Software Requirements Specification

for

Clinic Appointment and Treatment Management System

Version 1.0

Prepared by Group 2

230365D - LAKSHAN H.M.K.
230425M - NAWAGAMUWA N.A.K.
230453V - PADMASIRI G.R.H.D.
230518C - RAKESHAN K.R.K.
230611F - SHAZAN M.S.M.

CS3043 - Database Systems

2025/08/06

Table of Contents

Table of Contents ii
Revision History iii
1. Introduction 1
1.1 Purpose 1
1.2 Document Conventions 1
1.3 Intended Audience and Reading Suggestions 1
1.4 Product Scope 2
1.5 References 2
2. Overall Description 2
2.1 Product Perspective 2
2.2 Product Functions 3
2.3 User Classes and Characteristics 4
2.4 Operating Environment 8
2.5 Design and Implementation Constraints 8
2.6 User Documentation 8
2.7 Assumptions and Dependencies 8
3. External Interface Requirements 9
3.1 User Interfaces 9
3.2 Hardware Interfaces 12
3.3 Software Interfaces 13
3.4 Communications Interfaces 15
4. System Features 16
4.1 System Feature 1 16
4.2 System Feature 2 17
5. Other Nonfunctional Requirements 19
5.1 Performance Requirements 19
5.2 Safety Requirements 19
5.3 Security Requirements 20
5.4 Software Quality Attributes 20
5.5 Business Rules 21
6. Other Requirements 21
Appendix A: Glossary 21

Appendix B: Analysis Models ……………………………………………………………23
Appendix C: To Be Determined List ……………………………………………………………25
Appendix D: Assumptions ……………………………………………………………..25
Appendix E: Technical Considerations ……………………………………………………..25

Revision History

Name	Date	Reason For Changes	Version

1. Introduction

1.1 Purpose

The aim of this Software Requirements Specification (SRS) document is to outline both the functional and non-functional requirements for the "Clinic Appointment and Treatment Management System" (CATMS), which will be developed for MedSync, a medium-sized multi-specialty clinic with locations in Colombo, Kandy, and Galle. The goal of creating this system is to automate the management of appointments, treatments, and billing procedures, replacing the existing manual methods that rely on paper records and spreadsheets.

1.2 Document Conventions

This SRS document follows the standard formatting types to guarantee the consistency and clarity of the documentation

- Entities and Tables in CATMS are written using camelCase - (e.g., paymentDetails, patientRecords)
- Requirements identifiers follow the below format:
  Functional requirements - "fr-1", "fr-2", etc.
  Non-Functional requirements - "nfr-1", "nfr-2", etc.
- Function names are displayed in monospaced font - (e.g., calculateBill(), scheduleAppointment()).

1.3 Intended Audience and Reading Suggestions

This SRS document is intended for all individuals involved in the development of the "Clinic Appointment and Treatment Management System (CATMS)". The following groups are expected to use this document:

- System analysts - check the functional and data requirements of the system.
- Database developers - to implement the database structure as specified.
- Quality assurance (QA) - to prepare test plans for the system and make sure the system meets its requirements.
- Project manager - to monitor the development progress and ensure it aligns with the business objects.
- Clinic stakeholders - to review whether the proposed system address current operational needs.

Readers are advised to begin with the Introduction and Data Convention sections to obtain a comprehensive understanding of the system. Those with technical expertise (including developers and testers) should focus particularly on the External Interface Requirements, System Features, Other Non-functional Requirements, and Other requirements sections.
Individuals without technical expertise can concentrate on the general explanations to grasp how the system aids clinic functions.

1.4 Product Scope

The Clinic Appointment and Treatment Management System (CATMS) is being designed for MedSync to replace their existing methods that rely on manual processes and spreadsheets. The system will oversee essential clinic functions including patient registration, scheduling appointments, documenting treatments, and managing billing in a centralized and effective way.

It will facilitate access to patient records across multiple branches, ensure that doctor appointments are not scheduled simultaneously, and enable doctors to record their consultation notes and prescribed treatments. The system will generate invoices automatically based on the treatments completed and will accommodate both full and partial payments while monitoring any outstanding balances.

Additional functionalities will include managing insurance claims for specific treatments, addressing emergency walk-ins, and allowing appointment rescheduling. The system will also provide crucial reports for clinic management, like revenue summaries and appointment analytics.

1.5 References

- IEEE Recommended Practice for Software Requirements Specifications (IEEE Std 830 / ISO/IEC/IEEE 29148) - https://ieeexplore.ieee.org/document/720574
- Standard SQL and DBMS Guidelines - https://mrcet.com/downloads/digital_notes/ECE/III%20Year/DATABASE%20MANAGEMENT%20SYSTEMS.pdf, https://www.geeksforgeeks.org/dbms/dbms/
- User interface Design Guidelines: 10 Rules of Thumb - https://www.interaction-design.org/literature/article/user-interface-design-guidelines-10-rules-of-thumb
- Software requirements specification - https://en.wikipedia.org/wiki/Software_requirements_specification
- What is Database? - https://www.geeksforgeeks.org/dbms/what-is-database/
- What is Healthcare Technology - https://www.ibm.com/think/topics/healthcare-technology

2. Overall Description

2.1 Product Perspective

The Clinic Appointment and Treatment Management system (CATMS) is a new self-contained software product being designed for a medium scale multi-specialty clinic called MedSync which has branches in Colombo, Kandy and Galle. This system is designed to increase the efficiency and productivity of the prevailing paper and Excel based clinic management system. Therefore this is neither a follow-on member of a product family nor an extension of a certain existing system.

The CATMS will digitize the whole system integrating core operations such as appointment booking, treatment data recording and billing process. The new system will act as the key component of the MedSync and it would interact with its human users through an UI.

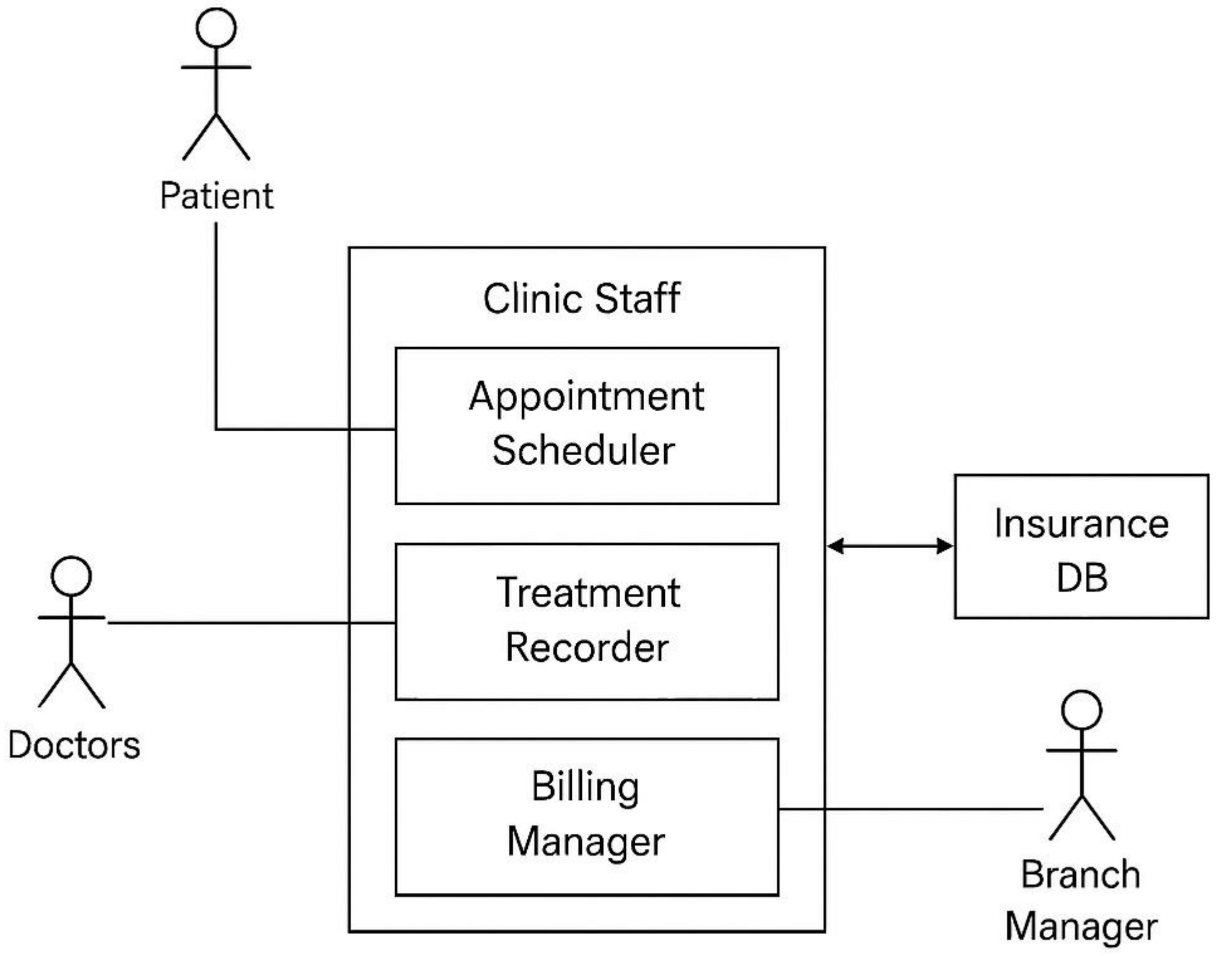


Figure 1 - System Environment

2.2 Product Functions

The top level picture includes many key functionalities that will be supported by CATMS.
The summary of the key functionalities are:

- Make the appointment booking process much more easier through online booking facilities.
- Manage doctor schedules and allow appointments to be booked or rescheduled accordingly.
- Prevent overlapping of appointments for doctors.
- Allow for emergency walk-ins and update appointment schedules accordingly.

- Record treatments and consultation notes securely after appointment completion and making it available to access through any branch.
- Maintain a service catalog including all pricing details and services provided.
- Generate invoices for treatments, consultations and other services provided.
- Manage full and partial payments.
- Track outstanding patient balances.
- Support insurance claims for certain treatments and reimburse fully or partially based on policy terms.
- Generate operational reports including daily branch-wise appointment summary, doctor-wise revenue tracking, patient dues list, treatment frequency by category, insurance vs. out-of-pocket expense summary.

2.3 User Classes and Characteristics

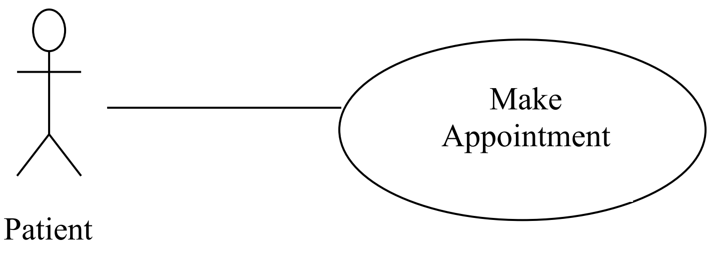
User Class members	User Class Characteristics
Doctors	● View and update appointments ● View and update doctor's profile ● Add treatments and notes ● View own earnings report
Patients	● View the reports ● View the pending bills
High technical expertise	● View and update appointments ● Managing database
Receptionists/Frequent Users of staff, Nurses	● Rescheduling ● Managing Walk – ins ● Processing payments
Branch Managers	● Administrative access ● Monitor branch operations ● Generate reports

This section outlines the use cases for each of the class members separately.

- **Patient Use Case**

Use case: ***Making an Appointment***

Diagram:



Brief Description
The Patient accesses the Channeling Website, searches for a doctor and makes an appointment in a preferred time slot.

Initial Step-By-Step Description
Before this use case can be initiated, the Patient has already accessed the Online Journal Website.

1. The Reader chooses to search by doctor name, speciality, and branch.
2. The system displays the choices to the Patient .
3. The Patient selects the available time slot.
4. The system presents the receipt of the appointment to the patient.
5. The patient chooses to download the receipt.

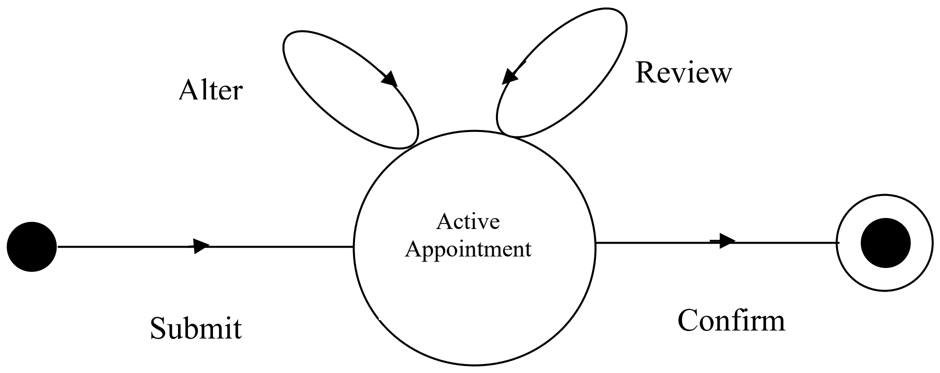
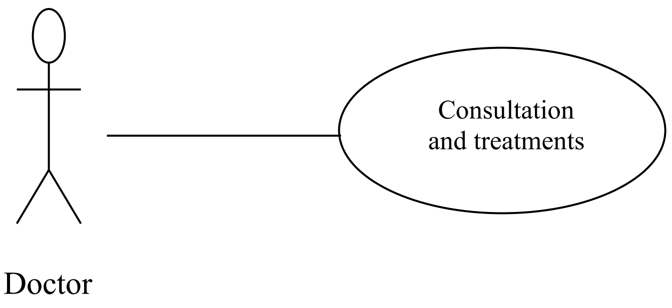


Figure 1 – Appointment making Process

The Appointment making Process state-transition diagram summarizes the use cases listed below. A patient make an appointment. The system automatically enters it into the system and assigns it to staff. Next, staff confirm the appointment according to the available details of the doctor availability.

- **Doctor Use Case**

Use case: Consultation and treatments
Diagram:



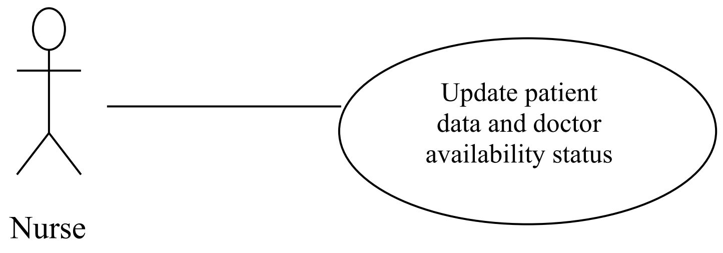
Brief Description
The doctor gives consultations and provides treatments

Initial Step-By-Step Description
Before this use case can be initiated, the Doctor has already connected to the Website.

1. The Doctor chooses the required patient.
2. The System uses a HTML tag to bring up the user's profile.
3. The Doctor fills in the consultations and treatments and attaches the medical files and confirms them.

- **Update Information use cases**

Use case: Update patient data and doctor availability status
Diagram:



Brief Description
The Nurse enters a new patient or updates information about a current patient. Nurse can also edit information about doctor availability.

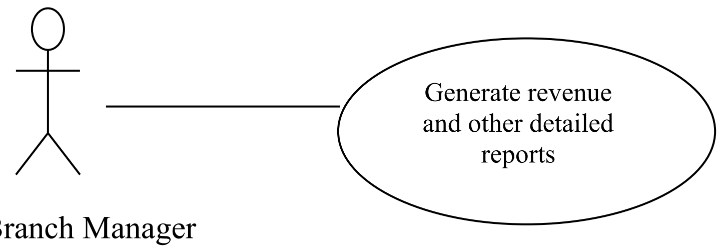
Initial Step-By-Step Description
Before this use case can be initiated, the Nurse has already accessed the profile page of the patient.

1. The nurse selects to *Add/Update data behalf of patient*.
2. The system presents a choice of adding or updating.
3. The nurse chooses to add or to update.
4. Do the correct and required action

- **Report generation use cases**

Use case: Generate reports as required by the Branch Manager

Diagram:



Brief Description

The Brach Manager generates the required revenue and other detailed reports to manage the system easily.

Initial Step-By-Step Description

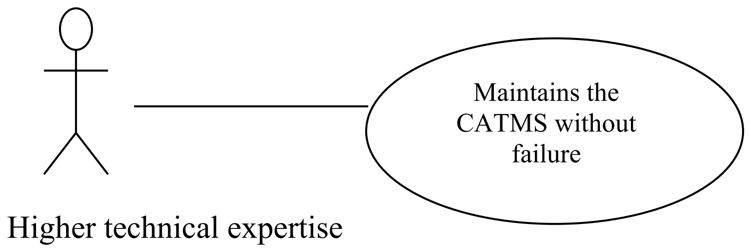
Before this use case can be initiated, the Branch Manager has already accessed the database through the system.

1. The Branch Manager selects to *generate the required reports.*
2. The system presents a choice of the report type.
3. Send the system report to the higher level.

- **Technical maintenance use cases**

Use case: Maintain the CATMS

Diagram:


Maintains the CATMS without failure
Higher technical expertise

Brief Description

The Higher Technical Expertise maintains the whole CATMS without a failure and with increased efficiency.

Initial Step-By-Step Description

Before this use case can be initiated, the Higher Technical Expertise has already accessed the whole system as an administrator.

1. Do tests for possible risks
2. Maintain system integrity

3. Controls the whole CATMS.

2.4 Operating Environment

Backend	SQL, Node.js based system
Database	MySQL
Frontend	HTML, CSS, JavaScript
OS Compatibility	Windows 7 or more recent versions of windows
Hardware	Standard desktop systems

2.5 Design and Implementation Constraints

Logical constraints

- Use Dummy data for inserting via SQL scripts
- Focus must be on Atomicity, Consistency, Isolation and Durability(ACID)
- Data integrity should be maintained
- System should handle rescheduling and walk-in logic
- It is Important to track partial payments and outstanding balances accurately
- CATMS should be able to support concurrent access
- Standard security practices for sensitive health and financial data
- Should support insurance logic
- CATMS should be able to scale up if needed

Technical Constraints

- At least 8GB RAM
- MySQL database
- At least i5 processor for Intel Users
- Knowledge about HTML, CSS, JavaScript, SQL, Bootstrap, Node.js

2.6 User Documentation

- Setup and configuration guide
- Sample SQL query guide
- ER diagrams and data flow diagrams
- Tutorials for testing
- Online help

2.7 Assumptions and Dependencies

- Details of all patients is assumed to be consistent across branches
- The same patient ID system is used across branches
- Treatment prices are same across all branches

- Insurance terms will be assumed where not specified
- External insurance claim processing is out of scope; Only internal tracking is included
- Logic of billing assumes payments are made either fully or partially per invoice
- Any integrations related to national health systems and third party APIs is out of scope

3. External Interface Requirements

3.1 User Interfaces

- **Overview**: The CATMS will provide a web-based graphical user interface (GUI) accessible to patients, clinic staff (receptionists, doctors, billing clerks) and QA testers.

Appointment Booking Interface

- Allow patients or staff to create new appointments with doctors for specific time slots.
- Display available time slots for each doctor at a given branch.
- Support rescheduling and cancellation of appointments.
- Allow creation of emergency walk-in appointments directly by staff.

Patient Registration Interface

- Enable registration of new patients at any branch.
- Input patient personal details, emergency contact information, and health insurance data.
- Support access to patient records from any branch.

Doctor Schedule Interface

- Display doctors' schedules including all booked appointments and availability.
- Prevent overlapping appointments for the same doctor.

Treatment Recording Interface

- Allow doctors to record treatments and consultation notes after appointment completion.
- Enable selection of treatments from a predefined treatment catalog.

Billing Interface

- Generate invoices for completed treatments and consultations.
- Allow partial or full payments against invoices.
- Track outstanding dues per patient.
- Handle insurance claims for eligible treatments, reflecting reimbursements according to policy terms.

Reports Interface

- Generate branch-wise daily appointment summaries (scheduled, completed, cancelled).
- Generate doctor-wise revenue reports.
- List patients with outstanding balances.
- Show treatment counts per category over a specified time period.
- Compare insurance coverage vs. out-of-pocket payments.

- **Logical Characteristics**:
  - **Screen Layout**:
    - Consistent header with clinic logo, navigation menu, and user profile info.

    - Main content area divided into logical sections as Appointment Booking, Patient Records, Billing.
    - Use of tabular views, forms, and modal dialogs for data entry and display.
- **Buttons/Functions**:
    - Common buttons like Save, Cancel, Edit, Delete, Search on all forms.
    - Help button linking to a user manual or quick tips popup on every page.
    - Confirmation dialogs before data-altering operations (e.g., cancel appointment).
- **Keyboard Shortcuts**:
    - Ctrl+S for Save, Ctrl+F for Search, Esc to close dialogs.
    - Tab key navigation through form fields.
- **Error Handling and Messages**:
    - Inline validation messages for form inputs.
    - Standardized error dialogs for system failures or constraint violations.
    - Clear and user-friendly error text (e.g., “Appointment conflicts with an existing booking”).
- **User Roles and Access**:
    - Role-based screens and controls
        - only doctors can enter treatments and consultation notes.
        - Medical laboratory technician can enter medical reports.
        - Patients can only view treatments and consultation notes.
        - Staff members and Patient both can create appointments.
- **Sample UI Components**:
    - Appointment Scheduler with calendar view and time slots.
    - Patient Registration form.

        - Treatment catalog selector.
        - Billing and payment entry screens.
        - Report generation dashboards.

- **GUI Standards/Style Guides**:
    - Bootstrap CSS framework for consistency and responsiveness.
    - Use clinic's official color scheme and branding for a professional look.
- **Documentation**:
    - Detailed UI designs and mockups to be captured in a separate User Interface Specification Document.

3.2 Hardware Interfaces

- **Supported Devices**:
    - Desktop and laptop computers used by clinic staff and QA team.
    - Patients and doctors can use smart phones, tablets, desktop and laptop computers.
    - Standard input devices: keyboard, mouse, touch screen.
    - Display devices supporting common screen resolutions.
- **Hardware Interaction**:
    - No direct integration with medical devices in this phase.
    - Software interacts with hardware peripherals only through the client operating system (no direct device drivers).
- **Communication Protocols**:
    - The client-server communication over HTTP/HTTPS for web-based UI.
    - Database server communication via standard SQL protocol.
- **Performance Constraints**:

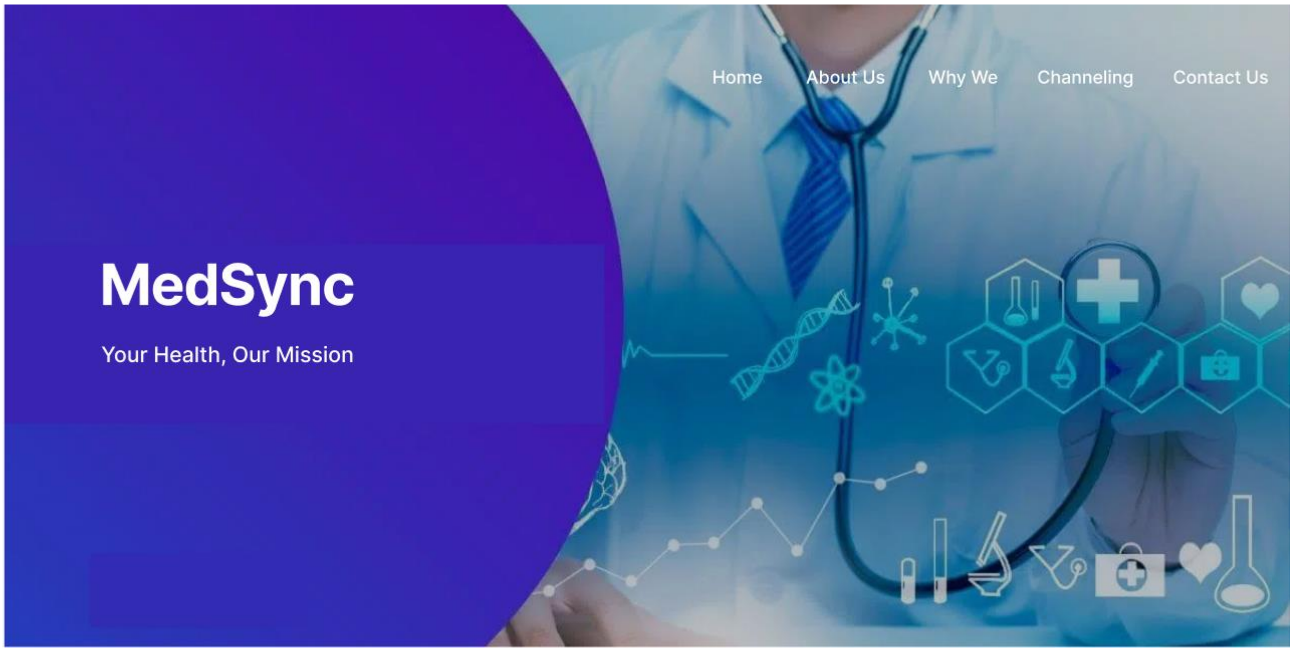
- System optimized for low latency interaction with the database to provide real-time appointment schedule.

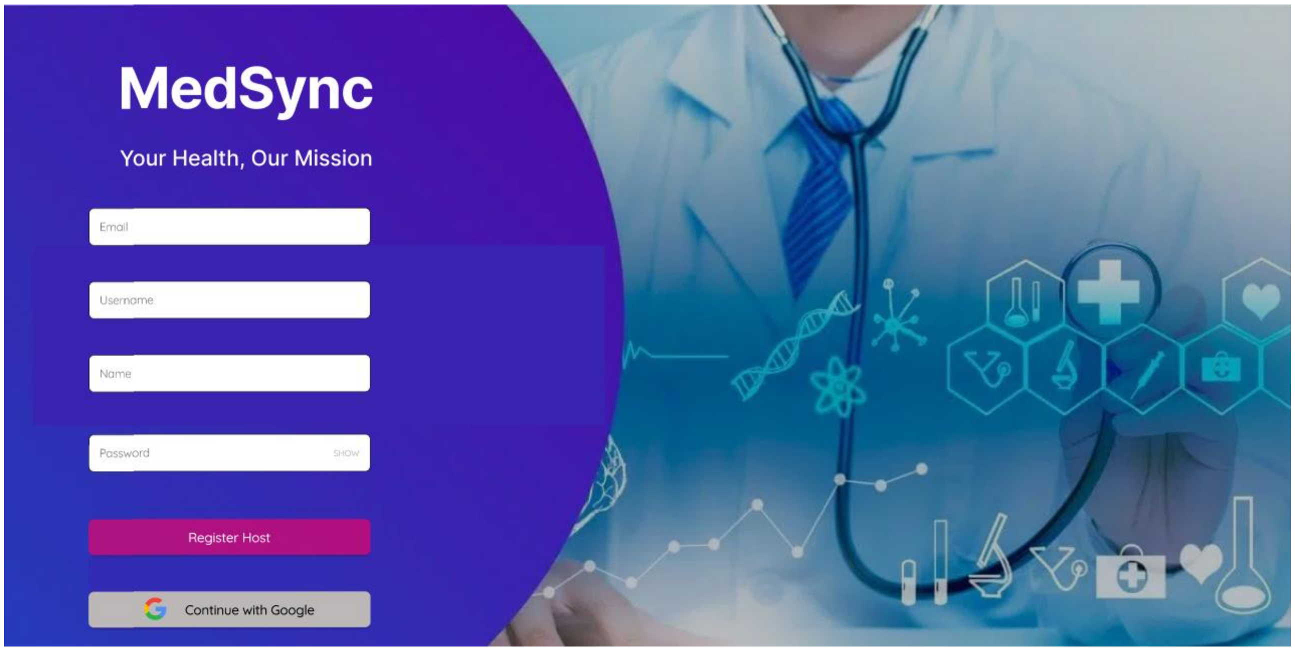
3.3 Software Interfaces

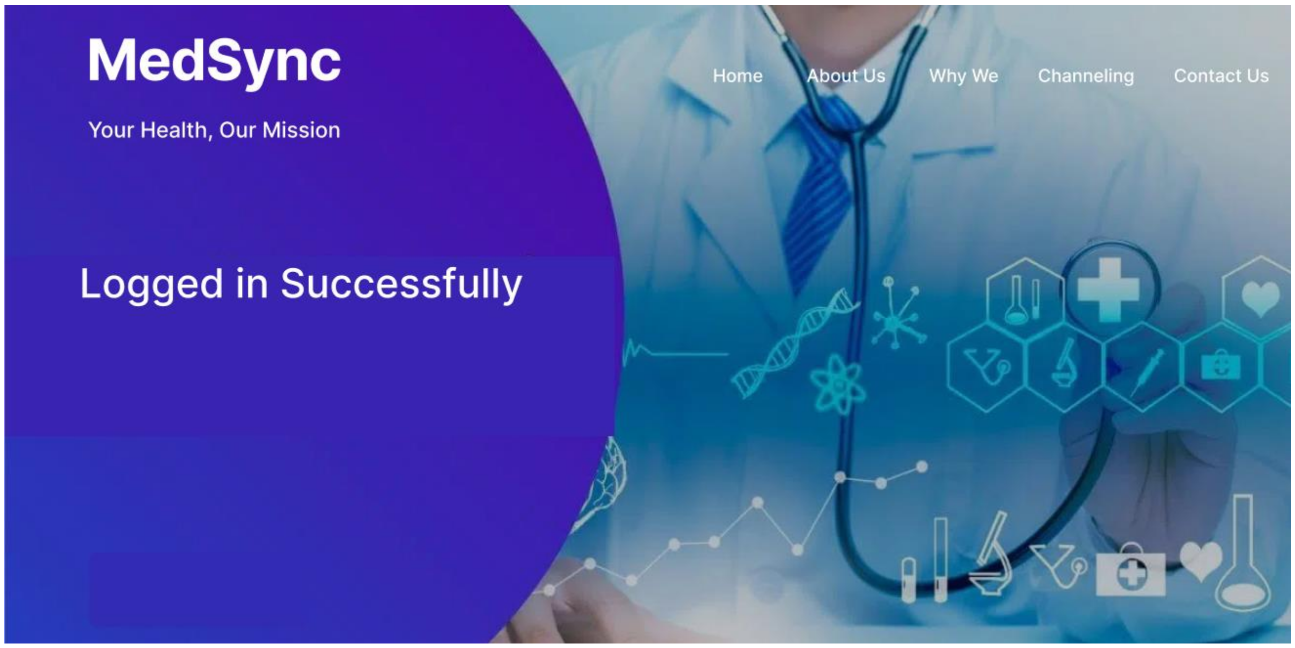
- **Operating System**:
    - Backend server hosted on Linux or Windows Server (version TBD).
    - Client machines running Windows 10/11 or modern web browsers (Chrome, Firefox, Edge).
- **Database**:
    - Relational Database Management System (RDBMS), e.g., MySQL 8.0 or PostgreSQL 14.
    - Data schema designed for ACID compliance.
- **Web Server / Application Server**:
    - Apache Tomcat, Nginx, or equivalent hosting for the web application.
    - RESTful API endpoints serving the UI and database interactions.
- **External Software Components / Libraries**:
    - Frontend: HTML, CSS, JavaScript, ReactJS.
    - Backend: Node.js with Express.js.
    - Authentication Libraries.
- **Data Exchange**:
    - JSON format for client-server communication.
    - All data validation enforced server-side for security and consistency.
- **Shared Data / Implementation Constraints**:

  - Session state managed via secure cookies or tokens.
  - Global error handling and logging mechanisms integrated.
- **API Documentation**:
  - REST API specifications and contracts documented separately.

Proposed few UI designs for the website







Logged in Successfully

3.4 Communications Interfaces

- **Communication Protocols**:
  - HTTP/HTTPS for all web client-server interactions.
  - Secure HTTPS mandatory to encrypt data-in-transit.
- **Messaging and Notifications** (future scope):
  - Email notifications for appointment reminders (SMTP protocol).
  - SMS gateway integration (via REST API) for urgent notifications (if implemented later).
- **Data Formatting**:
  - Standard JSON for API request and response payloads.
  - Date/time formats in ISO 8601 (e.g., `YYYY-MM-DD, HH: mm: ss`).
- **Security**:
  - Use of TLS 1.2 or above for secure communications.

- Input sanitization and validation to prevent injection attacks.
- Role-based access control enforced on all endpoints.

- **Data Transfer Rates & Synchronization**:
  - Optimized for typical broadband connections in clinic locations.
  - Real-time synchronization of appointment status changes across clients.

4. System Features

4.1 Making Appointments

4.1.1 Description and Priority

This feature allows staff to create, view, modify, reschedule, or cancel patient appointments with doctors. It supports both scheduled bookings and emergency walk-ins.

(We have given points on a relative scale from a low of 1 to a high of 9)
Priority: High

- **Benefit**: 9 – Enables core clinic operations
- **Penalty**: 9 – Without it, clinic operations halt
- **Cost**: 3 – Moderate effort to implement
- **Risk**: 5 – Moderate complexity due to conflict checks and concurrency

4.1.2 Stimulus/Response Sequences

Use Case: Book Scheduled Appointment

- **Stimulus**: Staff selects a doctor, patient, and a date/time slot.
- **Response**: System checks for conflicts and saves the appointment if valid.

Use Case: Reschedule Appointment

- **Stimulus**: Staff edits an existing appointment to change date/time.

- **Response**: System verifies availability and updates the record.

Use Case: Emergency Walk-in

- **Stimulus**: Staff enters walk-in details bypassing scheduling.
- **Response**: System records appointment with current timestamp and flags it as walk-in.

Use Case: Cancel Appointment

- **Stimulus**: Staff cancels an upcoming appointment.
- **Response**: Status is updated to Cancelled; slot becomes available again.

4.1.3 Functional Requirements

- **REQ-1**: The system shall allow appointment creation for a patient with a selected doctor and time slot.
- **REQ-2**: The system shall prevent overlapping appointments for the same doctor.
- **REQ-3**: The system shall allow staff to mark an appointment as emergency walk-in.
- **REQ-4**: The system shall update appointment status to "Cancelled" and make the time slot available.
- **REQ-5**: The system shall validate all input fields (e.g., valid date/time, patient and doctor IDs).
- **REQ-6**: The system shall provide real-time feedback if a selected slot is no longer available.
- **REQ-7**: The system shall support viewing upcoming appointments filtered by doctor, branch, or date.

4.2 Treatment Recording and Billing

4.2.1 Description and Priority

Allows doctors to record consultation notes and prescribe treatments after an appointment is marked Completed. The system uses this information to automatically generate invoices and calculate outstanding balances.
Priority: High

- **Benefit**: 8 – Enables revenue tracking and clinical documentation
- **Penalty**: 8 – Without this, billing and reporting become unreliable
- **Cost**: 5 – Requires moderate logic
- **Risk**: 4 – Low concurrency issues but needs careful design

4.2.2 Stimulus/Response Sequences

Use Case: Complete Appointment

- **Stimulus**: Doctor marks appointment as Completed.
- **Response**: System opens treatment entry form and enables invoice creation.

Use Case: Record Treatments

- **Stimulus**: Doctor selects treatments from a catalog and enters notes.
- **Response**: System stores records and calculates the total amount for the invoice.

Use Case: Make Payment

- **Stimulus**: Patient makes full or partial payment against an invoice.
- **Response**: System updates paid amount and recalculates outstanding amount.

Use Case: Insurance Claim Processing

- **Stimulus**: Billing staff submits a claim for eligible treatments.
- **Response**: System logs the claim and tracks reimbursement.

4.2.3 Functional Requirements

- **REQ-8**: The system shall allow doctors to record consultation notes linked to an appointment.

- **REQ-9**: The system shall provide a catalog of predefined treatments with associated service codes and prices.
- **REQ-10**: The system shall generate an invoice upon appointment completion, aggregating treatment costs.
- **REQ-11**: The system shall support partial and full payments toward an invoice.
- **REQ-12**: The system shall update invoice records upon receiving a payment.
- **REQ-13**: The system shall support adding insurance claim entries linked to treatments in an invoice.
- **REQ-14**: The system shall track the reimbursement status for each claim (Pending, Approved, Rejected).

5. Other Nonfunctional Requirements

5.1 Performance Requirements

- **Response Time**: The system should handle common processes, such as ensuring the booking of appointments, records of treatment and the generation of bills need to be completed within one seconds.
- **Scalability**: The system should have the feature of scaling as the bulk of data increases including the number of patients, doctors, treatments, and branches. It should support up to 1000 concurrent users without performance degradation.
- **Transaction Speed**: Tasks such as the scheduling of the appointments, the documentation of the treatments and generation of bills must complete within a very short time. The system should be able to complete each task that occurs in less than 1.5 seconds on average, when the system is at load.
- **Report Generation**: Reports such as branch-wise appointment summary, doctor revenue report, number of treatments per category over a given period should be generated within 2 seconds to ensure timely decision making.
- **System Load**: The system should be designed to handle peak loads for an example like ending of a month, without significant performance degradation.

5.2 Safety Requirements

- **Data Integrity**: The data should be always maintained consistently and precisely in regard to handling the patient records, appointment records, billing and recording of the treatment information without loss or corruption of any data.

- **Backup and Recovery**: The system should perform regular automated backups of the database to prevent data loss. An unplanned interruption of operations should be brought back to a full operational capacity within 4 hours.

- **Disaster Recovery**: A disaster recovery plan must be in place to ensure that the system can be quickly restored after catastrophic failures (server crashes, data corruption). This plan must minimize the downtime and data loss.

5.3 Security Requirements

- **Data Encryption**: All sensitive data, including patient records, billing details, and treatment information, should be encrypted both at storage in the database and its transferring on the web using SSL/TLS protocols

- **Authentication and Authorization**: Role-based access control should be implemented to restrict access based on user roles. Information should only be accessed by the doctors and other workers including the administrative staff when it comes to personally identifying information. Multi-factor authentication should be used for all administrative users.

- **Audit Logs**: The system must maintain detailed logs of all user actions, including appointment creation, treatment recording, and billing to support security auditing and compliance.

- **Session Management**: Session of the user needs to be managed with session time out and secure log in failure to protect unauthorized user access.

5.4 Software Quality Attributes

- **Usability**: System should be user friendly to help in reducing training time of the staff. The UI has to be minimalistic, clear, and interactive for doctors, administrative personnel, and patients.

- **Reliability**: The system must operate without frequent errors or crashes. Any operation must always be carried out with no data loss or error.

- **Maintainability**: The system should be modular, so that updating and maintenance can be easily done. The Database schema and code should be well documented to assist developers and administrators in maintaining and enhancing the system.

- **Scalability**: The system must be scalable, able to manage increasing data (for an example adding more branches, patients, doctors, treatments) without further reduction in performance.

- **Portability**: The system should be compatible with all branches and be deployable in different environments.
- **Interoperability**: The system must facilitate integration with other systems, including external billing systems, insurance companies, and hospital management programs, through the more open specifications.

5.5 Business Rules

- **Appointment Scheduling**: A patient can only book an appointment with one doctor at a given time. No doctor should have overlapping appointments.
- **Appointment Status**: There is a Scheduled, Completed, and Cancelled appointment status. Once marked as Completed, treatment and consultation notes must be recorded, and bills is created on the basis of the provided services.
- **Treatment and Billing**: All the services that are given to a patient should be documented with the price that was charged and a service code.
- **Insurance Claims**: The system should track insurance coverage for eligible patients. If a patient is insured, the system should calculate the insurance coverage and the amount of money patient will have to pay.
- **Payment Tracking**: The system should track full and partial payments made by patients. If a patient has an outstanding balance, the system must report the balance to the administrative staff.
- **Rescheduling and Walk-ins**: The system must support appointment rescheduling and accommodate emergency walk-ins which are appointments made without prior booking.

6. Other Requirements

6.1 Database Requirements

- The system must use a relational database to ensure data consistency and enforce referential integrity through foreign keys.
- All transactions related to appointment scheduling, treatment recording, and billing must follow ACID properties to prevent data anomalies.
- Indexes must be created on frequently queried fields to improve the speed of common reports and queries.

6.2 Internationalization / Localization

- The system should be designed to support multiple languages (English, Sinhala, and Tamil) for the user interface and reports. Initially only in English.
- All dates, times, currency symbols, and number formats must adapt to the local settings of each clinic branch.

6.3 Legal and Compliance Requirements

- The system must comply with local data protection laws.
- Patient data must be encrypted both at rest and in transit.
- Audit logs must be retained for at least five years to support regulatory audits.

6.4 Integration and Extensibility

- The system should expose REST APIs for future integration with external systems, including insurance providers and laboratory management systems.
- Should support adding SMS and email gateways to send appointment reminders and billing notifications to patients.
- Must be modular to allow integration of new features (e.g., telemedicine) with minimal changes to existing components.

6.5 Future Enhancements

- Patient self-service portal for booking appointments, checking bills, and viewing medical history.
- Telemedicine module to allow video consultations.
- Mobile application for patients and doctors.

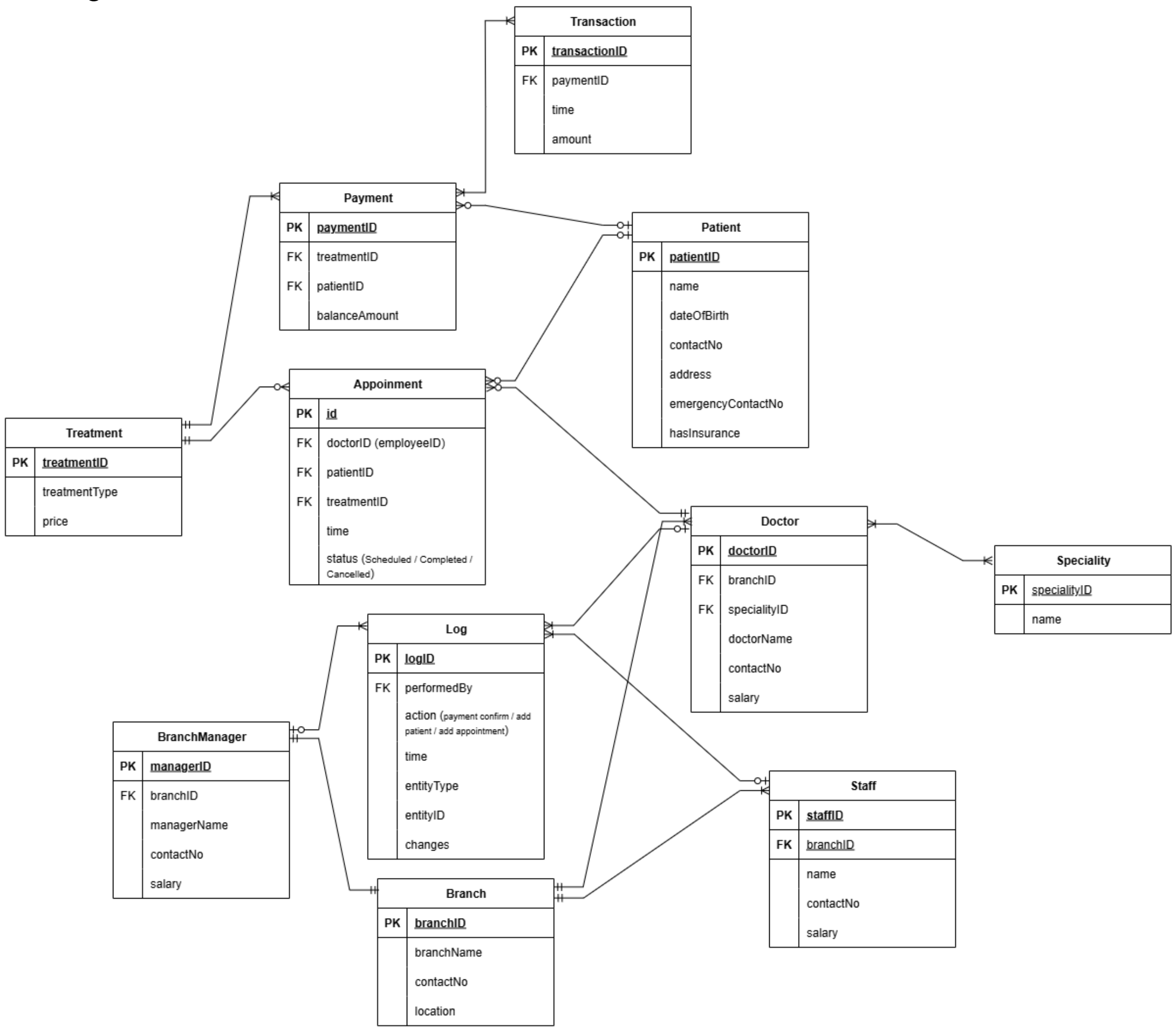
Appendix A: Glossary

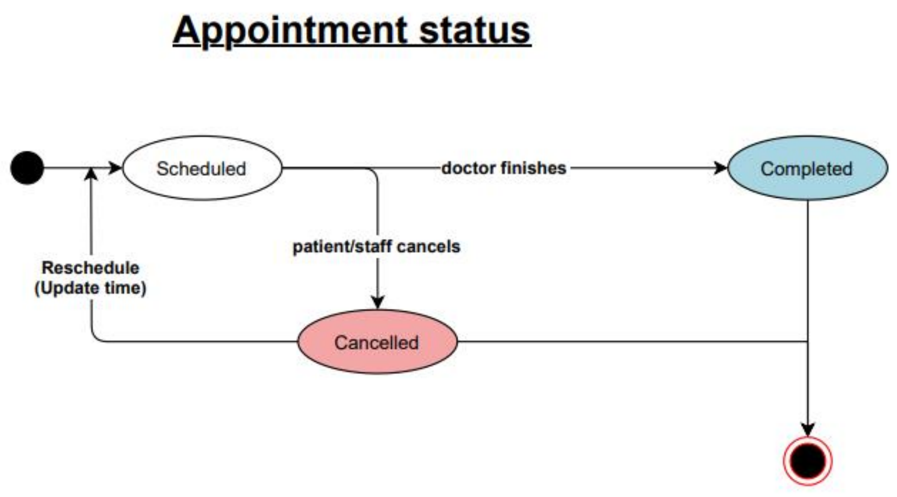
Term	Definition
Appointment	A scheduled consultation between a patient and a doctor for a specific time slot.
Emergency walk-in	An appointment created by staff without prior booking, usually in urgent situations.
Treatment catalogue	Predefined list of services offered by the clinic, each with a service code and price.
Outstanding balance	Amount still owed by a patient after partial payments or insurance adjustments.
Branch	Physical location of the clinic (e.g., Colombo, Kandy, Galle).
Billing	Process of generating invoices for completed treatments and consultations.
Insurance claim	Request to an insurer to cover all or part of the cost of eligible treatments.

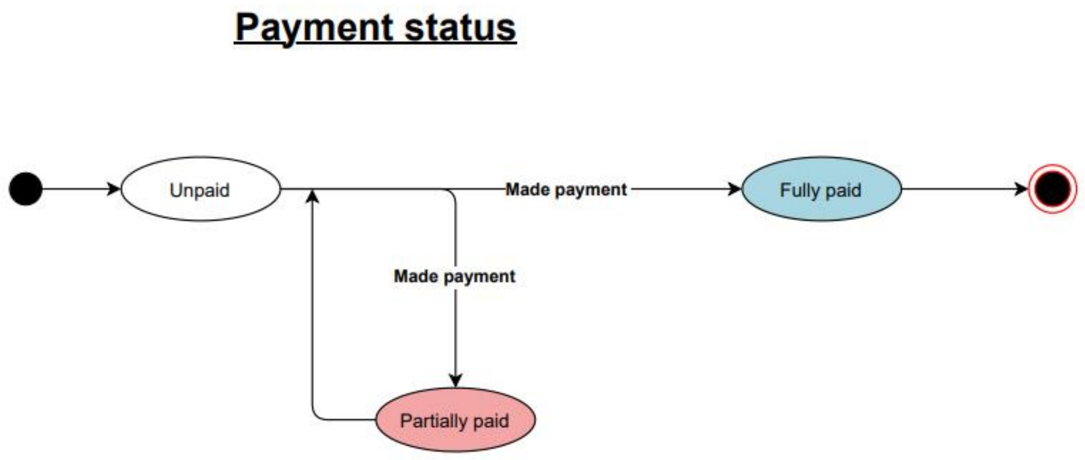
Appendix B: Analysis Models

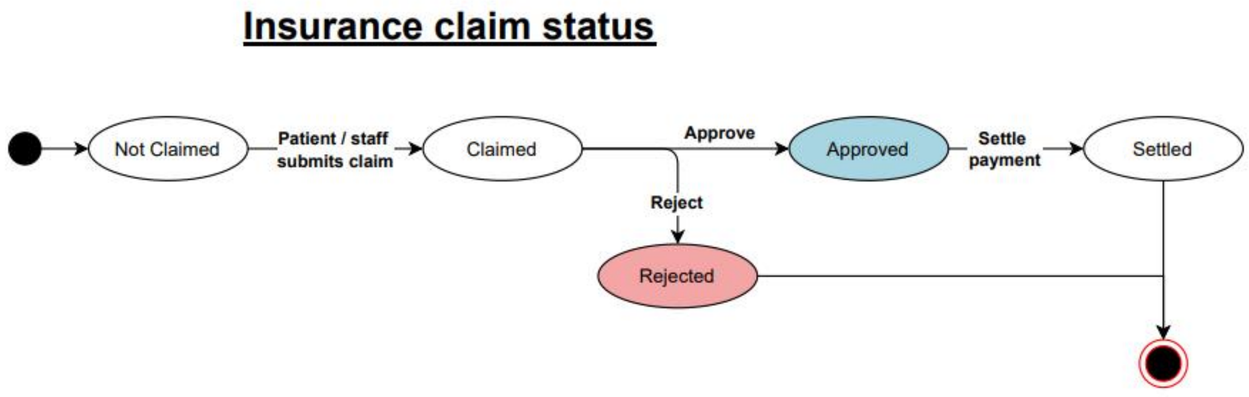
- **ER Diagram** showing entities such as Patient, Employe, Branch, Appointment, Treatment, Transaction, Payment, Doctor – Speciality, Log.
- **Use Case Diagram** identifying key actors: Patient, Doctor, Admin Staff, Branch Manager.
- **Sequence Diagram** (optional): e.g., showing the flow for scheduling an appointment or generating a bill.

ER Diagram




Appointment status
Scheduled
doctor finishes
Completed
patient/staff cancels
Reschedule (Update time)
Cancelled


Payment status
Unpaid
Made payment
Fully paid
Made payment
Partially paid


Insurance claim status
Not Claimed
Patient / staff submits claim
Claimed
Approve
Approved
Settle payment
Settled
Reject
Rejected

Appendix C: To Be Determined List

Detailed insurance coverage rules per treatment	Pending
UI framework selection	Pending
Exact list of reports required by management	Pending

Appendix D : Assumptions

- Each doctor can only have one appointment at a specific time slot.
- Each appointment lasts for a standard fixed duration (20 minutes).
- Only staff can add emergency walk-ins; patients cannot self-book them.
- Only administrative users have permissions to modify billing and insurance claim data.

Appendix E: Technical Considerations

- Use of transaction isolation to prevent double-booking when concurrent users schedule appointments.
- Storing audit logs in a separate secure table to avoid affecting performance of main transactional tables.