolve into a comprehensive ecosystem where healthcare is accessible to everyone.

Appendix A

Credits

1.1 Figures

- Elia Ghazal using Draw.io and plantUML.

1.2 Research

- Charbel Abou Akl - Chapter 1 and 2
- Elia Ghazal - Chapter 3, 4, 5, and 6
- George Khayat - PowerPoint

1.3 LaTeX Code

- Elia Ghazal

Annotated Bibliography

[1] I. Sommerville, *Software Engineering*. Pearson, 1988.

Pages 56-178

[2] V. P. Online, "Use case diagram tutorial."

Understand how Use Case Diagrams are made

[3] A. Roques, "plantuml," https://plantuml.com/.

Learn plantUML to code a diagram

[4] Hitchhiker, "Welcome to The Hitchhiker's Guide to PlantUML!" https://crashedmind.github.io/PlantUMLHitchhikersGuide/.

Tutorial to plantUML