IT2080 IT PROJECT

Activity 02

Group Details

Campus: SLIIT Campus Malabe

Group Number - ITP25_WE_B01_01_204

Topic - Medical Center Management System

Date of submission – 16/08/2025

Team Members

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

Contents

Introduction

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 mid-sized private clinics, often provide a range of services including general consultations, 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 However, many centers in Sri Lanka still rely on paper-based records or outdated standalone software systems.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 center 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

Project Overview

MediCura offers an end-to-end healthcare management solution that unites patients, physicians, and service providers within a single platform. Built on MongoDB, Express.js, React.js, and Node.js, it combines the best of modern technology with a focus on efficiency, reliability, and simplicity. The platform supports both patient-confronting services, such as scheduling appointments, accessing health records, and receiving test results as well as back-office functions, such as inventory management, assigning staff roles, and managing suppliers. Branches can function autonomously but still be completely integrated with the core system, updating real-time, accurate information throughout the network.

Key Functional Modules:

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 an available doctor – adding doctors and managing them
- Patients can view, edit, or cancel appointments
- Manage their profile and track medical records

2) **Laboratory Management**

- Check the patient's test request and identify required lab tests
- Track the test's 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)
- Booking submission and confirmation
- Automatic price calculation based on rental time
- Supplier information management
- Inventory usage logs and history tracking
- Barcode/QR code generation for labeling parts (using external API)

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)

Advanced Features:

- Role-based access control (administrator, doctor, lab assistant, pharmacist, receptionist, patient, supplier)
- Automated notifications for appointment updates, test results, and stock alerts
- Email/SMS integration for confirmations, reminders, and urgent alerts
- Advanced analytics and reporting for patient trends, inventory usage, and operational performance
- Real-time appointment scheduling with conflict detection and availability tracking
- Integrated health record management with secure patient data sharing
- Inventory and supply chain tracking with low-stock alerts and supplier coordination
- Multi-branch management with synchronized data across all locations
- Support for multiple languages (optional) to enhance accessibility
- Data security and compliance tools for patient confidentiality and regulatory requirements

System Benefits:

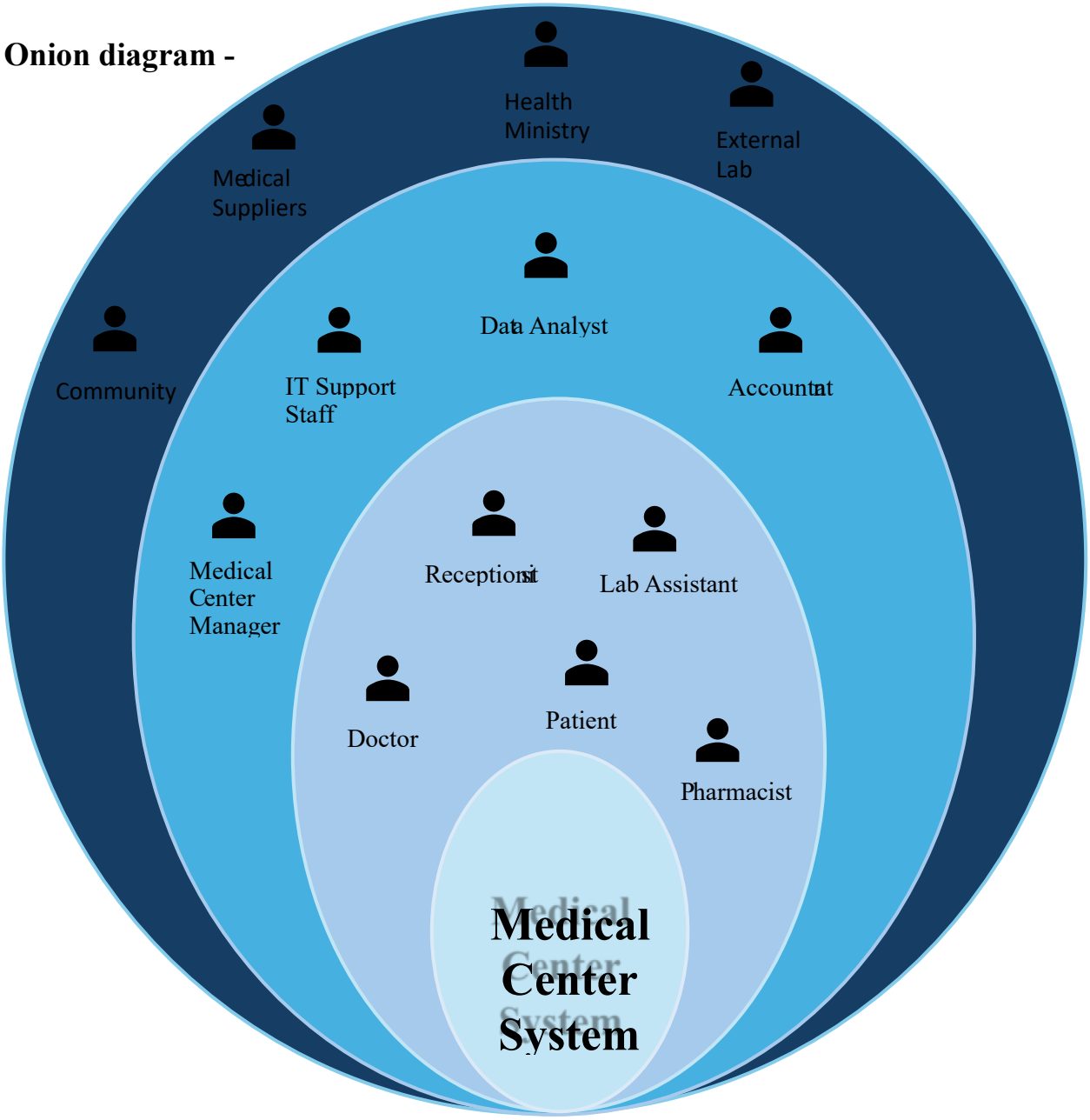
- Accessible 24/7 from any device for patients, staff, and administrators
- Scalable platform that can support multiple branches, clinics, or hospital networks
- Reduced operational costs through automation of scheduling, inventory, and communication
- Improved patient satisfaction with faster, easier appointment booking and results delivery
- Increased efficiency for medical staff through centralized data and real-time updates
- Enhanced revenue opportunities by optimizing appointment utilization and service offerings
- Better decision-making with analytics-driven insights and reports
- Streamlined supplier and inventory management to reduce waste and shortages

Conclusion:

MediCura is a revolutionary healthcare management system. It's a place where patients can easily acquire medical facilities, and doctors and medical practitioners can manage appointments, files, and resources efficiently, providing everyone with a hassle-free and secure experience.

With the convergence of technology, design, and user-centric functionality, MediCura powers the healthcare industry with efficiency, growth, and improved patient care utilities, paving the way for a new generation of medical services in the digital era.

Onion diagram -



https://1drv.ms/i/c/908584e38cbc242a/ER11BH4mpEpOpAkrNFxMnC4BayRSgtmiy-9hClx2u4bOcg?e=jfapY8

Functional Requirements

1. Patient Functional Requirements

- Patient Portal – Sign in to the patient portal to access personal medical information and services.
- Book Appointments – Schedule, reschedule, or cancel medical appointments online.
- View Medical Records – Access past prescriptions, test results, and treatment history.
- Order Medicines Online – Request home delivery or branch pickup of prescribed medicines.
- Make Payments – Pay for consultations, treatments, or medicines online.
- Provide Feedback – Share your experiences and suggestions about the services you received.

Track Appointment Status – Check confirmation, pending, or completed status of bookings.

2. Doctor Functional Requirements

- Doctor Portal – Sign in to the doctor portal to access schedules, patient lists, and medical tools.
- Manage Appointments – View, accept, reschedule, or cancel patient appointments.
- Access Patient Records – Review patient history, prescriptions, and test results.
- Update Medical Records – Add diagnoses, treatment plans, and progress notes.
- Prescribe Medicines – Create and send prescriptions to the pharmacy system.
- View Test Results – Access lab reports and imaging results linked to patients.
- Communicate with Patients – Send messages or updates through the patient portal.
- Generate Medical Reports – Create summaries or certificates for patients as needed.

3. Lab Assistant Functional Requirements

- Lab Assistant Portal – Sign in to the lab portal to view assigned tasks and patient test requests.
- Manage Test Requests – Receive, update, and confirm laboratory test orders.
- Record Test Results – Enter test findings into the system for doctor review.
- Update Test Status – Mark tests as pending, in progress, or completed.
- Manage Lab Inventory – Track and update stock of lab equipment and supplies.
- Generate Lab Reports – Create and upload detailed test reports for patients.
- Communicate with Doctors – Share urgent results or clarifications directly with doctors.

4. Pharmacist Functional Requirements

- Pharmacist Portal – Sign in to the pharmacist portal to access medicine inventory and prescriptions.
- Manage Medicine Inventory – Add, edit, or remove medicines and update stock levels.
- Process Prescriptions – Review and prepare medicines based on doctor prescriptions.
- Update Prescription Status – Mark prescriptions as pending, ready for pickup, or delivered.
- Generate Invoices – Create billing details for medicines provided to patients.
- Order Medicines from Suppliers – Request stock replenishment when items are low.
- Track Medicine Expiry Dates – Monitor and remove expired medicines from inventory.

5. Receptionist Functional Requirements

- Receptionist Portal – Sign in to the receptionist portal to manage front-desk operations.
- Manage Appointments – Book, reschedule, or cancel patient appointments.
- Process Walk-in Patients – Register new patients and enter their details into the system.
- Verify Payments – Check and confirm payment status before or after appointments.
- Update Patient Information – Edit personal or contact details when required.
- Direct Patient Queries – Respond to questions and guide patients to the right department.
- Generate Daily Schedules – Prepare and print daily appointment lists for doctors.

NFRs and analyze user-wise

1) Patient

- Performance: The system must load appointment schedules and medical records quickly to avoid wait times.
- Usability: The interface should be simple and user-friendly, allowing patients to book appointments and view their health information easily.
- Reliability: The system should be available 99.9% of the time to ensure patients can access services whenever needed.
- Security: Strong authentication and data encryption must protect patients' personal and medical information.
- Scalability: The system should handle many patients by accessing their records and booking appointments simultaneously without delays.

2) Doctor

- **Performance:** The system must load patient data quickly (within seconds) to avoid delays during consultations.
- **Usability:** The interface should be simple and easy to use, enabling doctors to access information with minimal effort.
- **Reliability**: High system availability (99.9%) is essential to ensure uninterrupted access during working hours.
- **Security:** Strong authentication, data encryption, and audit logging are needed to protect patient privacy.
- **Scalability:** The system should support many doctors using it at the same time without slowing down.

3) Lab Assistant

- **Accuracy:** The system must ensure that all test results and data entries are accurate and validated to prevent errors.
- **Traceability**: Every action (e.g., data entry, test updates) should be logged to maintain an audit trail for accountability.
- **Data Integrity**: The system must prevent unauthorized changes and ensure lab results remain consistent and trustworthy.
- **Notification**: Lab assistants should receive timely alerts about new test requests or abnormal results requiring attention.

- **Backup & Recovery:** Regular data backups must be maintained to protect lab data and enable quick recovery in case of failure.

4) Pharmacist

- **Inventory Accuracy:** The system must maintain precise real-time tracking of medicine stock levels to prevent shortages or overstock.
- **Expiry Management:** Automatically flag and alert pharmacists to medicines nearing expiry to ensure safety.
- **Prescription Validation:** The system should validate prescriptions against drug interactions and dosage guidelines to prevent errors.
- **Audit Trail:** Record all medicine dispensation and inventory changes for compliance and accountability.
- **Notification:** Provide timely alerts for low stock, new prescriptions, or urgent refill requests.

5) Receptionist

- **Response Time:** The system must quickly process patient check-ins and appointment scheduling to minimize waiting times.
- **User-Friendly Interface:** The interface should be simple and intuitive for quick data entry and retrieval.
- **Data Accuracy:** Ensure patient information entered is validated to reduce errors.
- **Privacy:** Patient data accessed by receptionists must be limited to what is necessary, protecting sensitive medical information.
- System Availability: The system should be operational during all working hours to handle continuous patient flow.

6) Supplier

- **Order Accuracy:** The system must ensure purchase orders are correctly generated and processed to avoid errors.
- **Timely Updates:** Suppliers should receive real-time notifications about order status changes and delivery schedules.
- **Secure Access:** Suppliers must have secure login credentials to protect sensitive procurement information.
- **Reliability:** The system should be consistently available to allow suppliers to update inventory and confirm deliveries without interruptions.
- **Integration**: Seamless integration with the medical center's inventory and payment systems to streamline order fulfillment.

7) Bank/Financial Institutions

- **Security** - Payment data should be secure.
- **Audit Logs** – Every transaction must be recorded in the System.

8) Medical Center Manager

- **Dashboard Performance**: The system should provide real-time, fast-loading reports and analytics for effective decision-making.
- **Access Control:** The manager should have secure, role-based access to sensitive data and management functions.
- **Reliability:** The system must be highly available, ensuring continuous monitoring and management capabilities.
- **Scalability**: The system should handle growing data and user volumes as the medical center expands.
- **Audit and Compliance:** Comprehensive logging and reporting features to track operations and ensure regulatory compliance.

Technical requirements for the system

1) System Architecture

- Web-based client-server architecture.
- RESTful API support for frontend–backend communication.
- Modular and scalable backend.

2) Platform & Technology Stack

- Frontend: React.js
- Backend Framework: Express.js
- Database: MongoDB
- JavaScript Runtime Environment: Node.js

3) Security Requirements

- HTTPS for secure data transfer.
- Role-based access control (RBAC) for admin, patients, doctors, receptionists, and pharmacists.
- Passwords are hashed and stored securely.

JWT or OAuth 2.0 authentication.

4) Performance

- System should support 100+ concurrent users (scalable architecture).
- Frontend loads main pages within 3 seconds on a 4G connection.

5) Availability & Reliability

- 99.5% uptime with failover mechanisms.
- Regular database backups (daily automated backup with weekly full snapshot).
- Real-time monitoring for crashes and error reporting.

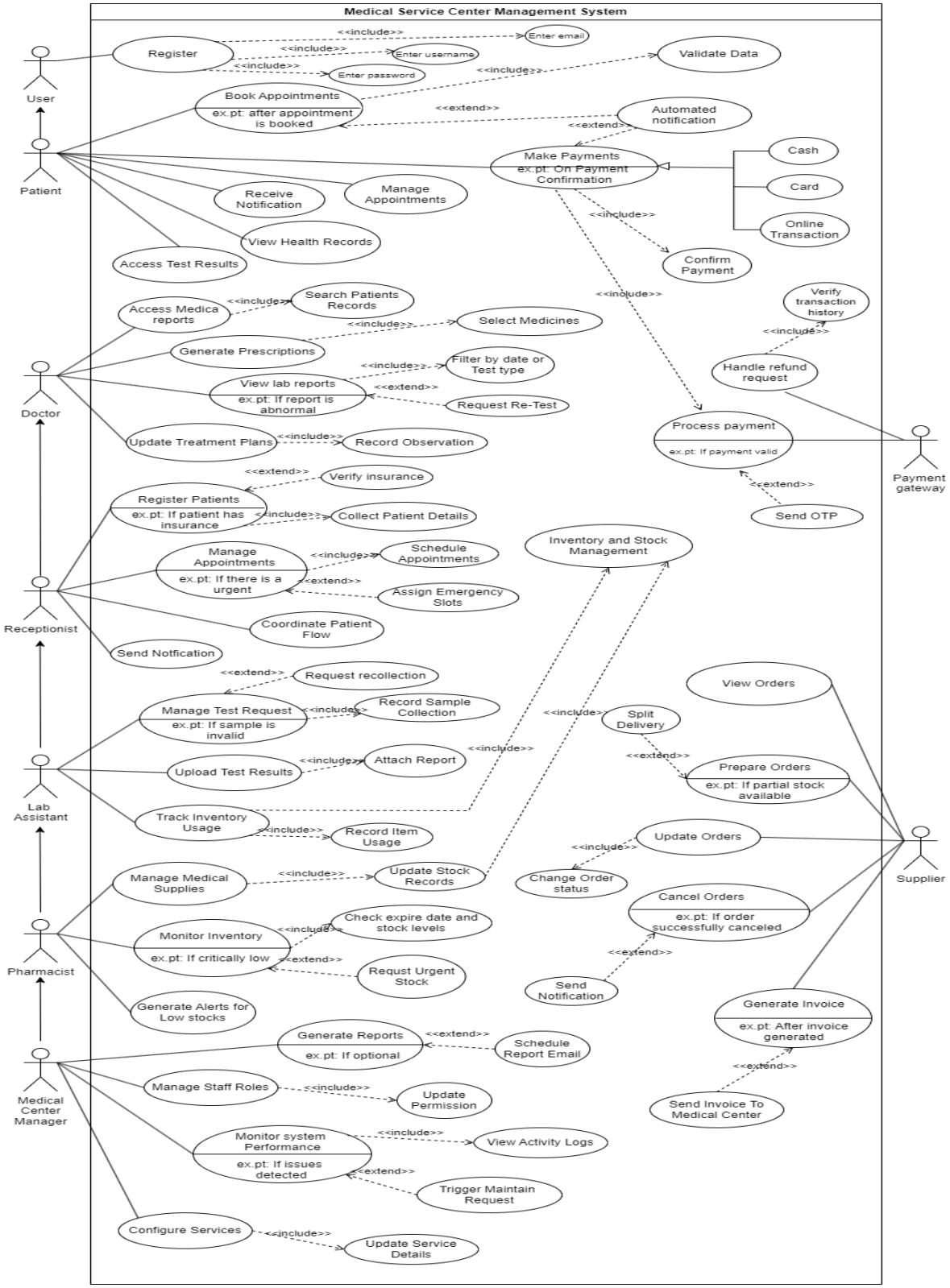
6) Maintainability

- Code should follow standard coding practices and be well-documented.
- Modular codebase for easier updates and debugging.
- Logging system to trace errors and performance metrics.

7) Data Management

- CRUD operations for user management, laboratory management, invoice management, supplier management, inventory management.
- Reporting modules for management (export to Excel/PDF).
- Data retention policies to comply with legal regulations.

Usecase Diagram



Use case diagram -
https://1drv.ms/i/c/908584e38cbc242a/EYQn2GMo4ZKsF4weL_LPI4BLFVNpaeOZgi90EHRRYfLYA?e=CiLG3j

Use case descriptions for the 5 main use cases

1. User registration & login

Use case Name	User registration & Login
Actor	Patient, Doctor, Pharmacist, Lab Assistant, Receptionist, Supplier, Medical Center Manager
Goal	To allow authorized users to register and log in securely to access the medical center system.
Overview	• New users can register by providing the required details. • Existing users can log in using valid credentials. • The system verifies identities, provides role-based access, and ensures data security.
Pre-conditions	• User has a valid email/contact information and role in the medical center. • The system is running and connected to the user database.
Post-conditions	• User is successfully registered or logged in. • User data is securely stored. • Invalid attempts are recorded for security.
Basic path	1) User opens the registration or login page. 2) For registration, user enters the required information and submits. 3) System validates details and stores them in the database. 4) For login, user enters the username/password. 5) System verifies credentials and grants access based on role.

Alternative path	• Invalid Data: Registration fails if mandatory fields are missing or incorrect. • Duplicate Registration: System prevents using an already registered email/contact. • Invalid Login: System denies access on incorrect username/password and prompts retry.
	• Account Lock: After multiple failed login attempts, the account may be locked or flagged.
NFRs & TRs	NFRs (Non-Functional Requirements) • Performance: Registration and login must complete within 2–3 seconds. • Usability: Interface should be simple and user-friendly. • Security: Strong authentication, encrypted passwords, and audit logs. • Reliability: System must be available 99.9% of the time for access. • Scalability: Support many users logging in at the same time without slowdown. TRs (Technical Requirements) • Secure password hashing • Role-based access control for different user types. • Database for storing user credentials securely. • SSL/TLS for secure data transmission. • CAPTCHA or OTP for additional verification if needed.

2. Check the patient's test request and identify required lab tests

Use case Name	Check the patient's test request and identify required lab tests
Actor	Lab Assistant
Goal	To verify a patient's test request and determine which lab tests need to be performed.
Overview	• The lab assistant checks for requests for tests from doctors. • The system helps decide what laboratory tests are required, eliminating incorrect and duplicate tests, and giving accurate results.
Pre-conditions	• The patient test request is made by an authorized doctor. • The system is functional and connected to the database of patient records.
Post-conditions	• The required lab tests are confirmed and marked for processing. • Any discrepancies (missing or unclear tests) are flagged for review.
Basic path	• Lab assistant logs in to the system. • Opens pending test requests. • Reviews details of the patient request. • Confirms required lab tests from the list. • Marks tests as approved for processing.
Alternative path	• Incomplete Request: If information is missing, the system alerts the lab assistant to contact the doctor. • Duplicate Request: The system flags repeated or overlapping test orders. • Invalid Patient ID: The request is rejected if patient details are incorrect.

NFRs & TRs	NFRs (Non-Functional Requirements) • Performance: Requests and lab test data must load within 2 seconds. • Accuracy: The system must prevent errors in identifying required tests. • Usability: The Interface should clearly display test requests and details. • Security: Patient data must be encrypted and accessible only to authorized staff. • Reliability: The system should be available 24/7 to handle urgent test requests. TRs (Technical Requirements) • Secure access with role-based authentication. • Integration with electronic medical records to retrieve patient data. • Automatic alerts for missing or duplicating information. • Audit logs for all actions taken by lab assistants.

3. Component operations for medicines

Use case Name	**Component operations for medicines**
Actor	Pharmacist
Goal	To efficiently manage medicine records — adding, updating, deleting, viewing, and tracking medicines in the inventory.
Overview	This function maintains accurate and up-to-date medicine information, ensuring correct stock levels, timely alerts for expiry or low stock, and smooth inventory management.
Pre-conditions	• The pharmacist is authenticated and authorized to access the medicine inventory. • The medicine database is connected and operational.

Post-conditions	• Medicine records are successfully updated (added, modified, deleted, or viewed). • Inventory levels are accurate and reflect real-time status. • Alerts for low stock or expiry are generated automatically if applicable.
Basic path	1. Pharmacist logs in to the system. 2. Opens the medicine inventory module. 3. Selects an action: Add, Update, Delete, or View medicines. 4. System processes the request and updates inventory records. 5. Confirmation is displayed to the pharmacist.
Alternative path	• Invalid Data: If incomplete or incorrect medicine details are entered, the system prompts correction. • Duplicate Records: The System prevents adding a medicine batch already in stock. • Unauthorized Access: If a non-pharmacist attempts to access, the system denies entry. • Database Error: If the system is offline, the operation is queued or rejected with an alert.
NFRs & TRs	NFRs (Non-Functional Requirements) • Performance: All inventory operations must complete within 2–3 seconds. • Usability: The interface should be clear and user-friendly for quick medicine management. • Security: Only authorized staff should modify inventory; data must be encrypted.

	• Reliability: The System should be available 99.9% of the time to avoid medicine stock issues. • Accuracy: Real-time stock levels must always match physical inventory. • Scalability: Capable of handling large medicine databases without slowing down. TRs (Technical Requirements) • Secure database connection with role-based authentication. • Automatic validation to prevent duplicate or incorrect records. • Real-time inventory tracking with automated low-stock and expiry alerts. • Audit logs for all inventory operations (who changed what and when). • Backup and recovery to protect against data loss.

4. Automatic purchase order generation

Use case Name	Automatic purchase order generation
Actor	System (automatic process) Pharmacist / Medical Center Manager (for monitoring/confirmation)
Goal	To automatically generate purchase orders when stock levels fall below minimum thresholds, ensuring timely replenishment.
Overview	• The system monitors inventory continuously. • When stock for a medicine or equipment drops below the predefined threshold, a purchase order is created automatically and sent to the supplier.
Pre-conditions	• Minimum stock levels must be configured. • Supplier information must be available in the system. • Inventory tracking is active and up-to-date.
Post-conditions	• Purchase order is created and sent to the supplier.

	• Notifications sent to relevant staff (pharmacist/manager). • Inventory replenishment process is initiated.
Basic path	1. System monitors stock levels. 2. Stock falls below the reorder point. 3. The system generates a purchase order with the required items and quantities. 4. The order is sent to the supplier. 5. The pharmacist/manager receives notification and can review if needed.
Alternative path	• Supplier details missing → system alerts pharmacist to manually assign supplier. • Auto-ordering disabled → pharmacist is notified to create a manual order. • Multiple low-stock items → system consolidates into a single order per supplier.
NFRs & TRs	NFRs (Non-Functional Requirements) • Performance: Orders generated immediately upon low stock detection. • Accuracy: Correct items, quantities, and supplier information • Reliability: Automatic ordering works consistently. • Security: Only authorized staff can view or modify autoorder settings. • Auditability: All auto-generated orders are logged. TRs (Technical Requirements) • Integration with the inventory module for real-time stock monitoring. • Supplier database with contact and ordering information. • Notification system for alerts to staff. • Logging mechanism for auditing orders. • Option to enable/disable auto-order per item.

6. Transaction recording and tracking

Use case Name	Transaction recording and tracking
Actor	Pharmacist (primary) Supplier (secondary, for order/payment updates) Medical Center Manager
Goal	To accurately record all financial and inventory transactions and enable real-time tracking for auditing, reporting, and decision-making.
Overview	• Every transaction, including medicine orders, deliveries, payments, and stock updates, is automatically recorded in the system. • Users can view transaction history, track ongoing processes, and generate reports. • The system provides transparency, traceability, and reliability in the medical center's operations.
Pre-conditions	• Users must be authenticated and authorized. • Inventory and supplier databases must be connected and up to date. • The system must be operational.
Post-conditions	• Transaction is recorded and stored in the system permanently. • Real-time updates reflect in inventory and financial records. • Auditable history is available for review.
Basic path	• User performs a transaction (e.g., placing an order, receiving a delivery, processing payment). • System automatically records transaction details: ID, date/time, items, quantities, amounts, user, and status. • Inventory and financial records are updated in real-time. • Notifications or confirmations are sent to relevant users (pharmacist, manager, supplier). • Transaction is stored for future auditing and reporting.

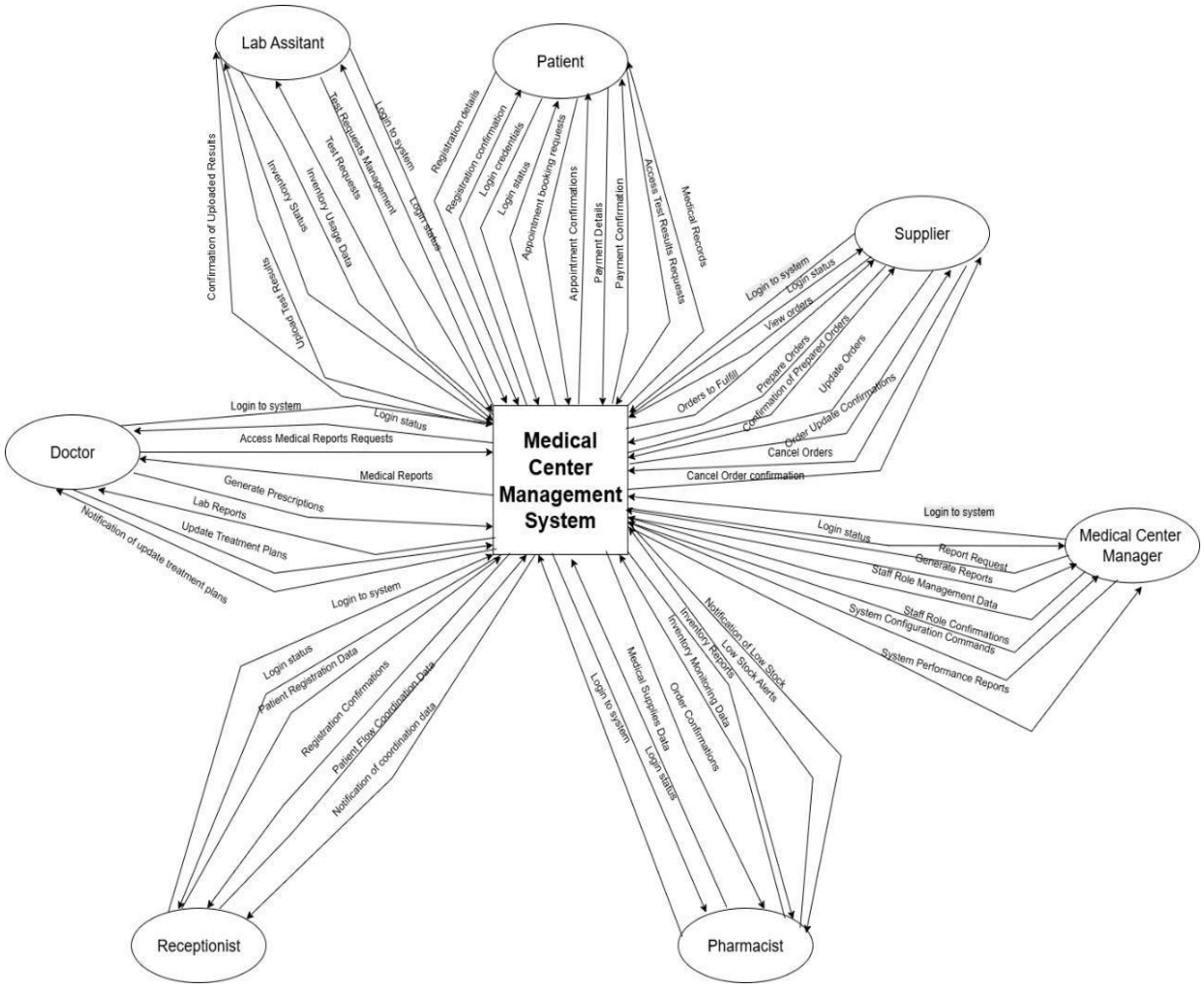
Alternative path	• Invalid Transaction: If data is missing or incorrect, system prompts user to correct it. • System Failure: If system is offline, transaction is queued and processed once available. • Unauthorized Action: If an unauthorized user attempts a transaction, the system denies it and logs the attempt.
	• Partial Delivery or Payment: System updates transaction with partial status and pending items/amounts.
NFRs & TRs	NFRs (Non-Functional Requirements) • Accuracy: All transactions must be correctly recorded with no duplication or loss. • Security: Role-based access, encrypted data storage, and secure transmission. • Performance: Real-time recording and retrieval of transactions. • Reliability: The System must be highly available (99.9%) to ensure uninterrupted operation. • Auditability: Every action is logged with timestamps and user IDs. TRs (Technical Requirements) • Secure database for storing all transactions. • Integration with inventory and supplier modules. • Notification system for transaction updates. • Logging mechanism for audit trails. • Reporting tools for exporting transaction history. • Backup and recovery to protect against data loss.

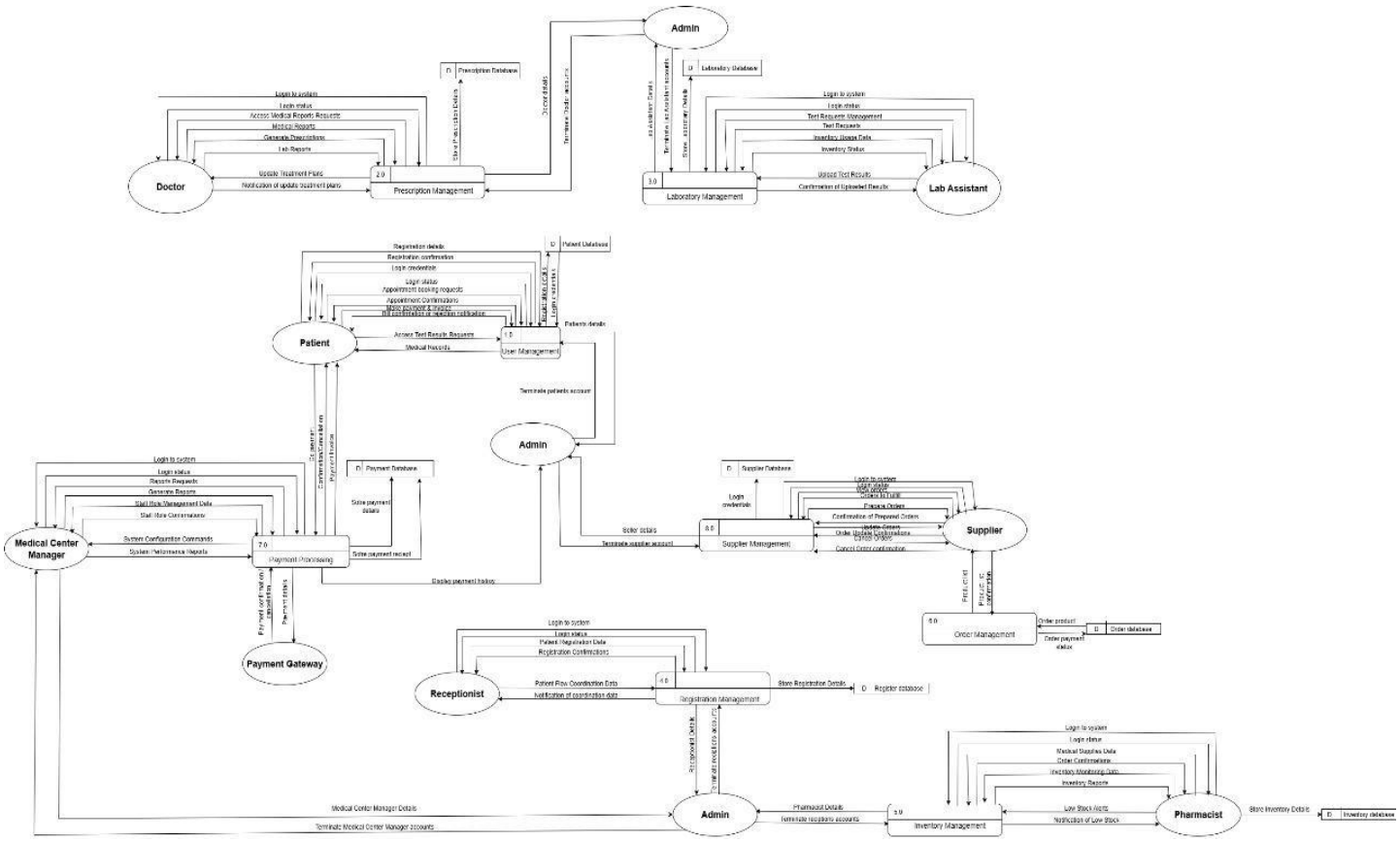
Develop suitable diagram to show a visual

.

,

1) **2) DFD level 1 diagram -** https://1drv.ms/i/c/908584e38cbc242a/EWA9KJAyz_xGqeEOATGD3mMBDIVPhLEP0Tc6R2PZdyp5xg?e=z5nlhB





Onion diagram - [DFD level 1.png]

Plan to develop the project as a team.

I. Project Planning and Requirement Analysis

- **Goal:** Define scope, roles, and timeline.
- **Activities:**
  - Gather requirements (appointments, patient records, inventory, billing, supplier management, etc.).
  - Prioritize features (MVP first: patient management, appointments, inventory).
  - Define user roles (doctor, lab assistant, pharmacist, receptionist, admin, patient).
  - Create use-case diagrams and workflows.
  - Allocate team roles:

    Frontend developers (React.js UI/UX)
    Backend developers (API, business logic)
    Database developers (MongoDB schemas, queries) **QA/testers** (manual + automated testing) **Project lead** (integration + code reviews).

II. System Architecture and Tech Setup

- **Goal:** Create a consistent foundation.
- **Activities:**
  - Decide on **MERN stack** structure:

    Frontend (React.js) → responsive components using Tailwind or Material UI.
    Backend (Node.js with Express) → RESTful API for all features.
    Database (MongoDB) → structured collections for patients, staff, appointments, inventory, suppliers, etc.

  - Plan authentication: JWT (JSON Web Tokens) for secure login.
  - Plan API routes and endpoints (e.g., /api/patients, /api/appointments).
  - Set up GitHub repo with **branching strategy** (main/dev/feature branches).
  - Configure ESLint + Prettier for consistent code formatting.

III. Database Design (MongoDB)

- **Goal:** Define schema for each module.
- **Activities:**
  - Collections:

Users (role-based): name, email, password (hashed), role, contact info.
Patients: medical history, appointments, prescriptions.
Staff: doctors, lab assistants, pharmacists, receptionists.
Inventory: medicines, stock levels, supplier ID.
Suppliers: company details, order history.
Appointments: patient ID, doctor ID, date, status.
Transactions: invoices, order logs.

- Implement indexes for fast searching (e.g., patient name, appointment date).
- Use **Mongoose ODM** to define a schema with validation.

IV. Backend Development (Node.js + Express)

Goal: Build secure and scalable APIs.
Activities:

- Develop APIs:
  - /api/users → manage staff and patient accounts.
  - /api/appointments → booking, rescheduling, cancellation.
  - /api/inventory → CRUD for medicines and equipment.
  - /api/suppliers → order generation, order history.
  - /api/transactions → invoice and payment logs.
- Apply **role-based access control (RBAC)** middleware.
- Add input validation.
- Add error handling and logging.
- Test endpoints with **Postman** before frontend integration.

V. Frontend Development (React.js)

Goal: Build interactive, role-specific dashboards.
Activities:

- Set up project structure with Vite or Create React App.
- Install libraries:
  - **Axios** for API calls.
  - **React Router** for navigation.
  - **Redux Toolkit or Context API** for state management.
  - **Tailwind CSS or Material UI** for UI components.
- Create components:
  - Login & Registration pages.
  - Patient dashboard (appointments, prescriptions).
  - Doctor dashboard (patient list, consultations).

Inventory management screens.
Supplier/order management screens.
Admin dashboard (user roles, reports).

- Connect to backend APIs via Axios.
- Implement responsive design.
- Add form validation and loading indicators.

VI. Integration and Testing

Goal: Ensure the full system works as one unit.
Activities:

- Connect frontend with backend APIs.
- Verify CRUD operations for all modules.
- Perform unit tests (Jest/Mocha for backend, React Testing Library for frontend).
- Test authentication flows with different roles.
- Check database consistency after operations.
- Conduct usability testing with sample users. Fix bugs and optimize performance.

VII.Deployment and Documentation

Goal: Make the system live and maintainable.
Activities:

- Deploy frontend → Netlify or Vercel.
- Deploy backend → Render, Heroku, or AWS EC2.
- Deploy database → MongoDB Atlas (cloud).
- Add **environment variables** for security (API keys, DB connection).
- Write documentation:
  API reference (endpoints, input/output).
  User manual for each role.
  System installation guide for developers.
- Create future enhancement list (e.g., online payment, analytics, AI chatbots)

VIII.Project Management and Timeline

Use **Agile Scrum methodology**:

- **Sprints of 2 weeks** with defined deliverables.
- **Sprint review/demo** at the end of each cycle. **Sprint retrospective** to improve teamwork.

• Tools:

- **Trello/Jira** for task tracking.
- **GitHub Projects** for code management. **Slack/Discord** for communication.

User Management	K.H.S. Dinsara
Supplier Management	Navodya A.K.
Laboratory Management	Thathsarani J.R.
Inventory Management	Liyanaarachchi L.A.D.A.
Billing & Payment Management	Ekanayaka E.W.I.D.