Chapter 3

**SYSTEM DESIGN AND ANALYSIS**

**3.0 INTRODUCTION**

This chapter will outline the system analysis and design process for a healthcare management system using a general practitioner. It will cover the system requirements, architectural framework, and database design essential to support and implement the system.

**3.1 DESCRIPTION OF THE SYSTEM:**

1. The developed software system shall reduce part of the major issues at this point either in healthcare management while introducing superior efficiency, accuracy, and end-user experience in the chosen area. This system shall integrate the most vital functionalities via a robust interface during data processing and secure data management.

**3.2 DESIGN MODEL:**

A software development process model defines the steps and activities required for the design, development, and deployment of a software product. Several process models can be used for a Healthcare Management System using a General Practitioner but the most suitable is the Agile Process Model, it aids development for flexibility, regular testing, and user feedback in healthcare applications. User feedback from medical staff, patients, and administrators is necessary to ensure the system is user-friendly and meets practical needs. Agile emphasizes regular feedback, allowing adjustments based on real user inputs. Healthcare requirements and regulations can change and it has iterative cycles that allow us to adjust quickly to new needs or unexpected challenges without delaying the entire project. The agile process model allows for higher quality with regular testing and encourages continuous testing, crucial in healthcare systems where accuracy and reliability are paramount, continuous testing helps catch errors early, reducing the risk of critical issues at deployment. It lets us focus on the most critical features (for example patient records, and scheduling) first, delivering a minimum viable product (MVP) early. This staged approach ensures essential functions are deployed quickly and refined as we continue. There is enhanced collaboration and communication as healthcare projects often involve several stakeholders like doctors, administrators, and general practitioners. Agile encourages cross-functional collaboration, ensuring that all perspectives are incorporated throughout development.

**3.3 PROPOSED SYSTEM ARCHITECTURE:**

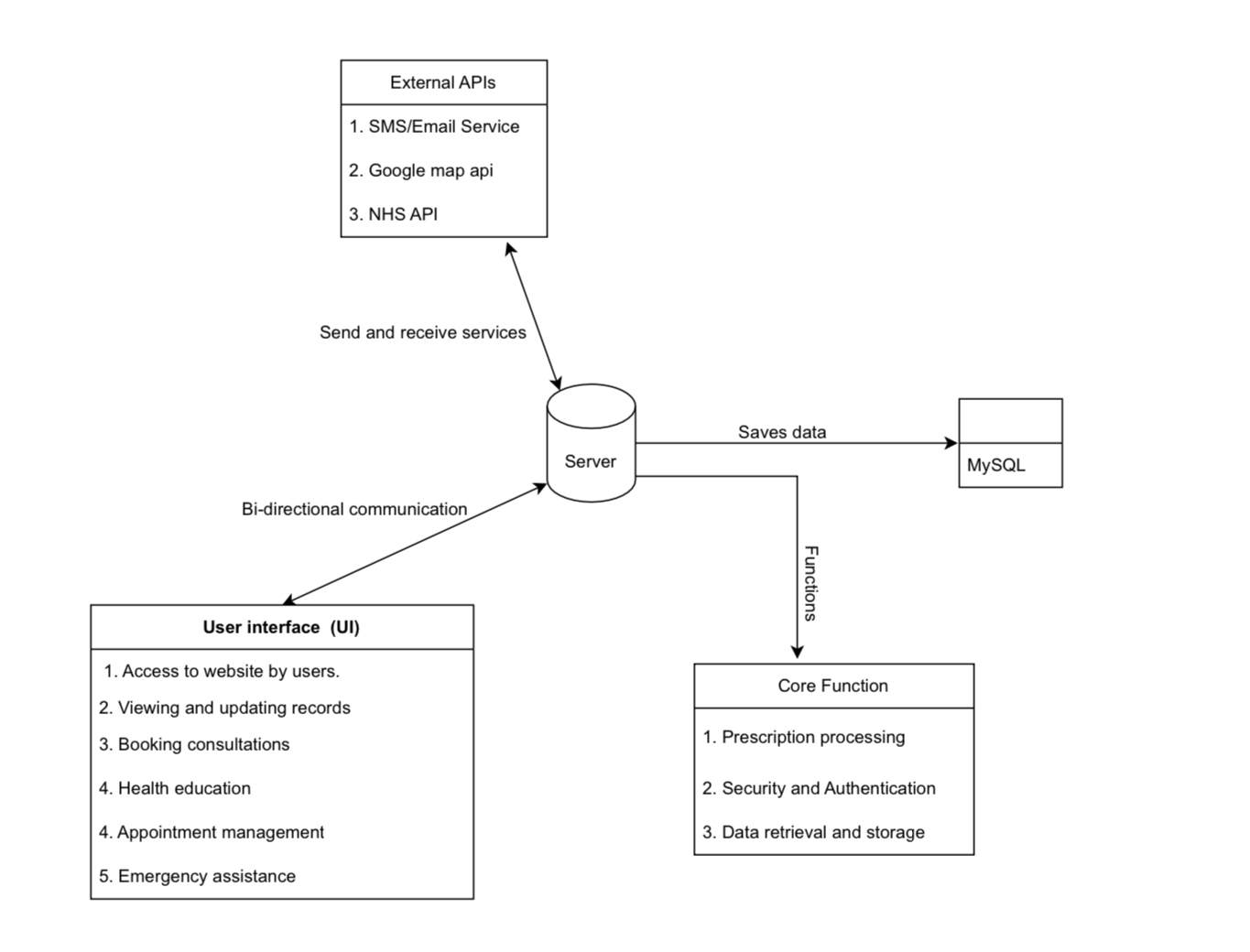
The high-level overview of the system architecture for the healthcare management system for General Practitioner practice would consist of key layers, each of them dealing with different aspects like patient care, data management, and system security, among others. It shall have on top the Client Layer serving the user interface as a web and mobile application to GPs, patients, and administrative staff who interact with it. This involves a General Practitioner Dashboard through which patient records are maintained and appointments booked. Then, there is a patient portal for booking appointments, viewing health records, and contacting the GP. The responsive design accesses both desktops and mobile devices for in-office and remote usage.

The Application Layer controls the business logic behind such core functions of the system. It features different modules: EHR management, appointment scheduling, billing, and insurance, and also includes functions of telemedicine. The EHR module securely stores and updates patient records if needed. The scheduling module in the software helps to manage appointment times, and it may even prompt reminders. The billing and insurance module automates the invoicing and also processes the claims. Further, the features for telemedicine allow consultations virtually. Another aspect of this layer is that there will also be a Role-Based Access Control system in place, where only specific users can access some functionality within the system or specific information concerning the patients, thus keeping personal data private.

Data Layer is used for storing and managing all the critical information like patient records, appointments, and billing in a much-secured way. The EHR data is stored in structured databases; audit trails are maintained concerning access and changes to the records. This forms the basis for various data privacy compliance requirements. It also allows backup and recovery processes to ensure data integrity and availability in case of any failure within the system. The API will provide secure gateways, thus allowing seamless data sharing between the GP's system and other healthcare providers, such as hospitals, labs, and pharmacies. A GP can send e-prescriptions to pharmacies through this layer electronically, request lab results, and even view patient data from external HIEs; therefore, enabling coordinated and efficient patient care.

The Security Layer recognizes that security is fundamental and ensures data related to the patients is kept confidential and that the system follows healthcare data regulations such as HIPAA. Such a layer utilizes encryption of data, multi-factor authentication, and access control. The set layer allows for continuous monitoring and auditing to identify unauthorized access or anomalies in data access, hence ensuring high security and confidentiality of patient data, it provides tools for report generation and the analysis of data trends in the Analytics and Reporting Layer for data insights. This includes monitoring patient outcomes, financial reporting for billing and revenue management, and operational efficiency analytics-monitoring appointment trends and resource utilization. These will help the GPs make informed decisions toward improving the lot of their patients and managing their practice.

An intuitive and user-friendly interface enables interaction between the general practitioner and the patient. This interface may include a graphical user interface (GUI) or a simple display screen where recognized text is presented.



**FIGURE 3.3 PROPOSED SYSTEM ARCHITECTURE**

**3.4 SYSTEM MODELING**

System modeling assists various modeling techniques, system modeling helps clarify requirements, improve design, and ensure that all components of the system work together efficiently. Here are some commonly used system modeling techniques for a healthcare management system in a General practitioner setting.

**3.4.1 USE CASE DIAGRAM**

The use case diagram outlines the key functionalities and user interactions within the system and identifies the main users (actors) of the system and the actions (use cases) they can perform.

1. The system will be able to access Patient Records
2. The system will be able to update Medical History
3. The system will be able to write Prescriptions
4. The system will be able to order Lab Tests.
5. The Patient will be able to book an Appointment
6. The Patient will be able to view medical History
7. The Patient will be able to request Prescription Refills.
8. The administrative staff will be able to manage appointments
9. The administrative staff will be able to process billings and insurance claims.

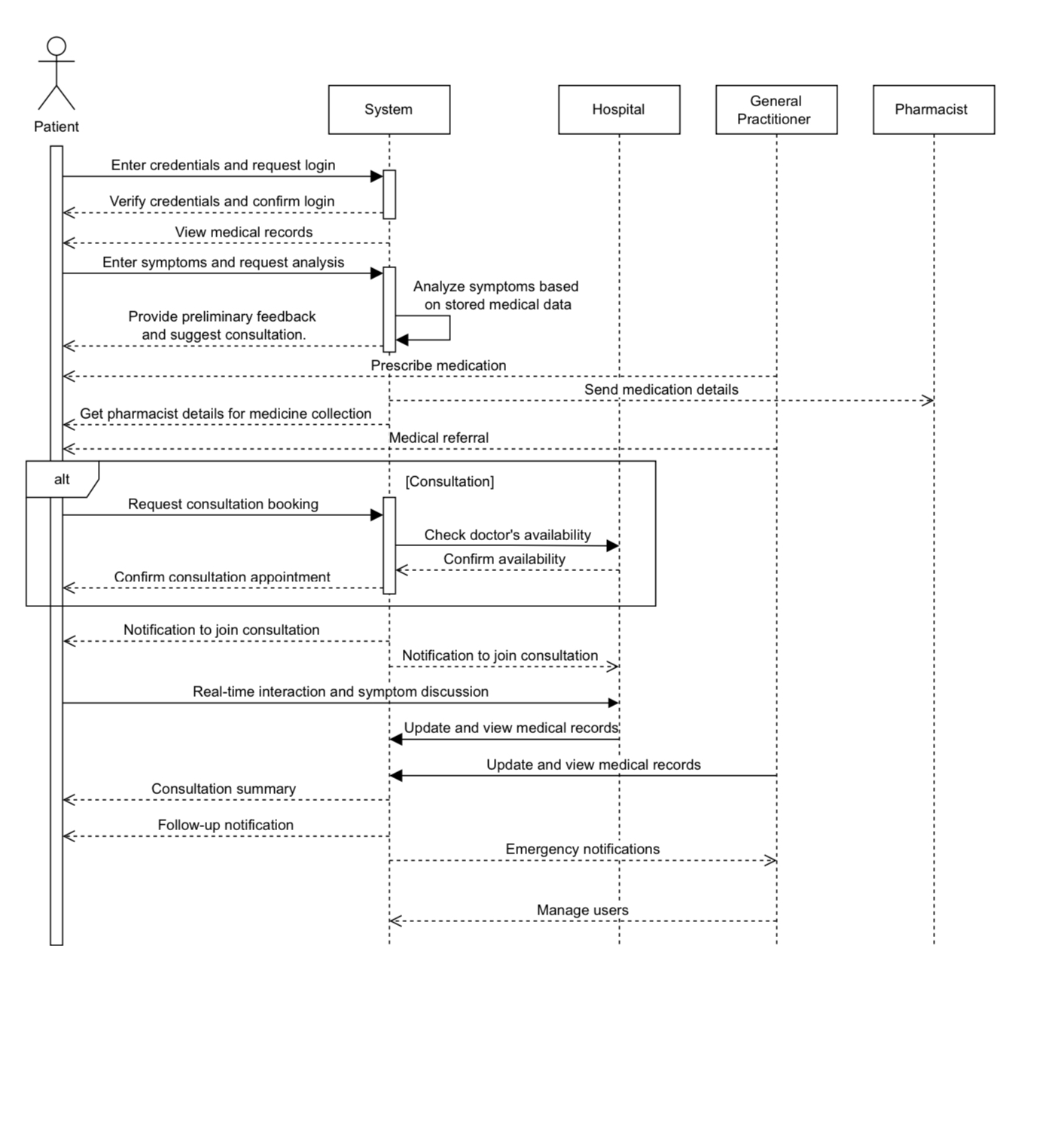


FIGURE 3.4 USE CASE DIAGRAM SHOWING INTERACTION WITH SYSTEM

**3.4.2 SEQUENCE DIAGRAM**

This is used to model the sequence of interactions between system components for specific processes, it visualizes how objects interact in a specific scenario by outlining the sequence of messages exchanged

1. The GP logs into the Healthcare Management System.
2. The Healthcare Management System authenticates the GP’s credentials.
3. Once authenticated, the GP requests access to a Patient’s Medical Record.
4. The Healthcare Management System sends the request to the Database to retrieve the patient’s record.
5. The Database responds with the requested medical record data.
6. The Healthcare Management System displays the medical record to the GP.
7. The patient initiates a “Book Appointment” request through the Patient Portal.
8. The Patient Portal sends the request to the Appointment Scheduling System.
9. The Appointment Scheduling System checks the GP’s Schedule for availability.
10. The GP’s Schedule responds with available slots.
11. The Appointment Scheduling System confirms the appointment by saving it to the Database.
12. The Patient Portal notifies the patient of the confirmed appointment.



**3.5 TECHNOLOGIES AND TOOLS**

Some of the tools and technologies that would be used to develop the proposed system include the following:

i. ASP.NET MVC: A framework within ASP.NET Core that supports the Model-View-Controller pattern, ideal for organizing the codebase and managing the application’s logic and UI separately. This is fundamental for creating web applications with clean architecture and manageable code.

ii. C#: The primary programming language for ASP.NET applications. Its robust features make it suitable for handling backend logic, data processing, and application workflows.

iii. Entity Framework Core: An ORM for ASP.NET that allows you to interact with databases using C# instead of SQL. It simplifies database management and is beginner-friendly for performing CRUD (Create, Read, Update, Delete) operations.

iv. SQL Server: A relational database that integrates well with ASP.NET applications. SQL Server provides a reliable way to store and retrieve patient records, user data, and other critical healthcare information.

v. Bootstrap: A front-end framework for creating responsive and mobile-friendly designs. Bootstrap provides pre-built components and styles, making it easier to develop a visually appealing user interface.

vi. GitHub: A version control and collaboration platform that allows you to track changes, manage code versions, and collaborate with other developers. GitHub is essential for managing your project’s codebase and supports CI/CD integrations.

vii. Identity and Authentication: ASP.NET Core Identity can handle user authentication and authorization out of the box, supporting features like login, registration, and role-based access control, which are critical for protecting healthcare data.

viii. Swagger (OpenAPI): A tool for documenting and testing your ASP.NET MVC APIs, making it easier to understand and debug API endpoints. Swagger is essential for providing a clear interface for developers interacting with your backend.

ix. Data Encryption Libraries (ASP.NET Core Data Protection): Security libraries ensure all sensitive user data is encrypted, aligning with healthcare privacy standards like GDPR or HIPAA.

x. Twilio/SendGrid: Third-party services for sending SMS and email notifications, useful for appointment reminders, prescription alerts, and emergency communications.

xi. SignalR: A real-time communication library in ASP.NET Core, ideal for enabling live consulting or chat functionality between users and healthcare providers.

**3.6 REQUIREMENTS ANALYSIS**

The requirement analysis phase is an integral part of the development of the application. It is the criteria that must be met to satisfy the needs and expectations of the stakeholders. This stage analyzes the various functionalities that are used to achieve the project goals and objectives. Requirements are divided into user and system requirements. User requirements describe the needs, expectations, and constraints of end-users in terms of the system's functionality and usability. System requirements describe the overall characteristics and features of the system. The system requirements are split into functional and non-functional requirements. Functional requirements define what a product must do and what its features and functions are. Nonfunctional requirements describe the general properties of a system.

**USER REQUIREMENTS**

Here are some of the user requirements of the proposed system:

1. Users shall be able to register on the application.

2. Users shall be able to enter personal information

3. Users shall update personal information when necessary.

4. Users shall be able to get feedback on symptoms.

5. Users shall be able to get a referral to a hospital.

6. Users shall be able to get a referral to a pharmacist.

7. Users shall be able to get real-time consulting from a doctor.

8. Users shall be able to get emergency services when necessary.

9. Users shall get reminders for appointments, prescription refills, and scheduled follow-up consultations.

10. Users shall be able to view and download medical history records.

**3.6.2 FUNCTIONAL REQUIREMENTS**

The functional requirements of the system are:

1. The system shall be able to perform multi-factor authentication.

2. The system shall be able to store the user’s medical records.

3. The system shall provide patient data encryption for user data safety.

4. The system shall provide a user audit trail.

5. The system shall set appointment bookings.

6. The system shall be able to set notifications for appointments, prescription readiness, and test results.

7. The system shall provide location-based referrals.

8. The system shall show possible health challenges.

9. The system shall send emergency information to the hospital.

10. The system shall provide comprehensive information on a user.

11. The system shall be able to provide a system checker.

12. The system shall be able to provide referral pharmacists.

13. The system shall be able to provide referral hospitals.

14. The system shall provide video consulting services.

15. The system shall be able to integrate with the external health care system.

**NON-FUNCTIONAL REQUIREMENTS**

Here are the non-functional requirements of the system:

1. The system should respond to the user's input within 2 seconds.

2. The mean time between failures (MTBF) should be at least 1000 hours.

3. The system should be available for 20 hours a day, with scheduled maintenance communicated in advance.

4. The system should maintain acceptable response times with 1000 concurrent users accessing the map.

5. The system should meet accessibility standards (such as WCAG) to ensure usability for people with disabilities.

6. The system shall ensure the privacy and confidentiality of user data, adhering to relevant data protection regulations.

7. It shall implement secure communication protocols to prevent unauthorized access or interception of sensitive information.

8. The system should be compatible with the latest versions of major web browsers.

9. The system should have regular system backup for disaster recovery.

10. The system should implement redundancy and failover mechanisms to ensure continuous service in case of server or component failures.