IT2080 IT PROJECT

Project Proposal Report

Group Details

Campus: SLIIT Campus Malabe

Group Number - ITP25_WE_B01_01_204

Topic - Medical Center Management System

Date of submission – 08/08/2025

Team Members

IT Number	Student Name	Student Email Address	Contact Number
IT23588714	K.H.S.Dinsara	it23588714@my.sliit.lk	0703287271
IT23543232	Navodya A.K.	it23543232@my.sliit.lk	0764449102
IT23572638	Thathsarani J R	it23572638@my.sliit.lk	0703119829
IT23810464	Ekanayaka E.W.I.D	it23810464@my.sliit.lk	0768370886
IT23669758	Liyanaarachchi L.A.D.A	it23669758@my.sliit.lk	0772712630

Contents

01. Background

Healthcare is one of the most vital sectors in society, requiring efficiency, accuracy and effective coordination among staff to ensure high quality patient care. Medical centers, especially midsized private clinics, often provide a range of services including general conclusions, laboratory tests, pharmacy services, and administrative functions. However, many centers in Sri Lanka still rely on paper based records or outdated standalone software systems. These methods are prone to errors, delays, and inefficiencies that directly affect patient satisfaction and treatment outcomes.

The proposed client is a private medical centers that provides outpatient consultations, laboratory testing, pharmacy services and basic health screening packages. The center operates with a team of doctors, nurses, lab assistants, pharmacists, and administrative staff, serving an average of 150 to 200 patients daily. The management has identified the need for a centralized, web based solution to streamline all operations from patient registration to billing to improve service quality, reduce administrative burdens and enhance patient engagement.

02. Problem and Motivation

02.1. Problem Statement

Currently, the medical center manages its patient appointments, medical records, billing, and inventory through manual processes or isolated applications that are not interconnected. This results in**:**

- **Scheduling conflicts** due to lack of centralized appointment management
- **Incomplete medical histories** because records are scattered across multiple files or departments
- **Delayed updates** on test results and prescriptions
- **Inefficient inventory tracking**, leading to stock outs or overstocking of
- medicines and medical supplies
- **Poor patient communication**, as updates are not provided in real time

02.2. Motivation

- By introducing a centralized medical center management System (MediCura), the center can
- Improve patient satisfaction through faster service, real-time updates and online access to records
- Reduce human errors in scheduling, recordkeeping, and billing
- Enable doctors and staff to make better decisions with instant access to patient data
- Optimize inventory management and reduce wastage
- Provide management with insights through detailed reports for better resource allocation and strategic planning

03. Aim and Objectives

03.1 Aim

- To design and develop a centralized, web-based Medical Center Management System (MediCura) that integrates appointment scheduling, electronic medical records, billing, inventory management, and reporting to improve operational efficiency, reduce errors and enhance patient experience**.**

03.2 Objectives

1. **Develop an appointment scheduling module** that allows patients and receptionists to manage bookings, cancellations and reminders in real time
2. **Implement an electronic medical records system** for doctors, nurses and lab assistants to access and update patient histories, prescriptions and test results.
3. **Integrate billing and payment processing** to handle consultation fees, lab charges and pharmacy transactions securely.
4. **Create an inventory management module** for tracking medicines, lab kits and medical supplies with low stock alerts
5. **Enable role based access control** to ensure secure and authorized access to system features.
6. **Design a reporting dashboard** for management to monitor patient flow, staff performance and financial summaries.
7. **Provide a notification system** to keep patients informed of appointments, test results and prescriptions
8. **Ensure system scalability and data security** through the use of secure web technologies and encrypted data storage

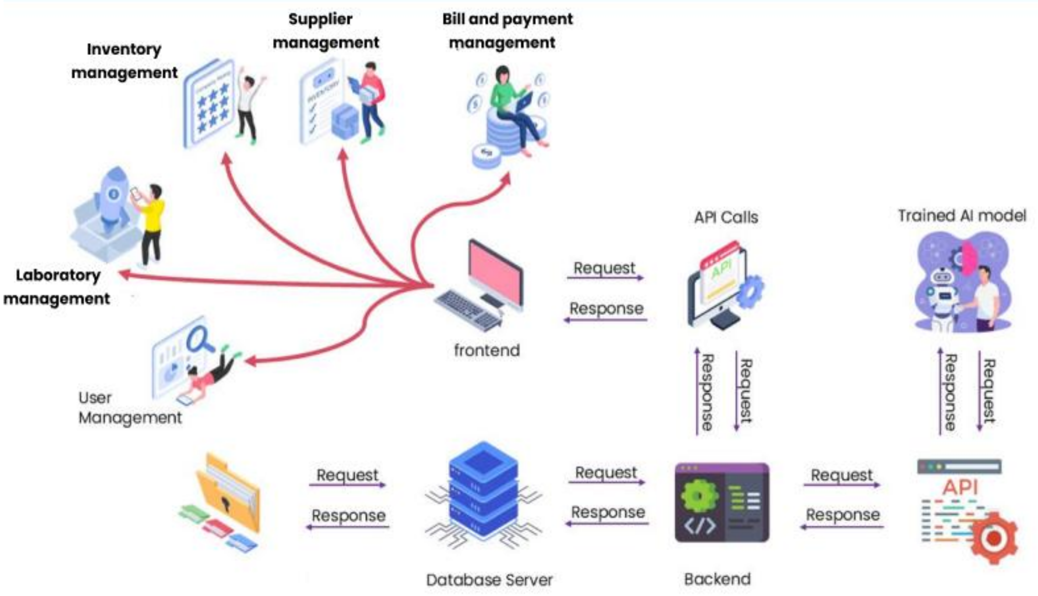
04. System Overview

04.1 Overview

The **MediCura** Medical Center Management System will be a **centralized, Web based platform** that connects patients, doctors, lab assistants, pharmacists, receptionists, and managers in a single system. The platform will provide modules for appointment scheduling, electronic medical records, billing, inventory management, reporting, and notifications.

The system is divided into five different crucial sub-systems in order to be developed efficiently within the time frame that we were given.

- **User Management**
- **Supplier Management**
- **Laboratory Management**
- **Bill and payment Management**
- **Inventory Management**



04.2. Functional and Non-functional requirements (User-wise)

04.2.1. Functional Requirements

1. **User Management and Booking Appointments**
    - User registration and login
    - Role-based access control (Patient, Staff, Manager)
    - Secure password handling (Encryption / Authentication)
    - Patient books appointments by selecting date, time, and type
    - Assign and available doctor – adding doctors and manage them
    - Patients can view, edit, or cancel appointments
    - Manager their profile and track medical records

2. **Laboratory Management**
    - Check the patients test request and identify required lab tests
    - Track the tests progress from sample collection to result generation
    - Generate a test cost slip or Invoice for the patient
    - Keep the patient updated about test status and result availability
    - Collect and label patient samples accurately
    - Verify test results before approval

3. **Inventory Management**
    - Component operations for medicines (Create, Read, Update, Delete)
    - Low-stock alert system based on reorder levels
    - Purchase order generation when stock is low
    - Supplier information management
    - Inventory usage logs and history tracking
    - Barcode/QR code generation for labeling parts (using externalAPI)

4. **Supplier Management**
    - Automatic purchase order generation
    - Order and delivery history tracking
    - Supplier information management
    - Supplier record operations

5. **Billing and Payment Management**

- Transaction recording and tracking
  (Invoices, payments, transactions, receipts)
- Budget management and expense monitoring
- Profit and loss analysis reports
- Integration with inventory for billing
- Supplier payment scheduling and tracking
- Tax calculation and compliance support
- Financial statement generation (income, balance sheet)
- Export options for reports.
  (PDF and Excel formats for record keeping and auditing)

04.2.2. Non-Functional Requirements

1. **Usability**

➢ The system should be simple and easy for all types of users, including doctors, patients, and administrative staff.

- Clear and logical navigation menus.
- Consistent design across all modules.
- Minimal training required for new users.
- Error messages that are easy to understand and help users fix mistakes

2. **Accessibility**

➢ The system must be available and usable on different devices and platforms to ensure convenience.

- Works on desktop, tablet, and mobile.
- Compatible with all major browsers
  (Chrome, Firefox, Edge, Safari).
- Responsive design that adjusts to different screen sizes.
- Multi-language support for English, Sinhala, and Tamil.

3. **Availability**

➢ The system must be operational and accessible at all times with minimal downtime.

- 99.9% uptime guarantee.
- Automated daily backups of all critical data.
- Disaster recovery plan for quick restoration after system failure.
- Server monitoring to detect and resolve issues promptly

.

4. Efficiency

- ➢ The system should make the best use of resources and respond quickly to user actions.

- Page load time under 3 seconds during normal usage.
- Optimized database queries for faster data retrieval.
- Handles multiple simultaneous users without lag.
- Low memory and CPU usage to reduce hosting costs.

5. Auditability

- ➢ The system should allow tracking and reviewing all important actions for accountability and compliance.

- Detailed logs for all user activities.
- Tracking changes to medical records, inventory, and financial data.
- Searchable audit trail for specific dates, users, or actions.
- Reports for system administrators to review past activities.

6. Privacy

- ➢ The system must protect personal and medical data from unauthorized access or disclosure.

- Data encryption during storage and transmission.
- Role-based access to restrict sensitive information.
- Compliance with data protection laws
- Automatic logouts after periods of inactivity.

7. Performance

- Fast response times even during high usage periods.
- Capable of processing large amounts of data efficiently.
- Uses caching techniques to improve loading speed.

8. Security

- Strong authentication and password policies.
- Protection against cyber threats

(SQL injection, XSS, CSRF).
- Secure data backups stored in encrypted format.

9. Confidentiality

- Only authorized users can view sensitive data.
- Strict control over sharing of medical and financial records.

10. Scalability

- System can handle an increase in users, patients, and transactions without performance loss.
- Supports future upgrades and additional modules.

11. Reliability

- Stable system operation without frequent crashes.
- Fault-tolerant design to prevent data loss.

12. Maintainability

- Modular code structure for easy updates.
- Well-documented system for smooth maintenance and troubleshooting.

04.3. Technical Requirements

➢ Front-End Development:

- React.js for building a responsive, modular, and scalable user interface.
- Uses CSS Grid and Flexbox for flexible layouts compatible with phones, tablets, and desktops.
- Progressive Web App features to provide a mobile-app-like experience.
- Real-time data updates with Context API or Redux and API integration.
- Accessibility compliance using WAI-ARIA standards.
- Performance optimizations such as lazy loading and code splitting.

➢ Back-End Development:

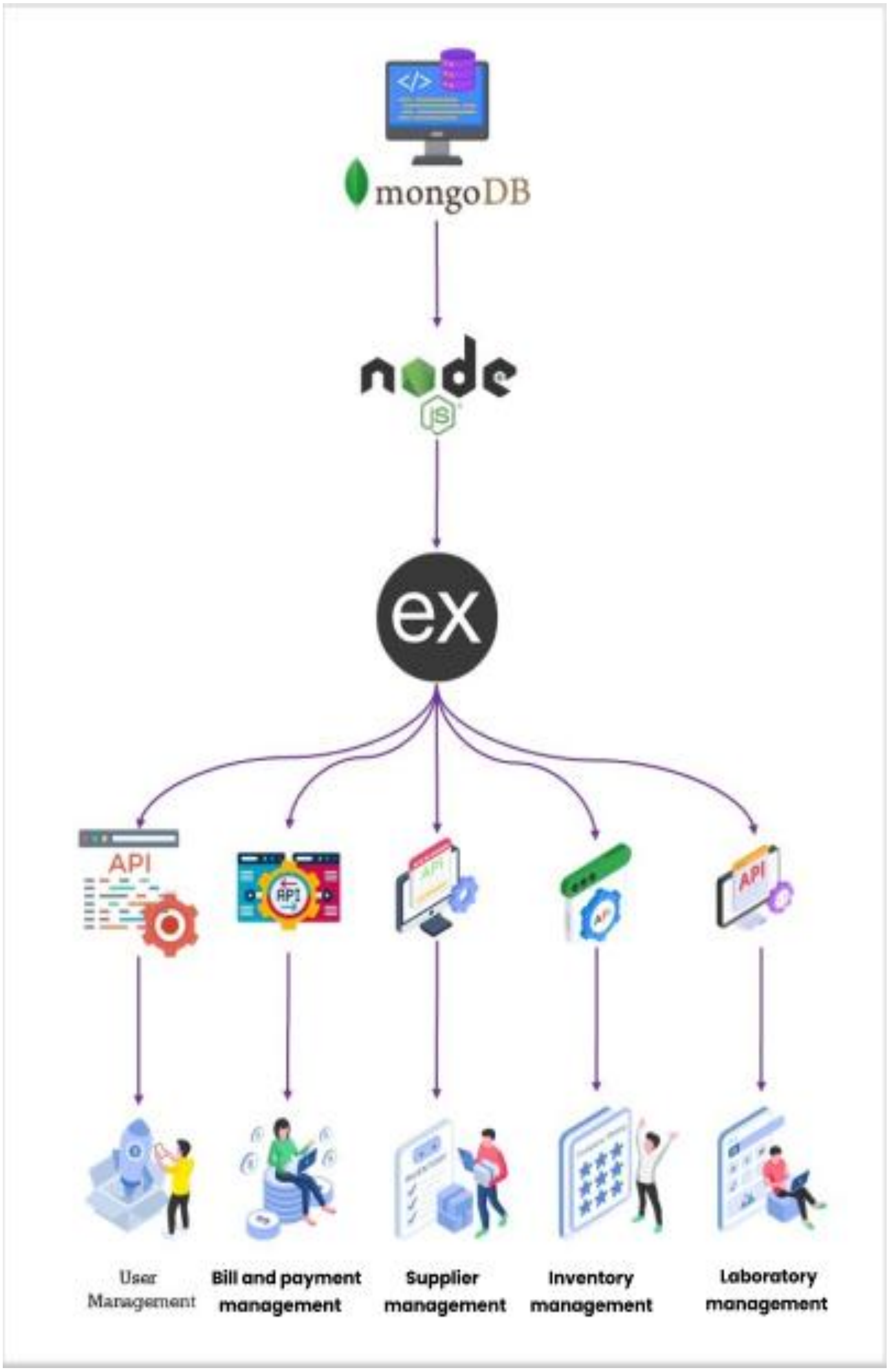
- Node.js runtime environment for building fast and scalable server-side applications.
- Express.js framework to create RESTful APIs that facilitate smooth communication between frontend and backend.
- Designed for scalability, supporting future modularization and microservices architecture if needed.
- Use of middleware for handling authentication, authorization, and role-based access control.
- Asynchronous, non-blocking I/O to efficiently manage time-consuming tasks and maintain high responsiveness.
- Logging and monitoring implemented using tools like Winston or Morgan for tracking system events and performance

➢ Database:

- MongoDB, a NoSQL document-oriented database, used to store user, inventory, order, and medical data in flexible JSON-like documents.
- Performance optimization through indexing, aggregation pipelines, and efficient schema design.
- Data security ensured with role-based access controls, authentication mechanisms, and TLS/SSL encryption for data in transit.
- Backup and recovery strategies implemented using MongoDB's built-in tools like mongodump and cloud backup services.
- Supports advanced querying and aggregation for generating custom reports and analytics.

- Database connectivity managed via Mongoose ODM, providing schema validation and streamlined data operations in Node.js..

- **Backend Testing:** Postman API
- **IDE:** Visual Studio Code.


mongoDB
node JS
ex
API
API
API
API
API
User Management
Bill and payment management
Supplier management
Inventory management
Laboratory management

05. Literature Review

- In developing the MediCura Medical Center Management System, it is important to review existing studies, technologies, and best practices related to healthcare management systems. This section examines relevant research on healthcare information systems, appointment and records management, inventory and billing systems, and user interface considerations.

05.1. Healthcare Information Systems (HIS)

- Healthcare Information Systems are designed to improve the quality and efficiency of healthcare delivery by centralizing patient and operational data. According to Bassi and Lau (2013), HIS can reduce medical errors, improve patient outcomes, and streamline administrative processes. However, many clinics in Sri Lanka still rely on paper-based or standalone systems, which hinder communication and cause delays in service delivery. The MediCura system aims to address these gaps by offering an integrated, web-based solution

05.2. Appointment and Medical Records Management

- Efficient appointment scheduling reduces patient wait times and prevents scheduling conflicts. Research by Al-Mutairi et al. (2020) highlights that centralized appointment systems improve patient satisfaction and clinic efficiency. Similarly, Electronic Medical Records (EMR) systems allow healthcare professionals to store and retrieve patient histories, prescriptions, and lab results instantly, leading to better clinical decision-making (Rahman & Abdullah, 2021). Successful implementations, such as Epic and Cerner, show the importance of secure, accessible, and real-time record management.

05.3. Inventory and Billing Systems in Healthcare

- Effective inventory management ensures the availability of medicines and medical supplies while minimizing wastage. Kumar and Singh (2021) emphasize that automated inventory systems with reorder alerts significantly improve supply chain performance. Integration of billing systems with inventory modules ensures accurate financial tracking and transparency (Patel et al., 2020). The MediCura system will combine inventory management with billing, enabling smooth pharmacy transactions, consultation fee processing, and report generation.

05.4. User Interface and User Experience (UI/UX) in Healthcare Applications

- The usability of healthcare systems is critical for adoption by medical staff and patients. Nielsen (1993) highlights that simplicity, clear navigation, and consistency are key to effective UI/UX design. In the healthcare sector, where speed and accuracy are vital, responsive design and accessibility features (such as WAI-ARIA standards) ensure that systems are inclusive and easy to use across multiple devices. Systems like My Chart and Health Hub demonstrate how intuitive interfaces can improve engagement and overall satisfaction.

06. Methodology (Agile Methodology)

06.1. Agile Framework Implementation

We implemented the Scrum methodology for our Medical Center Management System.What this means is we build the system in small sections named sprints, which take 2–3 weeks each. Each sprint produces working features that doctors, laboratory assistants, receptionists, and patients can view and test.

We sit down regularly with medical personnel, administrators, and other critical users and get feedback from them. That way, we develop what they truly need, rather than what we think they need. At the conclusion of every sprint, we discuss what worked and where we can improve.

Agile Benefits:

- Quick testing with real medical center users.
- Flexible changes as healthcare needs evolve.
- Continuous testing throughout development.
- Frequent delivery of working system features.
- Improved team communication.
- Early identification of issues and bugs.

06.2. Requirements Engineering Methods

6.2.1 Stakeholder Analysis and Engagement

We start by communicating with all of the different Medical Center Management System users, including physicians, patients, pharmacists, the manager of the medical center, suppliers, lab assistants, and receptionists.

To ascertain each group's unique requirements and system expectations, we interview them. We analyze how activities now operate using straightforward techniques and develop user stories that outline each user's experience interacting with the future system. This exercise ensures that we concentrate on developing appropriate functionality. Interviews with user groups, including physicians, patients, pharmacists, receptionists, lab assistants, suppliers, and the management, are how we collect requirements. Gathering input from the group and identifying common needs.

- Observing current work flows to understand current workflows and pain points
- Creating user stories that embody actual uses for each given role.
- Validating requirements through reviewing them with the actual users

Creating early prototypes and presenting them to users to get feedback and validation

6.2.2 Requirements Documentation and Management

We write down all requirements clearly in one place. User stories are written as: "As a user, I want feature so that benefit." Each story has clear rules for when it's done correctly. We track how each requirement fits into the system design. We review these requirements regularly with users to keep them accurate.

06.3. Design Methods

6.3.1 User-Centered Design Approach

We design the system thinking about users first. We want it to be easy to use and accessible for everyone. We hold workshops with users to understand what they really need. We create simple sketches called wireframes, then build more detailed prototypes. This lets users see how the system will work before we build it. Design Process:

- Creating user profiles
- Planning easy navigation
- Making simple sketches and interactive models
- Designing the look and colors
- Making sure it works on phones and on computers
- Making it easy for everyone to use

6.3.2 Database Design Methodology

We design the database in steps. First, we draw simple diagrams showing how data connects. Then we create a proper database structure that avoids duplicate data and keeps information accurate. We work closely with business people to make sure the database matches real business needs. We also plan how to move data from old manual systems to the new digital system.

06.4 Development Tools and Technologies

6.4.1 Frontend Development with React.js

We use React.js to build the user interface because it makes fast and smooth screens. React helps us create reusable parts, so the whole app looks consistent. We use modern JavaScript tools to keep code clean and manageable. We also use CSS tools to make sure the app looks good on all devices.

Frontend Tools:

- React.js for building user interfaces
- Modern JavaScript for app logic
- CSS frameworks for responsive design
- Tools for managing app data
- Build tools for fast performance
- Testing tools for frontend parts

6.4.2 Backend Development with Node.js

We use Node.js for the backend because it's lightweight, fast, and efficient. It handles key backend features like user authentication, data access, and APIs. We build Restful APIs using Express.js to allow smooth communication between the frontend and backend. Node.js helps keep our code modular, maintainable, and easy to test backend Tools:

- Express.js for fast and flexible API development
- JWT for secure user authentication
- MongoDB for handling database operations
- REST APIs for frontend-backend communication
- npm for managing packages and libraries
- Jest for backend testing

6.4.3 Database Technologies and Design

We use relational databases to keep data accurate and organized. The database is set up to avoid duplicate information and maintain proper connections. We use indexing to make searches faster and have backup plans to protect important data.

6.4.4 Testing Methods and Quality Assurance

We test the system at different levels. We test small parts first, then check how they work together, then test the whole system like a real user would. We use automated tools to run tests regularly, which helps us find problems early.

Testing Types:

- Unit testing for individual features
- Integration testing for how parts work together
- System testing for the complete system
- User acceptance testing with real users
- Performance testing for multiple users
- Security testing for safety

6.4.5 Integration Methods and API Design

We use REST APIs to connect the frontend and backend smoothly. These APIs follow clear rules to stay organized and easy to manage. We document APIs well so developers know how to use them. We also build in security so only the right people can access information.

6.4.6 Deployment and DevOps Practices

We use modern tools to make sure the system works perfectly from development to live use. We use containers so the app runs the same way on all computers. We have automated systems that test and release updates quickly without mistakes. We also monitor the system closely and have backup plans if something goes wrong.

06.5 Work Breakdown Structure and Team Organization

6.5.1 Project Phase Organization

Each step of the Medical Center Management System project has a defined set of deliverables, objectives, and quantifiable success criteria. To guarantee that the system satisfies practical requirements, it is essential that important stakeholders-physicians, patients, pharmacists, the medical center manager, suppliers, laboratory assistants, and receptionists-be actively involved in every stage.

Initiation Phase:

Consult with stakeholders (such as physicians, patients, pharmacists, etc.) to determine project scope and timetable, analyze responsibilities, and collect preliminary requirements.

Analyze Phase:

Document system specifications based on user feedback after conducting a thorough investigation of healthcare workflows, including patient registration, prescriptions, lab testing, stock handling, etc.

Design Phase:

Create user interfaces, database structures, and system architecture specific to various roles, including clinicians (for prescriptions), lab assistants (for test results), and appointment schedulers.

Development Phase:

Using agile sprint cycles, create system features based on stakeholder demands, such as appointment scheduling, inventory management, lab report creation, patient medical history, etc.

Testing Phase:

Conduct ongoing testing, which includes:

- Unit tests for components such as lab test updates and billing
- Cross-role integration tests (e.g., doctor-lab-patient flow)
- Testing user acceptability with end users, such as medical center management and pharmacists

Deployment Phase:

Set up the system in the medical center setting, move any data that is already there, and train stakeholders according to their roles so they can utilize their interfaces efficiently.

Maintenance Phase:

Provide ongoing assistance, such as bug patches, performance tracking, and the introduction of improvements in response to user requests or changing healthcare center requirements.

6.5.2 Work Breakdown Structure - Management Area Based Allocation

The project follows Agile methodology with management area-based responsibility allocation among team members. Each team member is assigned specific functional areas based on their expertise and interests, ensuring comprehensive coverage of all business requirements while maintaining clear accountability and ownership.

07. Evaluation Method

07.1 Performance Evaluation

To ensure the Medical Center Management System performs reliably under different conditions, we conduct performance testing focused on speed, scalability, and resource efficiency. This include assessing the system's responsiveness to commands, ability to manage several users at once, and effectiveness in using memory and CPU. Monitoring mobile performance, including battery usage and network speed, is also essential especially for roles like doctors and field staff who may access the system via mobile devices.

Performance Testing Areas:

- Examine response times for critical operations, such as scheduling appointments, creating reports, and updating inventories.
- Test with several users at once, including lab assistants, doctors, and receptionists. Track the battery life and load time of your mobile device.
- For quicker access to patient records, prescriptions, etc., assess database performance and make sure queries are optimized.
- Check the speed and stability of the network, particularly for cloud-based or remote access components.

07.2 Functional Testing Evaluation

We will verify that all system features work as expected and using test cases for each part of the system. Both automated and manual tests will be used.

- Verifying business rules and user stories
- Check that the features of the system satisfy the needs and guidelines of the doctors, lab assistants, administrators, patients, pharmacists, and receptionists.

- Verifying the accuracy and validity of data.
- Check that patient records, medications, test results, inventory data, and appointment information are processed, stored, and displayed correctly.
- Testing the integration of modules and external services.
- Check that modules like lab reporting, pharmacy, patient registration, and supplier order systems all function as a cohesive unit. Verify any connections to outside services as well (e.g., email or SMS notifications).
- Handling errors and exceptions.
- Make that the system provides unambiguous messages and guards against data corruption by testing its response to incorrect inputs, system malfunctions, or user errors. Regression testing following updates.
  After each new feature or bug fix, retest the current features to make sure nothing breaks.

The system will also be tested by end users (User Acceptance Testing) to make sure it satisfies practical requirements. Their feedback will be used to improve the system before launch.

07.3 Usability Evaluation

We will evaluate how simple the system is to use. This involves testing whether users can accomplish tasks effectively, determining areas of misunderstanding, and gathering feedback on usability and design.

Usability Testing Components:

- The rate of task completion
- Usability and navigation flow analysis
- Surveys of user satisfaction
- Accessibility for all user types
- Compatibility across devices and browsers

07.4 Security and Reliability Evaluation

We will carry out extensive security testing to protect sensitive medical data and stop unwanted access in order to guarantee the Medical Center Management System is reliable and safe. This entails protecting patient data from common risks like SQL injection and session hijacking, as well as validating login systems.

- Authenticity and authorization confirmation
  Make sure that the right system regions are only accessible by authorized users, such as physicians, patients, pharmacists, and lab assistants.
- Security of data and encryption
  Make sure that prescription medications, billing data, and medical records are all secured and maintained safely.
- Preventing injection attacks and verifying input
  Stop efforts to abuse forms, database queries, or URLs, as well as unwanted input.

- Network and session security
  Examine the data protection and secure session management (timeout, logout, and token handling) during network transfers.
- Regular evaluations of vulnerabilities
  To find and fix security flaws, do routine scans and manual tests.

07.5 Business Impact Evaluation

We will assess the impact of the system on business performance. This includes measuring efficiency, cost savings, customer satisfaction, and employee productivity.

Business Impact Metrics:

- Enhancement of task completion time improvements
- Reduced operational costs
- A better experience for customers
- Increased productivity of employees
- Return on investment and financial benefit analysis

To ensure the **long-term success** of the Medical Center Management System and to identify areas for **future improvement**, we will continue to **monitor system performance** and **track user adoption** even after deployment. Ongoing feedback from doctors, patients, pharmacists, and other stakeholders will guide future updates, optimizations, and feature enhancements.

08. References

- Al-Mutairi, H. A.-M. (2020). *Improving healthcare appointments scheduling system using web-based technologies. International Journal of Advanced Computer Science and Applications, 11(6), pp.657–662. Available at: https://doi.org/10.14569/IJACSA.2020.0110679 [Accessed 7 Aug. 2025].*

- Bassi, J. &. (2013). *Measuring value for money: a scoping review on economic evaluation of health information systems. Journal of the American Medical Informatics Association, 20(4), pp.792–801. Available at: https://doi.org/10.1136/amiajnl-2012-001422 [Accessed 7 Aug. 2025].*

- Corporation, E. S. (2022). Retrieved from Epic EMR Software Overview. [online] Available at: https://www.epic.com/ [Accessed 7 Aug. 2025].

- HealthHub. (2023). Retrieved from HealthHub - Your Digital Health Companion. [online] Available at: https://www.healthhub.sg/ [Accessed 7 Aug. 2025].

- Kumar, R. &. (2021). Retrieved from Inventory management in healthcare: A case-based approach to reduce drug wastage in hospitals. Journal of Supply Chain Management Systems, 10(1), pp.34–45.

- Nielsen, J. (1993). Retrieved from Usability Engineering. 1st ed. San Diego: Academic Press.

- Patel, K. S. (2020). Retrieved from Integrated hospital management system for pharmacy and billing. International Journal of Engineering and Advanced Technology, 9(5), pp.520–524. Available at: https://doi.org/10.35940/ijeat.E9498.069520 [Accessed 7 Aug. 2025].
- Rahman, A. &. (2021). Retrieved from The impact of electronic medical records on healthcare quality: A literature review. International Journal of Medical Informatics, 150, p.104454. Available at: https://doi.org/10.1016/j.ijmedinf.2021.104454 [Accessed 7 Aug. 2025].

09. Appendix

09.1 System Architecture Documentation

Our system uses three-tier architecture:

- Frontend: It is built with React.js to create user interfaces.
- Backend: Node.js manages business logic, API handling, and user authentication.
- Database: MongoDB manages a relational schema that ensures data integrity and performance.

Additional System Layers and Technical Architecture:
The **Medical Center Management System** is designed with a modular, scalable architecture that supports security, integration, monitoring, and a responsive user experience.

Layers of the System:

Security Layer: Role-based access control is implemented using Spring Security, guaranteeing that only authorized users such as physicians, pharmacists, lab assistants, and receptionists can access particular modules in accordance with their roles.

Integration Layer: Ensures smooth interoperability by utilizing RESTful APIs to link external third-party services (such as lab equipment interfaces, insurance providers, and SMS gateways) with internal system modules.

Monitoring Layer: Enables early detection and resolution of system problems by continuously tracking system performance, logging faults, and supporting real-time monitoring.

Database Design: To remove duplication and guarantee data integrity, the database schema is completely normalized. For effective access and updates, entity relationships between users, appointments, lab reports, prescriptions, inventory, and suppliers are meticulousl-designed.

Listings and Alerts: Help with efficiency monitoring and audit log recording. Stored Procedures: Manage intricate database-level transactions safely and effectively.

User Interface Design: To guarantee responsiveness on all device sizes, the front-end is created using CSS Grid and Flex box in accordance with mobile-first principles. Designed using a set of reusable components to provide uniformity throughout the platform. Complies with accessibility and branding guidelines to provide a welcoming user-experience.

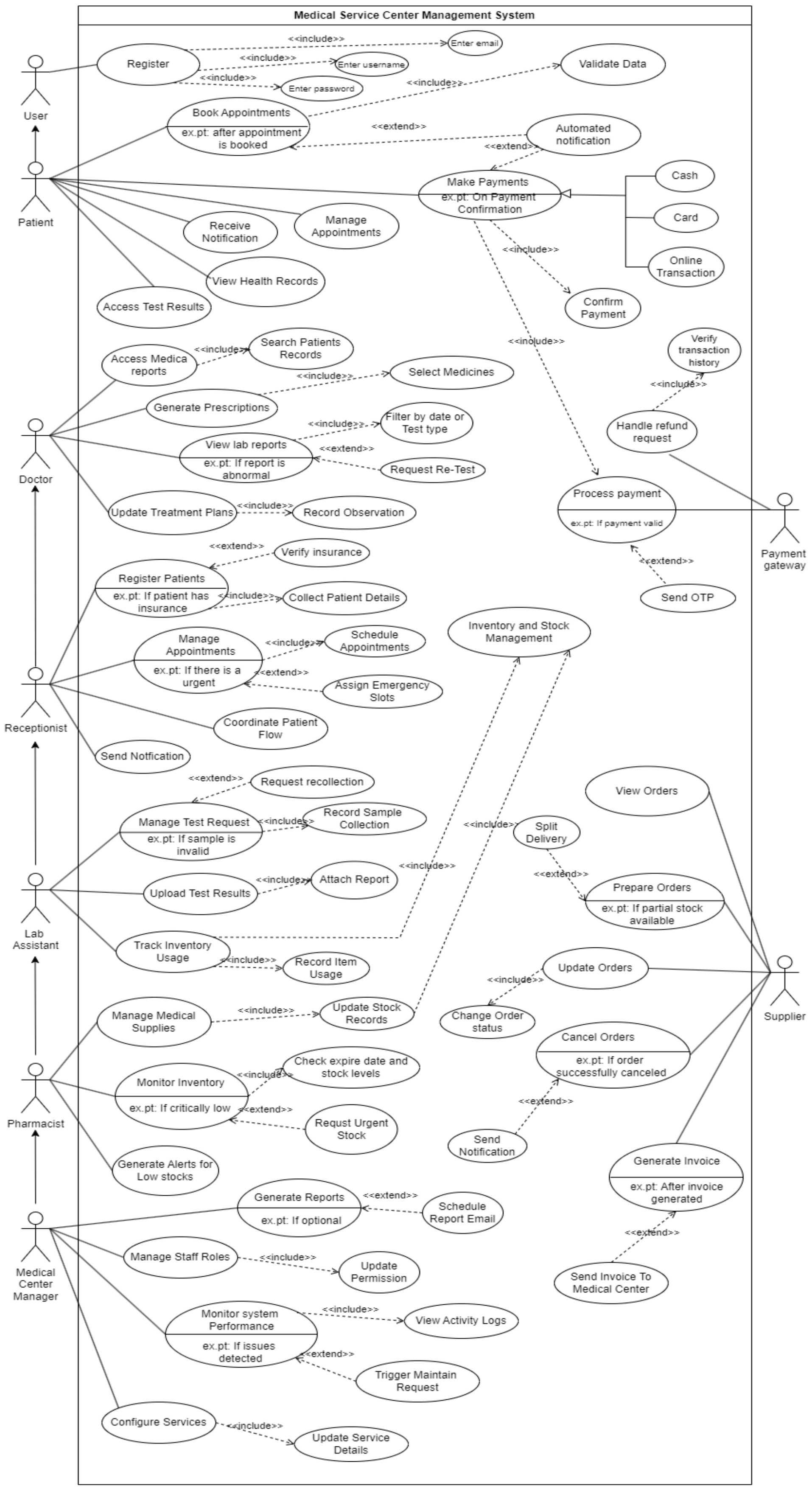
Workflows for users are well defined, with instructions for:

- Onboarding
- Error handling
- User feedback collection

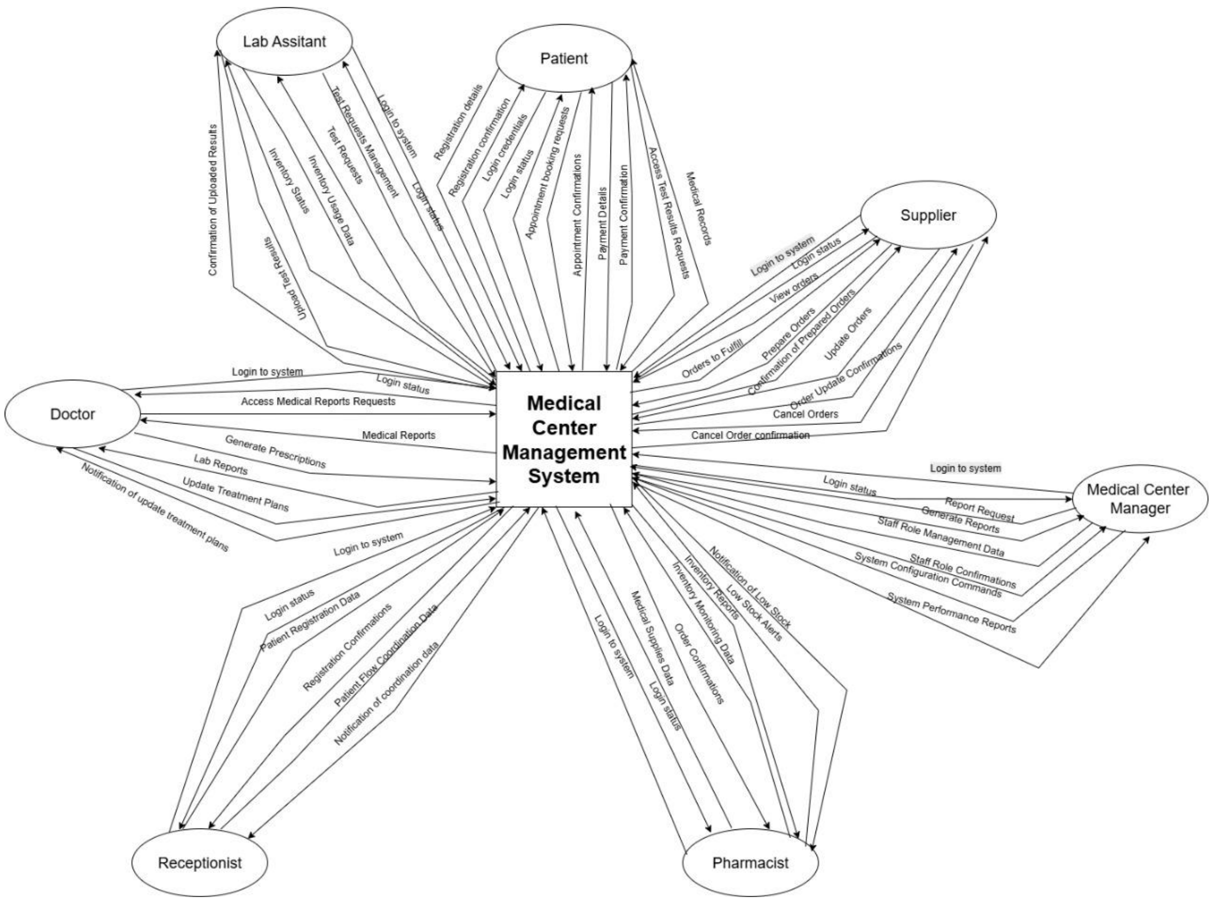
Integration and API Capabilities: Proper versioning, security protocols, error management, and thorough developer documentation are all part of a RESTful API design. Import/export tools and web hooks facilitate smooth interaction with outside services such as labs, delivery providers, and payment gateways.

09.2. Project Management Documentation

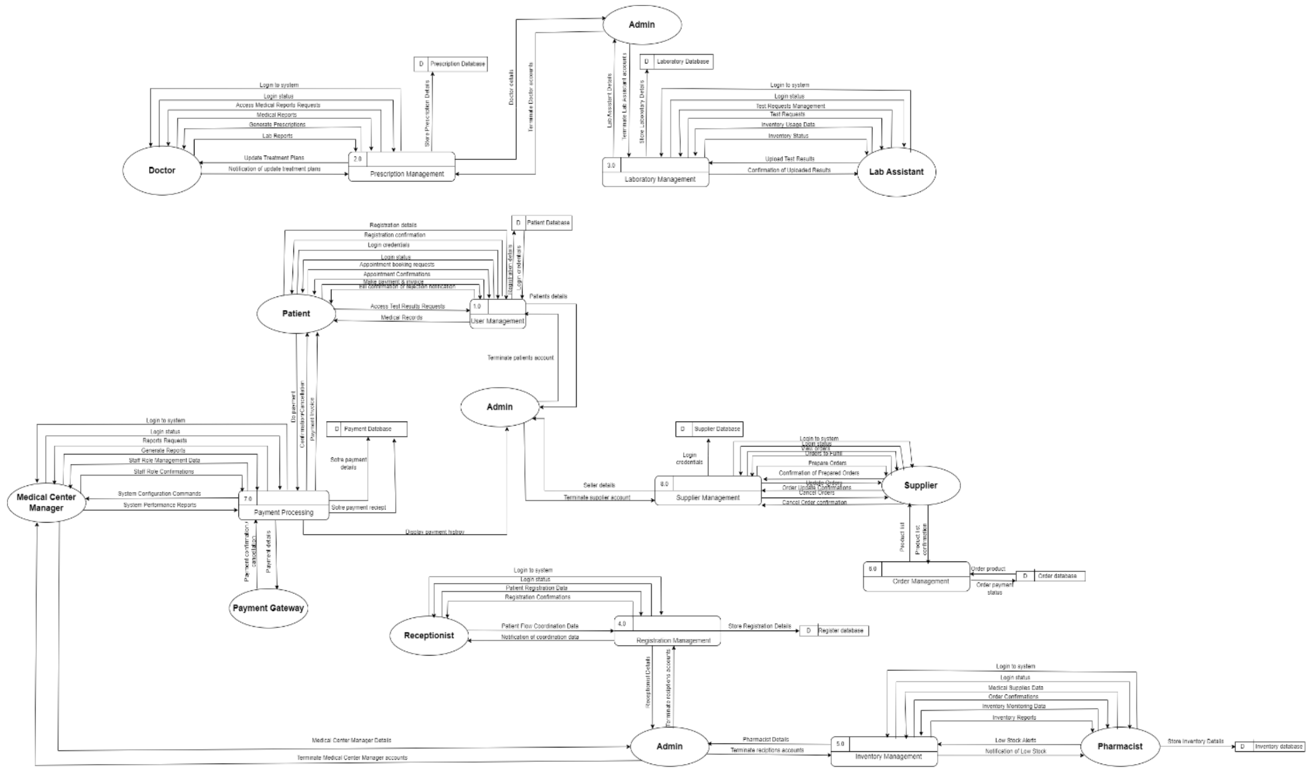
9.2.1. Use case diagram – Use Case Diagram.png



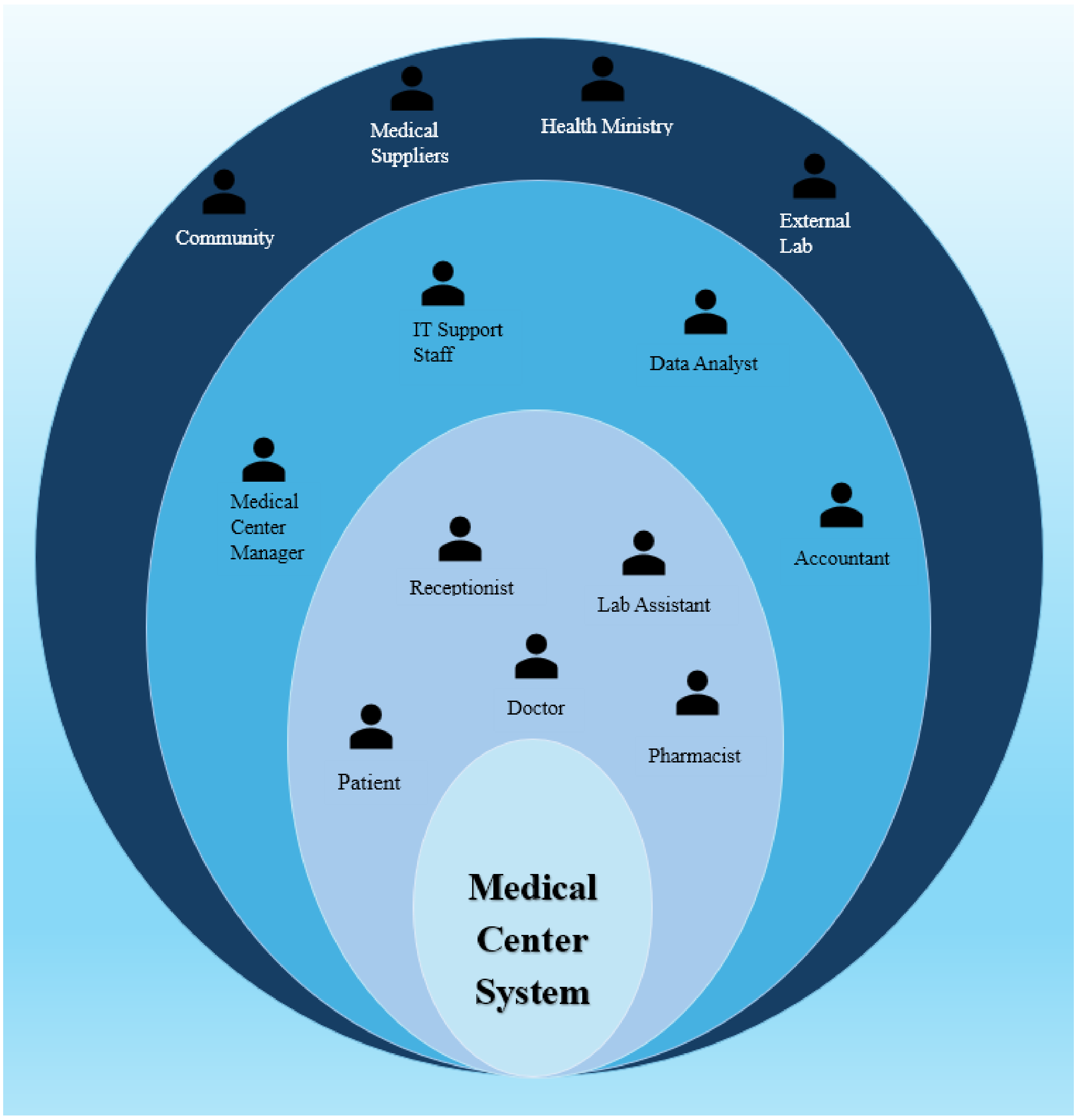
9.2.2. DFD Level 0 – DFD level 0.jpeg



9.2.3. DFD Level 1 – DFD level 1.png



9.2.4. Onion diagram – Onion.png



Medical Suppliers
Health Ministry
Community
External Lab
IT Support Staff
Data Analyst
Medical Center Manager
Accountant
Receptionist
Lab Assistant
Doctor
Pharmacist
Patient
Medical Center System