IT 2080 – IT Project

Assignment 02

Group Number - ITP25_WE_B01_01_204

Topic - Medical Center Management System

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
IT23669758	Liyanaarachchi L.A.D.A	it23669758@my.sliit.lk	0772712630
IT23810464	Ekanayaka E.W.I.D	it23810464@my.sliit.lk	0768370886

Contents

01.Progress

Our Smart Medical Center Management System has achieved significant progress, with most of the core functionalities successfully implemented and seamlessly integrated. The system is built using a modern three-tier architecture, ensuring scalability, maintainability, and efficient performance across all modules.

The presentation layer (frontend) is developed using React.js, providing a dynamic, responsive, and user-friendly interface for both patients and medical staff. React’s component-based structure allows efficient rendering and easy maintenance of UI components such as appointment booking forms, patient dashboards, and medical records displays.

The application layer (backend) is powered by Node.js and Express.js, which handle the core business logic and communication between the frontend and database. This layer manages operations like user authentication, appointment scheduling, data validation, and secure API endpoints for data transfer. The backend architecture ensures high performance and asynchronous processing, which is crucial for handling multiple user requests in real-time.

The data layer (database) uses MongoDB, a NoSQL database known for its flexibility and scalability. MongoDB stores data in a JSON-like format, making it ideal for managing diverse healthcare-related data such as patient profiles, doctor details, prescriptions, and feedback. Its schema-less design allows the system to evolve easily as new modules and data types are introduced.

Overall, this three-tier architecture ensures a clear separation of concerns, making the system more secure, modular, and easy to update. The combination of React.js, Node.js, and MongoDB forms a powerful MERN stack, allowing smooth integration, rapid development, and strong performance suitable for modern healthcare management needs.

User Management

User Story	Completed (Yes/ No)	Comment	Member
As a user, I want to register to the system, so I can access the system	Yes	API and UI done. validation & error messages added.	
As a user, I want to log in securely, so I can access my account safely	Yes	Node Security auth wired. login UI done.	
As an admin, I want to add doctors to frontend, so I can control all doctors.	Yes	UI and Endpoint is implemented	
As an admin, I can get a chart of all of the booking appointments.	No	UI implemented and Endpoint is not implemented	Dinsara K.H.S (IT23588714)
As a user, I want to log my account role-base, so I can access features specific to my role	Yes	API and UI done.	
As a user, I want to update my profile, so I can keep my information current	Yes	API and UI done. error messages added.	
As a user, I want to logout my account, so I can remove my data if I no longer use the system	Yes	API and UI done. error messages added.	
As an admin, I want to manage staff, so I can control system access	Yes	UI and Endpoint is implemented	
As a user, I want password hashing and safe storage, so I can ensure my credentials are protected	Yes	BCrypt PasswordEncoder configured, user passwords stored as hash	
As an admin, I want to view all appointments in the system, so I can monitor and manage appointments.	Yes	UI and Endpoint is implemented	Dinsara K.H.S (IT23588714)
As an admin, I want to search and filter appointments, so I can quickly find specific appointments.	yes	UI and Endpoint is implemented	

As a user, I want to export my appointment data, so I can have a copy of my information	No	UI is done and Endpoint is not implemented	

Laboratory Management

User Story	**Completed (Yes/No)**	**Comment**	**Member**
As a patient, I want to fill a personal details form (including name, test type, preferred date & time) so that I can request a lab test.	Yes	UI and API implemented. Validation added for required fields.	Thathsarani J R [IT23572638]
As a system, I want to validate the submitted form so that incorrect or incomplete details are not sent to the lab assistant.	Yes	Backend validation rules implemented. Error handling messages displayed on UI.	
As a lab assistant, I want to view submitted patient forms so that I can book appointments accurately.	Yes	API and UI completed. Lab assistant dashboard displays pending form submissions.	
As a lab assistant, I want to create lab appointments based on the patient's submitted details so that tests are scheduled properly.	Yes	Appointment creation feature functional done. Data linked with patient form details.	
As a lab assistant, I want to edit or update booked appointments so that I can manage schedule changes.	Yes	Edit functionality implemented with proper validation and update confirmation messages.	
As a lab assistant, I want to delete appointments so that I can handle cancellations or mistakes.	Yes	Delete endpoint implemented with confirmation message in UI.	
As a lab assistant, I want to send an E-mail to each patient after booked the lab appointment. So that patients receive confirmation message.	Yes	Implemented using EmailJS API. Sends automatic confirmation email to patient after appointment booking	

As a patient, I want the system to inform me that appointment changes or cancellations must be requested via phone call so that I follow the correct process.	Yes	Informational message displayed after form submission and in appointment details.	
As a patient, I want to use a Medical AI Bot to analyze uploaded lab reports and answer simple medical questions so that patients receive instant insights.	Yes	AI module integrated and trained. still in testing phase for accurate report reading and response generation.	
As a lab assistant, I want to view a daily appointment schedule so that I can manage workload efficiently.	Yes	Schedule view implemented. Daily appointments displayed in appointments page UI.	
As a system, I want to manage list of all booked appointments so that it has an overview of upcoming tests.	Yes	UI designed. export and send as an Email.	
As a lab assistant, I want to receive instant dashboard notifications when a patient submits a lab form so that I can respond quickly.	Yes	Real-time notification panel implemented using event listeners. New submissions trigger alert messages.	
As a lab assistant, I want to use search bars to filter patients and tests by name so that I can find records quickly.	No	Search and filter functionality implemented on the dashboard with instant results using dynamic queries.	

Billing and Payment Management

High level use story/ funtion	Completed (Y/N, comment)	Responsible member(Name)
1. As a staff member, I want to generate patient invoices so that billing is accurate.	Y – 95% completed. CRUD, graph, PayHere payment integration and Reporting fully functional some UI enhancements pending.	E.W.I.D. Ekanayaka
2. As an admin, I want to track expenses to manage the budget	Y	E.W.I.D. Ekanayaka
3. As HR, I want to manage employee payments	Y	E.W.I.D. Ekanayaka
4. As a user, I want to make online payments through PayHere.	Y	E.W.I.D. Ekanayaka
5. Generate daily, weekly, monthly financial reports.	Y	E.W.I.D. Ekanayaka
6. CRUD operations for expenses, invoices, and employee payments.	Y	E.W.I.D. Ekanayaka
7. Validation to prevent incorrect payments.	Y	E.W.I.D. Ekanayaka
8. Print receipts for invoices.	Y	E.W.I.D. Ekanayaka
9. Integration with patient and employee modules.	N	E.W.I.D. Ekanayaka
10. Filter and search functionality for all records	Y	E.W.I.D. Ekanayaka

Supplier Management

High level use story/ funtion	**Completed (Y/N, comment)**	**Responsible member (Name)**
1. As a Supplier Manager, I want to view all incoming inventory requests so I can decide whether to approve or ignore them	Yes – Feature implemented and tested successfully	Navodya A.K.
2. As a Supplier Manager, I want to approve or ignore requests so that only valid inventory needs are processed.	Yes – Approval and ignore functions work as expected	Navodya A.K.
3. As an Inventory Manager, I want to edit or delete my inventory requests only until the Supplier Manager has taken action, so I can manage my requests responsibly.	Yes – Restrictions properly applied after supplier action	Navodya A.K.
4. As a Supplier Manager, I want to create invoices for approved requests, so that I can formalize transactions.	Yes – Invoice creation tested and functioning correctly	Navodya A.K.
5. As a Supplier Manager, I want to update or delete invoices until the Inventory Manager approves or cancels them, to maintain data accuracy.	**Yes – Update and delete options disabled after final approval**	Navodya A.K.
6. Generate and download invoice details as a PDF, so I can keep records offline.	Yes – PDF generation and download working properly	Navodya A.K.
7. Export all invoice details as a CSV file, so I can analyze them easily.	Yes – CSV export tested and verified successfully	Navodya A.K.
8. As a Supplier Manager, I want to add, view, update, and delete client details, so that my supplier records remain current.	Yes – Client management module completed and validated	Navodya A.K.

Inventory Manager

High level use story/ funtion	**Completed (Y/N, comment)**	**Responsible member (Name)**
1. As inventory manager, I **t**rack current stock levels	Yes – Inventory manager can view stock levels, backend ready, frontend in progress	Liyanaarachchi L.A.D.A
2. As inventory manager, I receive alerts low stock	Yes – Inventory manager should receive alerts when stock falls below 10 medications	Liyanaarachchi L.A.D.A
3. As inventory manager, I add new medication	Yes – Inventory manager can add new stock entries when supplies arrive	Liyanaarachchi L.A.D.A
4. Record stock usage	Yes – Invoice creation tested and functioning correctly	Liyanaarachchi L.A.D.A
5. As inventory manager, I generate reports of inventory, invoices, request	**Yes** – Inventory manager can generate inventory reports, backend in progress	Liyanaarachchi L.A.D.A
6. As inventory manager, I request to supplier	Yes – The inventory manager can place a request from the supplier if the necessary medication is available.	Liyanaarachchi L.A.D.A
7. As inventory manager, I update or delete inventory medication	Yes – Inventory manager can update/delete medication functionality	Liyanaarachchi L.A.D.A
8. As inventory manager, I accept or cancel invoices	Yes – Accepting or canceling invoices related to stock sent by the supplier to the inventory manager	Liyanaarachchi L.A.D.A

02.Test Case

User Management

Test Case ID	Scenario	Steps	Expected Outcome	Status
TC0001	User Registration	1. Navigate to registration page 2. Enter user details (Name, Email/Phone, Password) 3. Click "Register"	Account created, user details saved in database	Pass
TC0002	User Login	1. Navigate to login page 2. Enter valid credentials 3. Click "Login"	User logged in, session stated	Pass
TC0003	Role-Based Login	1. Login with credentials 2. Redirect to dashboard based on role	Role-based features displayed correctly	Pass
TC0004	Update Profile	1. Login to system 2. Navigate to "Profile" 3. Edit details (e.g., address, phone) 4. 4. Save	Profile updated successfully in database	Pass
TC0005	Logout Account	1. Login 2. Navigate to "Logout" 3. Confirm logout	Successfully logout	Pass
TC0006	Admin manage doctors	1. Login as Admin 2. Navigate to "add doctors" 3. Add/edit/delete doctors	Doctor adding, edit and delete successfully done	Pass

TC0007	Password Hashing	1. Register/Login 2. Check stored password in DB	Password stored in hashed format, not plain text	Pass
TC0008	Admin View All appointments	1. Login as Admin 2.Navigate to "Appoitnents" page	All appoitments displayed	Pass
TC0009	Admin View All Contactus messages	2. Login as Admin 3. Navigate to "Contactus messages" page	All messages displayed	Pass
TC0010	Admin Search & Filter appointments	1. Login as Admin 2. Use search/filter by date, role, status	Matching appointments displayed	Pass
TC0011	Export appointments report	1. Login 2. Navigate to "My Appointments" 3. Click "Download"	Report excel download	Pass
TC0012	Admin view appointments charts by according to the booking of doctor specialization	1. Login 2.Navigate to “Report Section” and choose date range and can generate reports and see charts.	Charts are displaying	Ongoing

Laboratory Management

Test case ID	Scenario	Steps	Expected Outcomes	Status
TC001	Submit Patient Details Form	1. Patient navigate laboratory and click "Book Lab Appointment". 2. Enter personal info, test details, preferred date & time and etc. 3. Click "Submit"	Form submitted successfully and stored in database	Pass
TC002	Validate Required Fields	1. Leave required fields empty 2. Click "Submit"	System displays validation error messages	Pass
TC003	View Submitted Forms (Lab Assistant)	1. Login as Lab Assistant 2. Navigate to "New Requests"	All newly submitted patient forms are displayed	Pass
TC004	Book Appointment from Submitted Form	1. Select patient form 2. Enter appointment details 3. Click "New Appointment"	Appointment successfully created and stored in system	Pass
TC005	Edit Booked Appointment	1. Navigate to "Appointments" 2. Click "Edit" on an existing record 3. Modify time/date 4. Save changes	Appointment details updated in database	Pass
TC006	Delete Booked Appointment	1. Navigate to "Appointments" 2. Select appointment 3. Click "Delete" 4. Confirm deletion	Appointment removed from system	Pass
TC007	Receive Appointment Confirmation (Patient)	1. Book an appointment for patient 2. Check patient email or notification area	Confirmation message sent/displayed to patient	Pass
TC008	Send a confirmation email to the patient after scheduled the appointment.	1. Select the scheduled appointment 2. Click the email button and send the email.	Message displayed: "Email sent successfully"	Pass

TC009	View Daily Appointment Schedule (Lab Assistant)	1. Login as Lab Assistant 2. View “Scheduled appointments”	All scheduled appointments displayed	Pass
TC010	Dashboard Notification for New appointment request	1. Submit new patient form 2. Observe lab assistant dashboard	Real-time notification appears for new submission	Pass
TC011	Search Patients or Tests	1. Enter patient/test name in dashboard search bar	Matching records filtered instantly	Ongoing
TC012	Medical AI Bot Report Analysis	1. Upload a lab report 2. Ask simple medical question	AI bot analyzes report and provides summarized response	Pass
TC013	Email Confirmation after Appointment	1. Book lab appointment 2. Check patient email inbox	Confirmation email sent using Emailjs API	Pass

Billing and Payment Management

Test Case ID	Module	Test Scenario	Preconditions	Test Steps	Expected Result
TC01	Patient Invoice	Add new invoice	Patient registered	Fill invoice form & click save	Invoice saved and visible in list
TC02	Patient Invoice	Edit invoice	Invoice exists	Edit fields & click update	Invoice updated
TC03	Patient Invoice	Delete invoice	Invoice exists	Select invoice & delete	Invoice removed
TC04	Patient Invoice	Invalid data entry	Invoice form open	Leave required fields empty	Error message displayed
TC05	Expenses Management	Add new expense	Admin logged in	Fill form & save	Expense saved

TC06	Expenses Management	Edit expense	Expense exists	Edit record & save	Expense updated
TC07	Expenses Management	Delete expense	Expense exists	Select expense & delete	Expense removed
TC08	Employee Payment	Add new payment	Employee registered	Fill form & save	Payment saved
TC09	Employee Payment	Edit payment	Payment exists	Edit record & save	Payment updated
TC10	PayHere Integration	Online payment	Invoice exists	Select invoice → PayHere → Complete payment	Payment successful & invoice marked paid

Supplier Management

Test Case ID	Module	Test Scenario	Preconditions	Test Steps	Expected Result
TC01	Supplier Manager	Approve inventory request	Request sent by Inventory Manager	1. Login as Supplier Manager → 2. View Request management → 3. Click "Approve"	Request status updates to **Approved**
TC02	Supplier Manager	Ignore inventory request	Request sent by Inventory Manager	1. Login as Supplier Manager → 2. View Request management → 3. Click "Ignore"	Request status updates to **Ignored**
TC03	Supplier Manager	Create invoice	Approved request exists	1. View Invoice management → 2. Click "Create invoice" → 3. Create	Invoice successfully created
TC04	Supplier Manager	Update invoice	Invoice not yet approved/cancelled by Inventory Manager	1. View Invoice Management→ 2. View Invoice→ 3. Click "Edit" →	Error message displayed

				4. Click “Update”	
TC05	Supplier Manager	Generate invoice PDF	Invoice exists	1. Click “PDF”	PDF file downloaded successfully
TC06	Supplier Manager	Export invoice CSV	Multiple invoices exist	1. Click “Export CSV”	CSV file downloaded with all invoice data
TC07	Supplier Manager	Manage clients	Supplier Manager logged in	1. Navigate to “Clients” → 2. Add or update client → 3. Save	Client record successfully saved
TC08	Supplier Manager	Export client CSV	Multiple clients exit	1.Click “Export CSV”	CSV file downloaded with all client data

Inventory Management

Test Case ID	Module	Test Scenario	Preconditions	Test Steps	Expected Result
TC01	Inventory Addition	Add new inventory item		Enter valid details → Click "Save"	Item successfully added and visible in list
TC02	Inventory Update	Update inventory quantity, description	Item exists in inventory	Select an item → Update quantity → Save	Quantity updated successfully
TC03	Inventory Deletion	Delete inventory item	Item exists in inventory	Select an item → Click "Delete" → Confirm	Item removed from inventory list
TC04	Inventory Disposal	Dispose inventory item	Item exists in inventory	Select item → Enter reason → Approve disposal	Disposal record created, stock updated

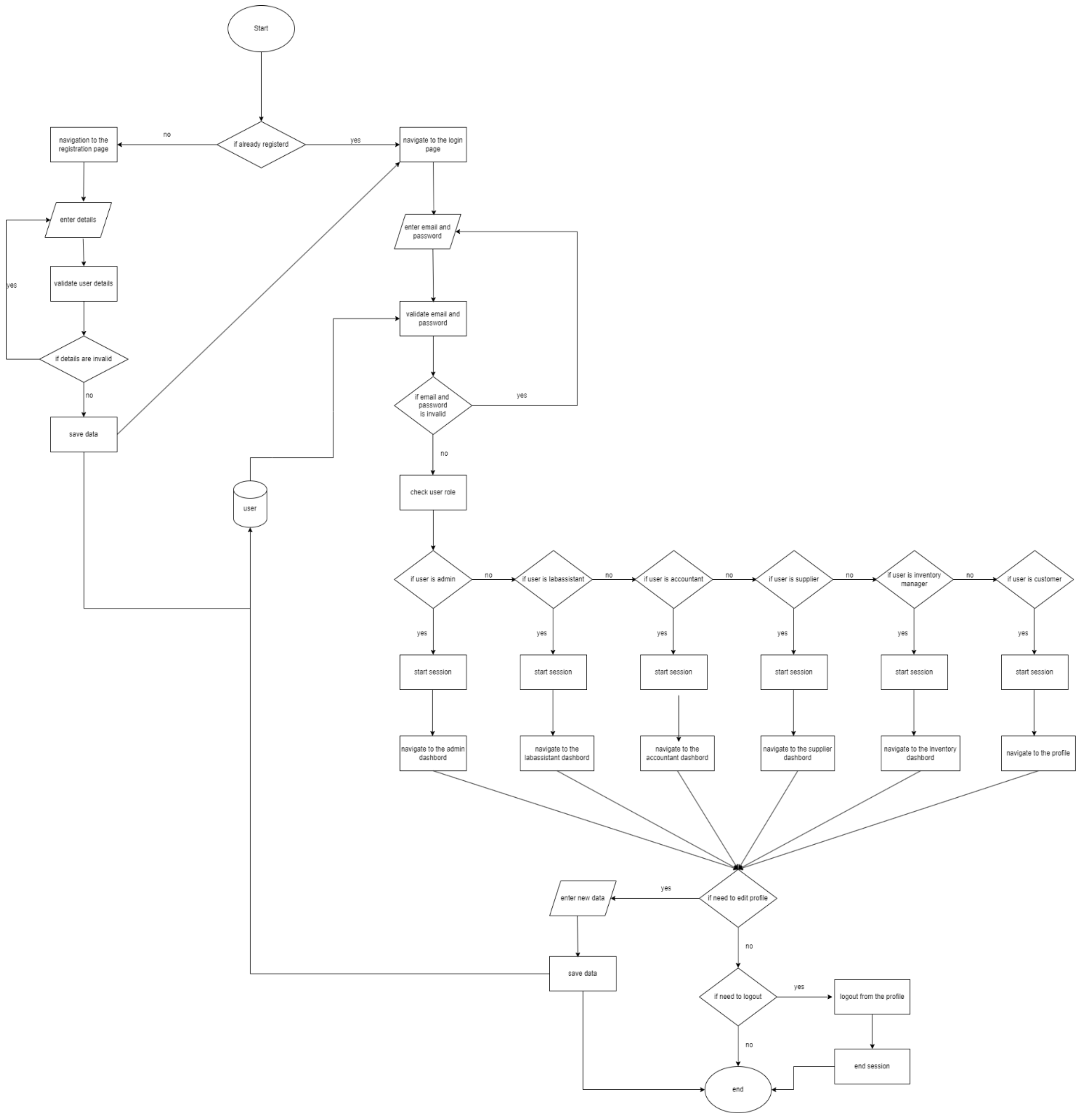
TC05	Inventory Alerts	Low stock alert	Set threshold (e.g., < 10)	Set threshold (e.g., < 10) → Item reaches threshold	"Low Stock Alert"
TC06	Inventory Viewing	Fetch inventory list	Login as admin	Login as admin → View inventory	All items displayed with correct details
TC07	Inventory Exporting	Export pdf Inventories	Inventories exists	1. Click "PDF"	PDF file downloaded successfully
TC08	Invoices Exporting	Export pdf invoices	Invoice exists	1. Click "PDF"	PDF file downloaded successfully
TC09	Requests Exporting	Export pdf requests	Request exists	1. Click "PDF"	PDF file downloaded successfully

03. Individual Contribution

3.1 K.H.S.Dinsara (IT23588714)

1. User Management

Task	Status	Note (Implemented parts and not parts)
User Registration	Completed	Implemented registration page with account creation and DB storage
User Login	Completed	Implemented secure login with session/JWT generation
Role-Based Login	Completed	Implemented role-based redirection and dashboard access
Update Profile	Completed	Implemented profile edit and update functionality
Password Hashing	Completed	Implemented-secure password hashing in database
Admin View All Appointments in charts	Not Completed	Read the backend and generate line and a bar chart of showing appointments booking all of the time.
Admin View All Appointments	Completed	Implemented admin dashboard listing all appointments
Admin Search & Filter appointments	Completed	Implemented search and filter by specialty and status
Export Personal Data	Completed	Export function partially implemented; data export formats done.


Start
if already registerd
no
yes
navigation to the registration page
navigate to the login page
enter details
validate user details
yes
if details are invalid
no
save data
enter email and password
validate email and password
if email and password is invalid
yes
no
user
check user role
if user is admin
no
if user is labassistant
no
if user is accountant
no
if user is supplier
no
if user is inventory manager
no
if user is customer
yes
yes
yes
yes
yes
yes
start session
start session
start session
start session
start session
start session
navigate to the admin dashbord
navigate to the labassistant dashbord
navigate to the accountant dashbord
navigate to the supplier dashbord
navigate to the inventory dashbord
navigate to the profile
enter new data
yes
if need to edit profile
no
save data
if need to logout
yes
logout from the profile
no
end session
end

➢ Pseudocode

```
BEGIN
  IF user is already registered THEN
    NAVIGATE to login page
      ENTER email AND password
      VALIDATE credentials
      IF invalid THEN
        DISPLAY error
      ELSE
        CHECK user role
        IF role = Shop Admin THEN
          START session
          NAVIGATE to Admin Dashboard
        ELSE IF role = Accountant THEN
          START session
          NAVIGATE to Accountant Dashboard
        ELSE IF role = Lab assistant THEN
          START session
          NAVIGATE to Lab assistant Dashboard
        ELSE IF role = Supplier THEN
          START session
          NAVIGATE to Supplier Dashboard
        ELSE IF role = Inventory manager THEN
          START session
          NAVIGATE to Inventory manager Dashbord
        ELSE IF role = Customer THEN
          START session
          NAVIGATE to customer profile
        ENDIF
        IF need to edit profile THEN
          ENTER new data
          SAVE data
        ELSE IF need to logout THEN
          LOGOUT
        ENDIF
      ENDIF
    ENDIF
  ELSE
    NAVIGATE to registration page
    ENTER details
    VALIDATE details
    IF valid THEN
      SAVE data into user
    ELSE
```

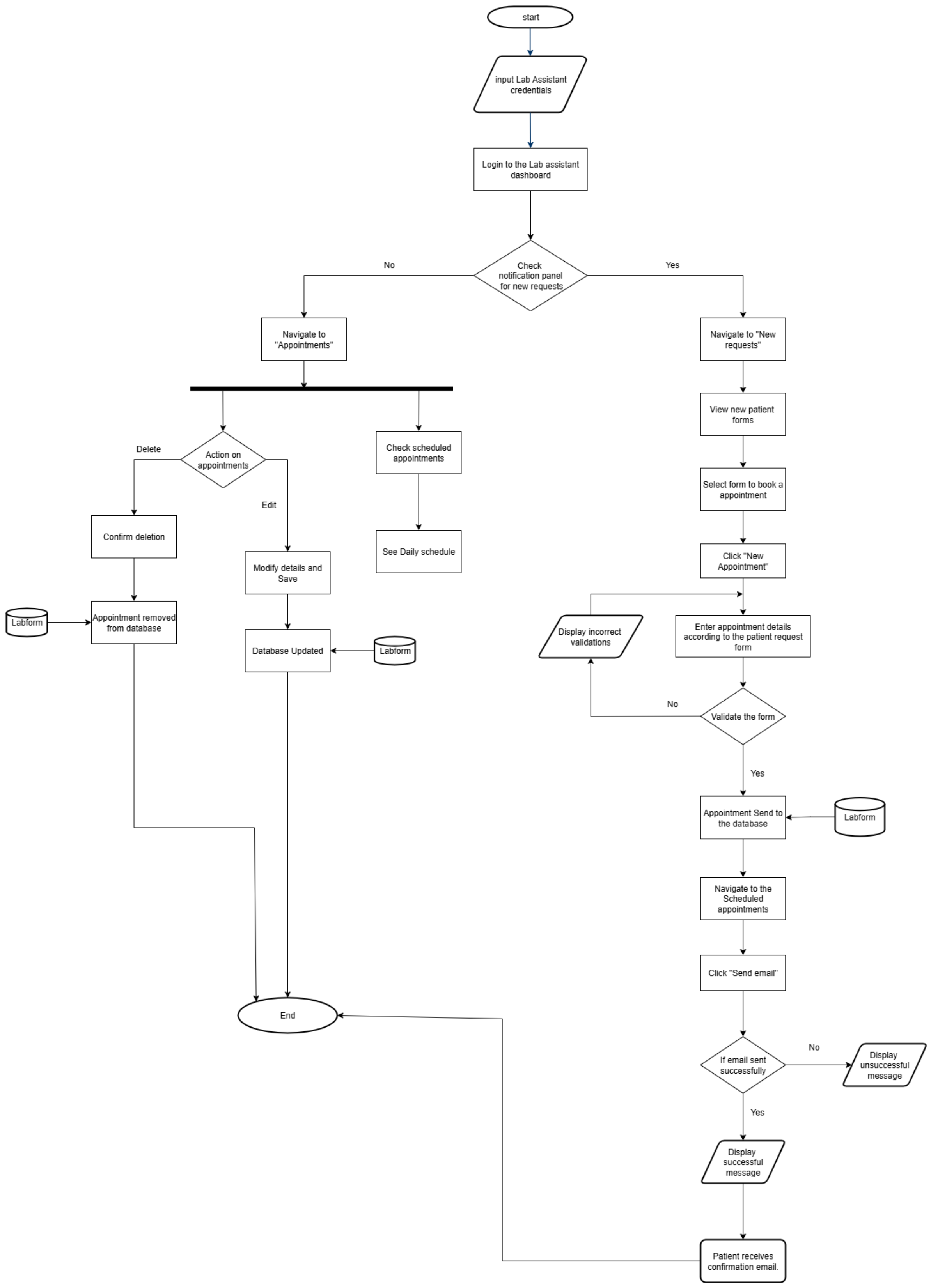
```
            DISPLAY error
         ENDIF
      ENDIF
   END
```

3.2 Thathsarani J. R. (IT23572638)

2. Laboratory Management

Task	Status	Note (Implemented parts and not parts)
Patient details form submission	Completed	Designed and implemented patient form with field validation and data submission to backend.
Lab assistant view of submitted forms	Completed	Created dashboard section displaying all patient-submitted forms for review and booking.
Appointment creation	Completed	Developed API and UI to allow lab assistants to create appointments using patient form data.
Appointment edit/update	Completed	Added appointment editing feature and update confirmation messages.
Appointment deletion	Completed	Implemented delete functionality with confirmation messages to avoid accidental deletions.
Email confirmation using EmailJS	Completed	Integrated EmailJS API to send automatic appointment confirmation emails to patients.
Medical AI Bot integration	Completed	Added AI bot to analyze uploaded lab reports and answer basic medical questions.
Dashboard notifications	Completed	Configured notification panel for real-time alerts when patients submit new forms.

Search and filter functionality	Not Completed	Implemented dynamic search bar for filtering patients and tests by name.


start
input Lab Assistant credentials
Login to the Lab assistant dashboard
Check notification panel for new requests
No
Yes
Navigate to "Appointments"
Navigate to "New requests"
Action on appointments
Delete
Edit
Check scheduled appointments
See Daily schedule
Confirm deletion
Modify details and Save
Labform
Appointment removed from database
Database Updated
Labform
View new patient forms
Select form to book a appointment
Click "New Appointment"
Display incorrect validations
Enter appointment details according to the patient request form
No
Validate the form
Yes
Appointment Send to the database
Labform
Navigate to the Scheduled appointments
Click "Send email"
End
If email sent successfully
No
Display unsuccessful message
Yes
Display successful message
Patient receives confirmation email.

➢ Pseudocode

```
START
DISPLAY "Lab Assistant Login"
INPUT Lab assistant credential
IF credentials ARE VALID THEN
  LOGIN successful
  REDIRECT to Lab Assistant Dashboard
ELSE
  DISPLAY "Invalid credentials"
  EXIT
END IF
CHECK notification panel FOR new requests
IF new requests EXIST THEN
  DISPLAY "New appointment requests available"
ELSE
  DISPLAY "No new requests"
END IF
NAVIGATE to "Appointments" section
WHILE IN appointments section DO
  IF user selects "Check scheduled appointments" THEN
    DISPLAY all scheduled appointments
  END IF
  IF user selects "See Daily schedule" THEN
    DISPLAY daily schedule appointments
  END IF
  IF user selects existing appointment THEN
    DISPLAY action options: ["Edit", "Delete"]
    CASE OF selected action:
      WHEN "Delete":
        DISPLAY "Confirm deletion?"
        IF user confirms THEN
          EXECUTE DELETE appointment FROM database
          IF deletion successful THEN
            DISPLAY "Appointment removed from database"
            UPDATE database
          ELSE
            DISPLAY "Deletion failed"
          END IF
        END IF
      WHEN "Edit":
        DISPLAY current appointment details
        INPUT modified details
        CALL validate form(modified details)
        IF validation passed THEN
          EXECUTE UPDATE database WITH modified details
          DISPLAY "Database Updated"
        ELSE
          DISPLAY validation errors
        END IF
    END CASE
  END IF
```

```
    IF user selects "Create New Appointment" THEN
        DISPLAY appointment form
        INPUT appointment details FROM patient request form
        CALL validate form(appointment details)
        IF validation passed THEN
            EXECUTE SAVE appointment details TO database
            DISPLAY "Appointment sent to database"
            NAVIGATE TO scheduled appointments
            SELECT "Send email" FOR new appointment
            IF email sent successfully THEN
                DISPLAY "Successful message"
                PATIENT receives confirmation email
            ELSE
                DISPLAY "Unsuccessful message"
            END IF
        ELSE
            DISPLAY "Incorrect validations"
        END IF
    END IF
    IF user selects "Logout" THEN
        BREAK FROM LOOP
    END IF
END WHILE
FUNCTION validate form(form data)
    SET is valid = TRUE
    SET errors = []
    IF form data. Patient name IS EMPTY THEN
        APPEND "Patient name is required" TO errors
        SET is valid = FALSE
    END IF
    IF form data. Appointment date IS EMPTY OR INVALID THEN
        APPEND "Valid appointment date is required" TO errors
        SET is valid = FALSE
    END IF
    IF form data. test type IS EMPTY THEN
        APPEND "Test type is required" TO errors
        SET is valid = FALSE
    END IF
    RETURN is valid, errors
END FUNCTION
FUNCTION send email(appointment data)
    COMPOSE email message WITH appointment details
    SET email status = SEND email TO appointment data. Patient email
    RETURN email status
END FUNCTION
END
```

3.3 Ekanayaka E.W.I.D (IT23810464)

Billing and Payment Management

Completion level of features

Billing & Payment Management:

Patient Invoices:

- Add, edit, delete invoices (CRUD)
- Validation on invoice fields (amount, patient ID, date)
- PayHere integration for online payments Print/Download invoice functionality
- Reports module (daily/weekly/monthly)
- Advanced filtering and sorting for invoices and payments

Expenses Management:

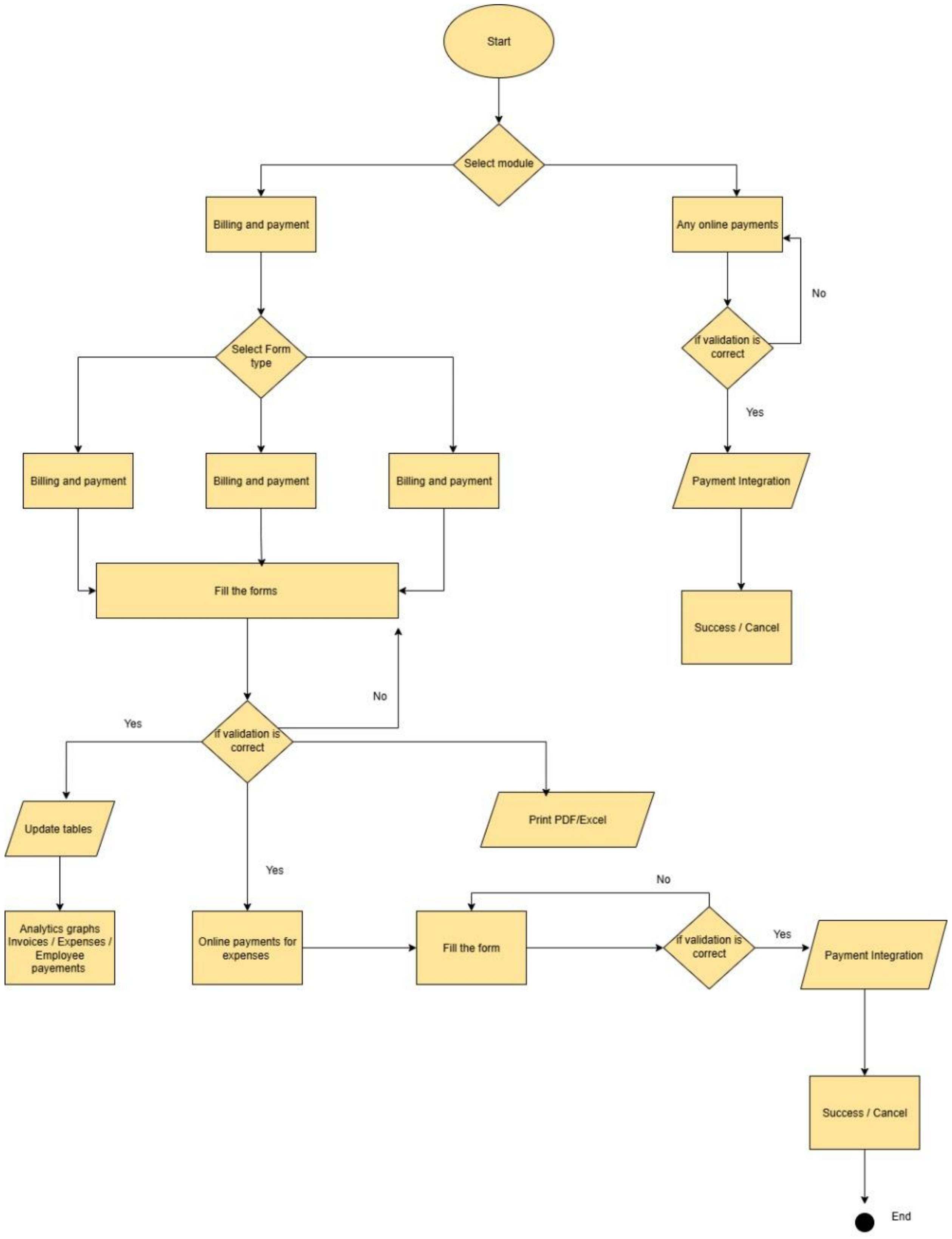
- Add, edit, delete expense records
- Categorize expenses (supplies, equipment, maintenance)
- Validation on amounts and date
- Reports module (daily/weekly/monthly)
- Advanced filtering and sorting for invoices and payments

Employee Payments:

- Add, edit, delete salary payments
- Integration with employee module
- Frontend validation
- Reports module (daily/weekly/monthly)
- Advanced filtering and sorting for invoices and payments

Pending/To Complete:

- Reports module (daily/weekly/monthly)
- Advanced filtering and sorting for invoices and payments
- Minor UI enhancements for mobile responsiveness


Start
Select module
Billing and payment
Any online payments
No
if validation is correct
Yes
Select Form type
Billing and payment
Billing and payment
Billing and payment
Payment Integration
Fill the forms
Success / Cancel
No
Yes
if validation is correct
Update tables
Print PDF/Excel
Yes
No
Analytics graphs Invoices / Expenses / Employee payements
Online payments for expenses
Fill the form
if validation is correct
Yes
Payment Integration
Success / Cancel
End

➢ Pseudocode:

```
Function: Process Patient Invoice
def processInvoice(invoice_id):
    invoice = getInvoiceFromDB(invoice_id)
    patient = getPatientFromDB(invoice.patient_id)

    Validate invoice data
    if not invoice.services or invoice.amount <= 0:
        return "Invalid Invoice Data!"
    Save invoice to database
    saveInvoice(invoice)

    Online payment using PayHere
    if invoice.payment_method == "PayHere":
        payment_status = redirectToPayHere(invoice)
        if payment_status == "Paid":
            updateInvoiceStatus(invoice_id, "Paid")
        else:
            updateInvoiceStatus(invoice_id, "Pending")

    Generate printable receipt
    generateReceipt(invoice)

     Update Analytics
    updateAnalyticsDailyRevenue(invoice.amount)
    updateAnalyticsMonthlyRevenue(invoice.amount)
    updateAnalyticsPaymentMethod(invoice.payment_method)
    updatePatientInvoiceCount(invoice.patient_id)

    return "Invoice Processed Successfully!"
```

Function: Record Expense

```
def recordExpense(expense_id):
    expense = getExpenseFromDB(expense_id)
  Validate expense data
    if expense.amount <= 0 or not validDate(expense.date):
        return "Invalid Expense Data!"
    Save expense to database
    saveExpense(expense)
    Update Analytics
    updateTotalExpenses(expense.date, expense.amount)
    updateCategoryExpenses(expense.category, expense.amount)
    updateMonthlyExpenses(getMonth(expense.date), expense.amount)
    return "Expense Recorded Successfully!"
```

Function: Process Employee Payment

```
def processEmployeePayment(payment_id):
    payment = getEmployeePaymentFromDB(payment_id)
    employee = getEmployeeFromDB(payment.employee_id)
    Validate payment
    if payment.amount <= 0:
        return "Invalid Payment Amount!"
    Save payment to database
    saveEmployeePayment(payment)
    Generate payment slip
    generatePaymentSlip(payment)
Update Analytics
    updateTotalPayroll(getMonth(payment.payment_date), payment.amount)
    updateEmployeePaymentCount(payment.employee_id)
    updateDepartmentPayroll(employee.department, payment.amount)


    return "Employee Payment Processed Successfully!"
```

Function: Generate Billing Analytics

```
def generateBillingReport(start_date, end_date):
    invoices = fetchInvoices(start_date, end_date)
    expenses = fetchExpenses(start_date, end_date)
    employee_payments = fetchEmployeePayments(start_date, end_date)

     Compute totals
    total_revenue = sum([inv.amount for inv in invoices])
    total_expenses = sum([exp.amount for exp in expenses])
    total_payroll = sum([pay.amount for pay in employee_payments])
    net_profit = total_revenue - (total_expenses + total_payroll)

    Top patients by invoice amount
    patient_totals = {}
    for inv in invoices:
        if inv.patient_id in patient_totals:
            patient_totals[inv.patient_id] += inv.amount
        else:
            patient_totals[inv.patient_id] = inv.amount
    top_patients = sorted(patient_totals.items(), key=lambda x: x[1], reverse=True)[:5]
    report = {
        "total_revenue": total_revenue,
        "total_expenses": total_expenses,
        "total_payroll": total_payroll,
        "net_profit": net_profit,
        "top_patients": top_patients
    }
   return report
```

Function: Apply Discount / Promotion

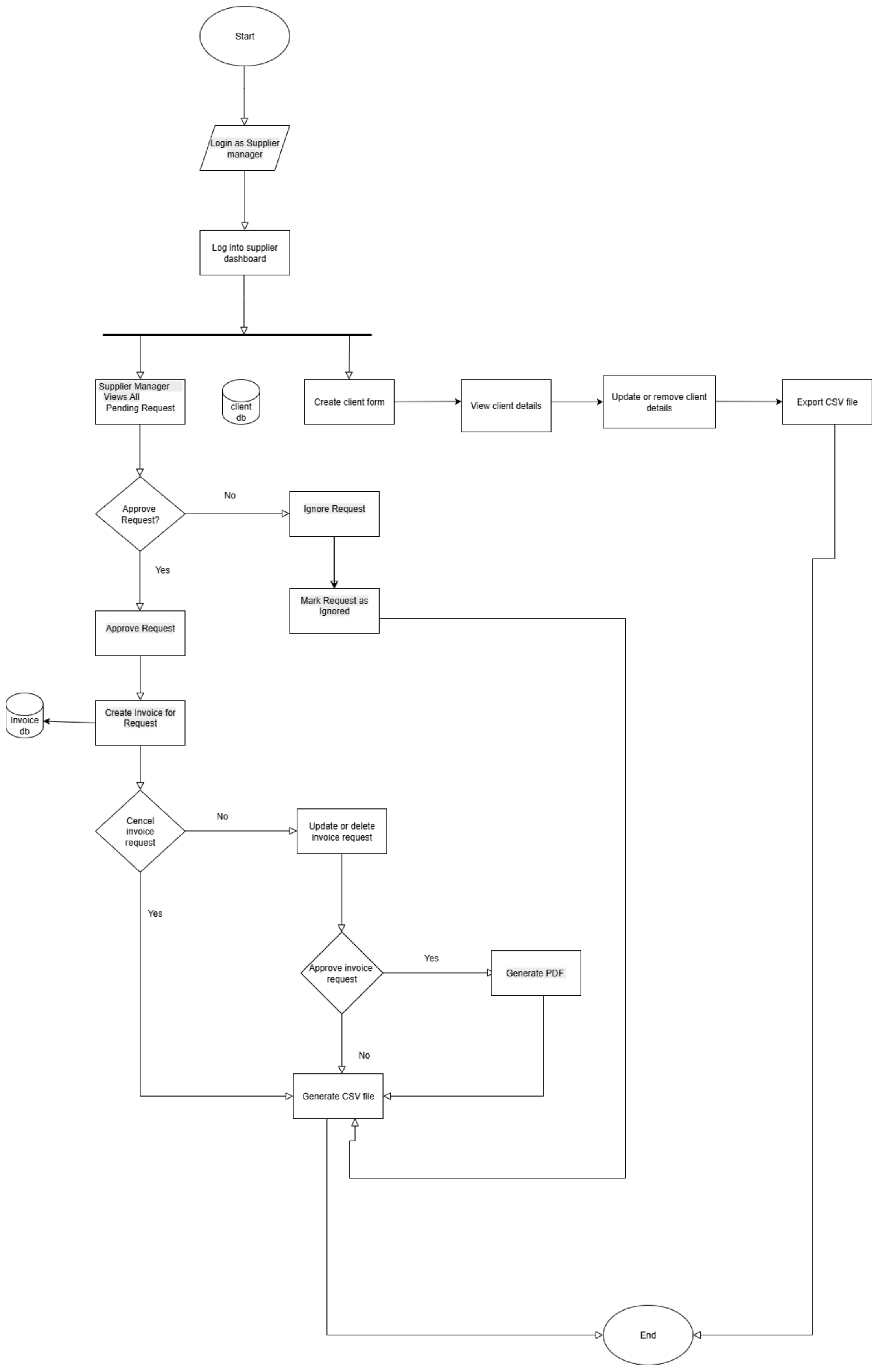
```
def applyInvoiceDiscount(invoice_id):
    invoice = getInvoiceFromDB(invoice_id)
    discount = 0
    Discount rules
```

```
if invoice.amount > 500:          # Example: 10% discount on invoices > 500
    discount = invoice.amount * 0.10
elif len(invoice.services) >= 5:  # Example: 5+ services, 5% discount
    discount = invoice.amount * 0.05
final_amount = invoice.amount - discount
updateInvoiceDatabase(invoice_id, {"final_amount": final_amount, "discount_applied": discount})
return f"Discount Applied! New Total: Rs.{final_amount}"
```

3.4 Navodya A.K. (IT23543232)

3. Supplier Management

Task	Status	Note (Implemented parts and not parts)
View all inventory requests sent by the Inventory Manager	Completed	Successfully implemented and tested. Supplier Manager can view all pending inventory requests.
Approve or ignore inventory requests	Completed	Approve and ignore buttons fully functional; updates status in the database
Restrict Inventory Manager from editing/deleting requests after action	Completed	Lock mechanism works correctly once the Supplier Manager approves or ignores a request.
Create new invoice for approved requests	Completed	Supplier Manager can create invoices linked to approved requests.
Update and delete invoice until Inventory Manager acts (approve/cancel)	Completed	Update/Delete access restricted after Inventory Manager's action.
Generate and download invoice as PDF	Completed	PDF generation and download function tested and working properly.
Export all invoice details as CSV	Completed	CSV export file generated successfully with all invoice records.
Add, view, update, and delete client details	Completed	Client management CRUD operations fully implemented.
Notification system for invoice or request updates	Completed	Basic alerts implemented; real-time notifications planned for future update.


Start
Login as Supplier manager
Log into supplier dashboard
Supplier Manager Views All Pending Request
client db
Create client form
View client details
Update or remove client details
Export CSV file
Approve Request?
No
Ignore Request
Yes
Mark Request as Ignored
Approve Request
Invoice db
Create Invoice for Request
Cencel invoice request
No
Update or delete invoice request
Yes
Approve invoice request
Yes
Generate PDF
No
Generate CSV file
End

- Pseudocode:

```
BEGIN

    DISPLAY "Login as Supplier Manager"
    LOGIN(user_role)

    IF login_successful THEN
        OPEN Supplier_Dashboard

        DISPLAY "View Pending Requests"
        FETCH all_pending_requests FROM request_db

        FOR each request IN all_pending_requests DO
            DISPLAY request_details
            ASK "Approve this request? (Yes/No)"

            IF response = "Yes" THEN
                APPROVE request
                CREATE invoice FOR request
                STORE invoice IN invoice_db

                ASK "Cancel invoice request? (Yes/No)"
                IF response = "Yes" THEN
                    CANCEL invoice
                ELSE
                    ASK "Approve invoice request? (Yes/No)"
                    IF response = "Yes" THEN
                        GENERATE PDF(invoice)
```

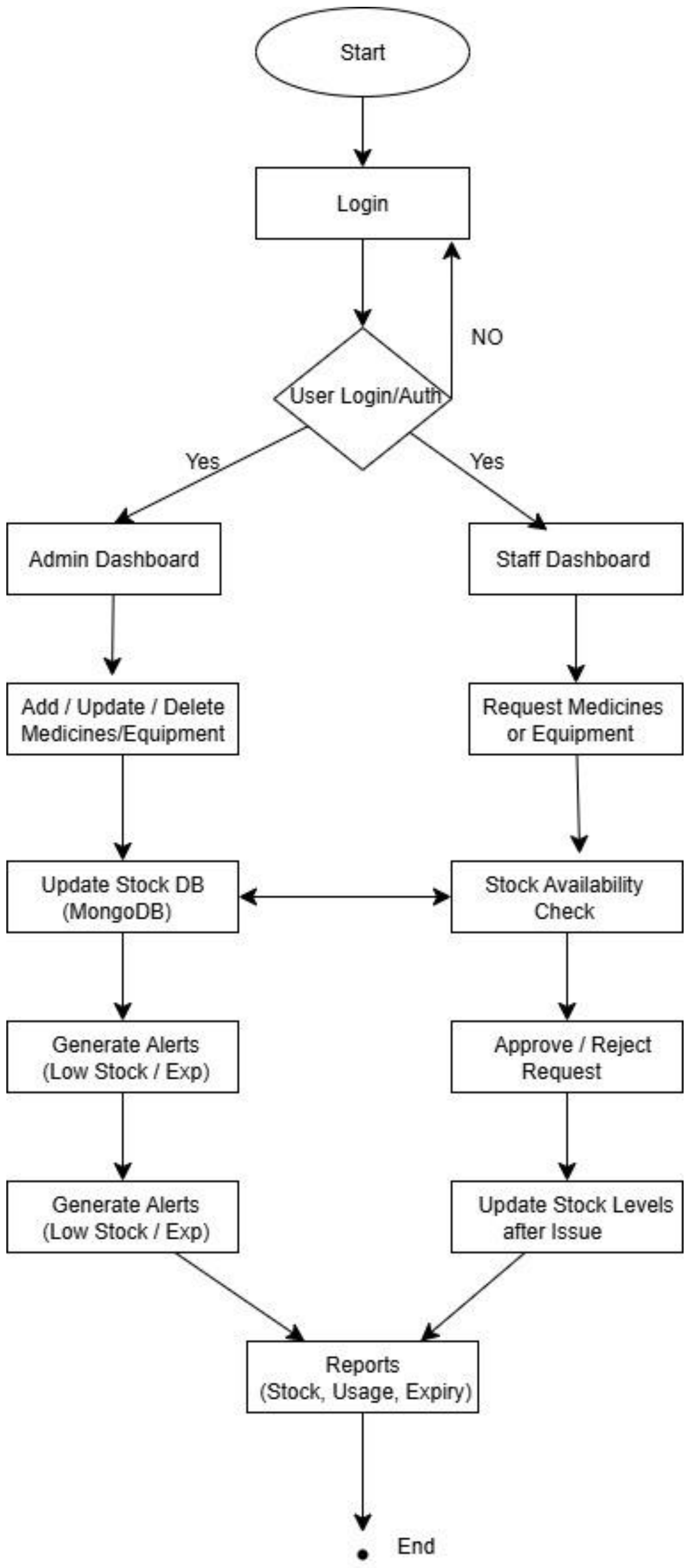
```
          ELSE
            GENERATE CSV(invoice)
          ENDIF
        ENDIF

      ELSE
        IGNORE request
        MARK request AS ignored
      ENDIF

    END FOR

    DISPLAY "Manage Client Details"
    CREATE new_client_form
    VIEW client_details
    UPDATE_OR_REMOVE client_details
    EXPORT client_details TO CSV

  ELSE
    DISPLAY "Login Failed. Try again."
  ENDIF

END
```

3.5 Liyanaarachchi L.A.D.A (IT23669758)

5. Inventory Manager

Task	Status	Note (Implemented parts and not parts)
Track current stock levels	Completed	Inventory Manager can view all available stock items; backend implemented, frontend integration in progress.
Receive low-stock alerts	Completed	System triggers alerts automatically when medication quantities drop below the minimum threshold (e.g., less than 10 units).
Add new medication items	Completed	Inventory Manager can add new stock entries when new supplies arrive; data is stored securely in the database.
Record stock usage	Completed	Function allows tracking of used or dispensed medication when fulfilling internal or external orders.
Generate inventory and invoice reports	Completed	Inventory Manager can generate detailed reports for inventory, invoices, and requests; backend functionality is complete, frontend interface in development.
Send request to supplier	Completed	Inventory Manager can send supply requests directly to the Supplier Manager if stock for certain medication is low.
Update or delete medication records	Completed	Inventory Manager can edit existing medication details or delete outdated records; validations applied for data accuracy.

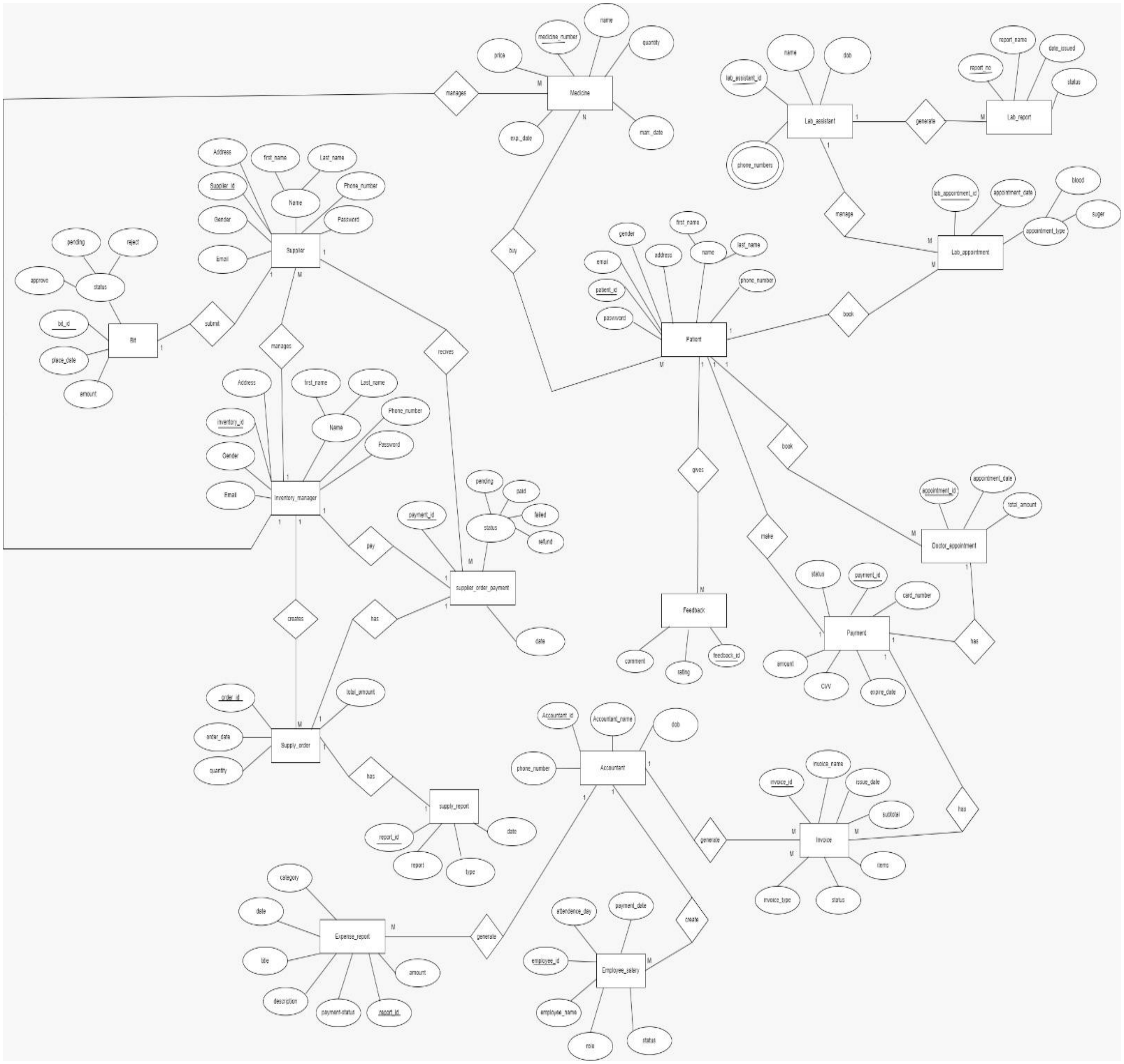
Notification system for invoice or request updates	Completed	Basic alerts implemented; real-time notifications planned for future update.


Start
Login
User Login/Auth
NO
Yes
Yes
Admin Dashboard
Staff Dashboard
Add / Update / Delete Medicines/Equipment
Request Medicines or Equipment
Update Stock DB (MongoDB)
Stock Availability Check
Generate Alerts (Low Stock / Exp)
Approve / Reject Request
Generate Alerts (Low Stock / Exp)
Update Stock Levels after Issue
Reports (Stock, Usage, Expiry)
End

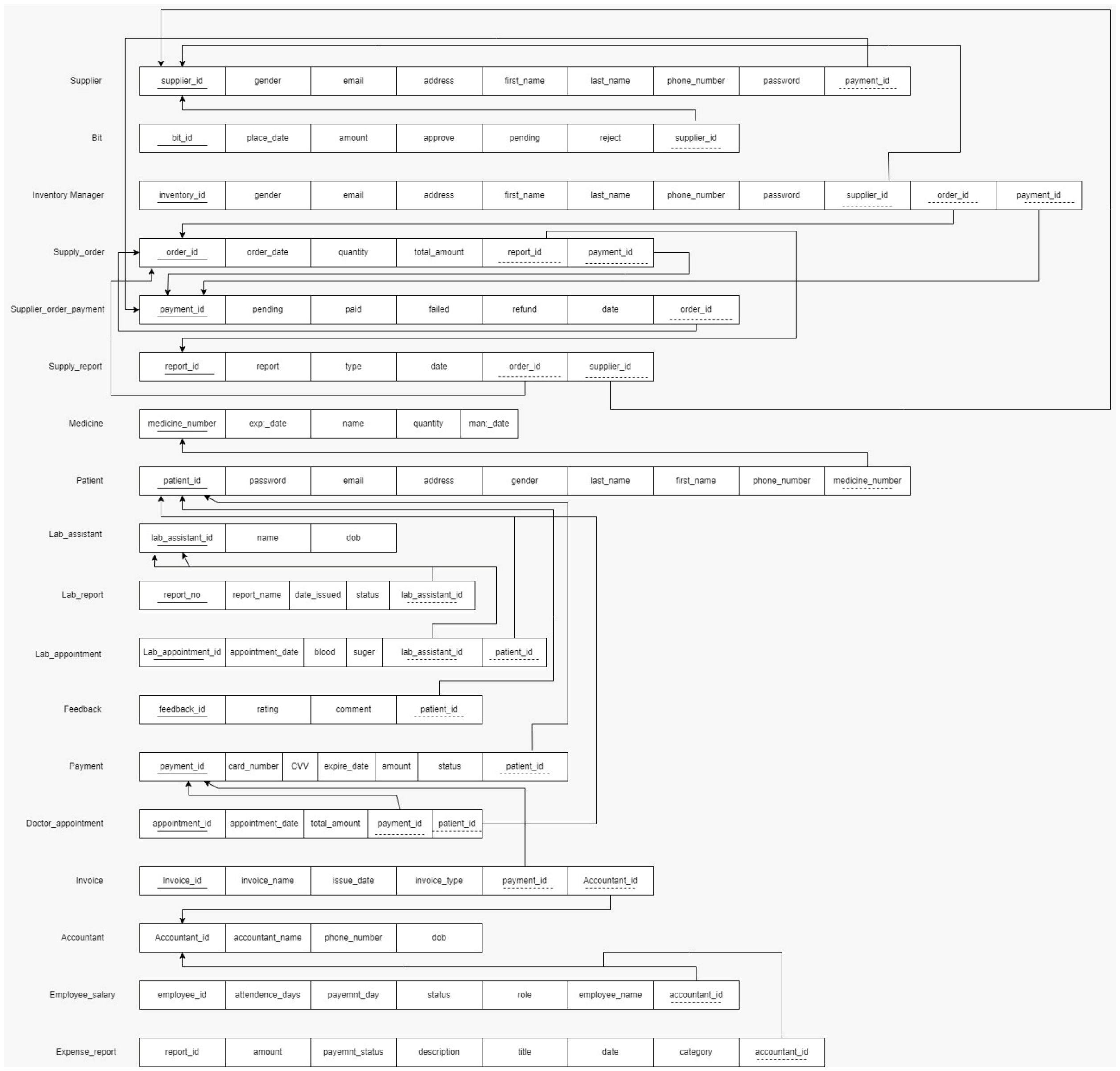
- **pseudocode**

```
START

LOGIN()
IF authentication_success THEN
   IF user_role == "Admin" THEN
      DISPLAY Admin Dashboard

      DO
         DISPLAY options: [Add, Update, Delete Medicines/Equipment]
         SELECT option
         IF option == "Add" OR option == "Update" OR option == "Delete" THEN
            PERFORM operation on Stock Database (MongoDB)
         ENDIF

         CHECK Stock Levels
         IF Stock < Threshold OR Item Near Expiry THEN
            GENERATE Alert (Low Stock / Expiry)
         ENDIF

         GENERATE Report (Stock, Usage, Expiry)
      WHILE user_wants_to_continue

   ELSE IF user_role == "Staff" THEN
      DISPLAY Staff Dashboard

      DO
         DISPLAY options: [Request Medicines or Equipment]
         SELECT item
         CHECK Stock Availability
         IF Stock Available THEN
            APPROVE Request
            UPDATE Stock Levels after Issue
         ELSE
            REJECT Request
         ENDIF

         IF Stock < Threshold OR Item Near Expiry THEN
            GENERATE Alert (Low Stock / Expiry)
         ENDIF

         GENERATE Report (Stock, Usage, Expiry)
      WHILE user_wants_to_continue

   ENDIF
ELSE
   DISPLAY "Login Failed. Try Again."
ENDIF

END
```

04.ER Diagram – (**ER.png**)



05.Normalized Schema – (**Normalization.png**)



06.High-level system design diagram

MediCura is a hospital management web application that operates under the management of user management, laboratory management, inventory management, billing & payments, and supplier management. It is primarily developed using the MERN web development stack. This ensures efficient data processing, real-time updates, and improved accuracy, enabling hospitals to provide optimal service to patients, staff, and stakeholders. The technologies used in the system are as follows:

- Frontend: React.js
- Backend: Express.js + Node.js
- Database: MongoDB

System modules:

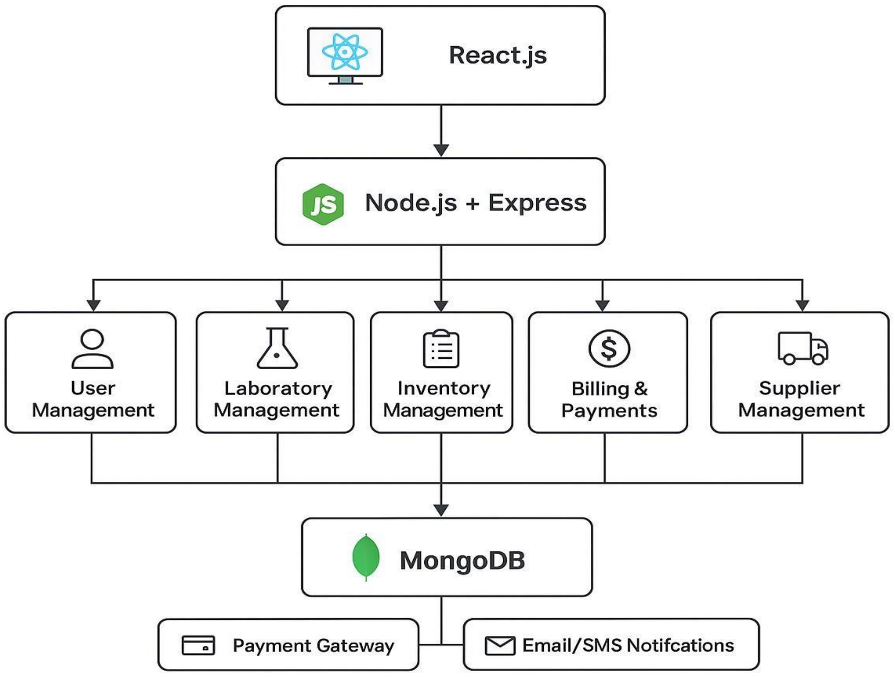
1. User Management – Handles patient/staff registration, authentication, and role-based access.

2. Laboratory Management – Manages test bookings, results, and reports.

3. Inventory Management – Tracks medicines, equipment, and stock levels.

4. Billing & Payments – Handles invoices, receipts, online payments, and salary management.

5. Supplier Management – Manages supplier details, orders, and procurement records.



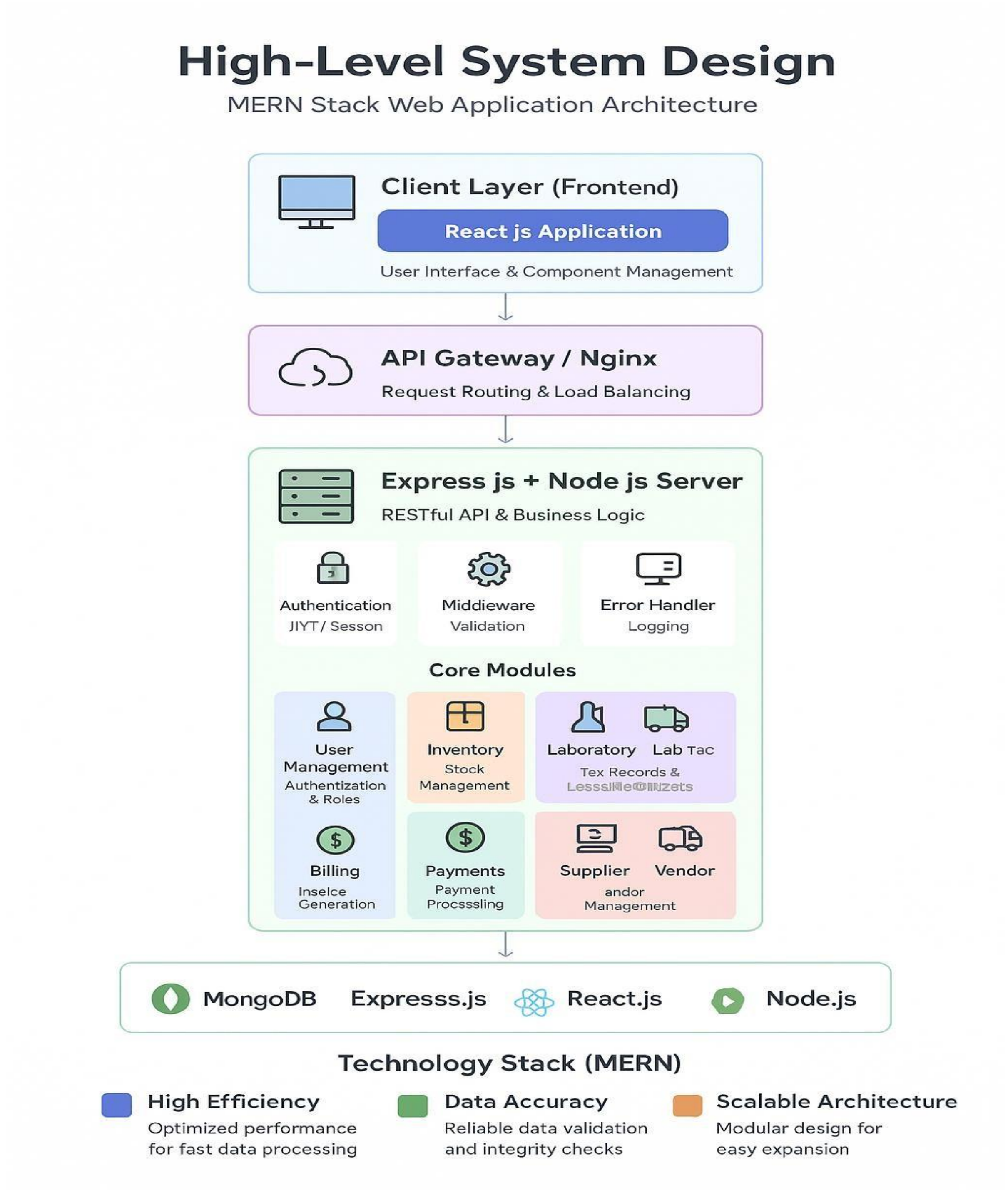

High-Level System Design
MERN Stack Web Application Architecture
Client Layer (Frontend)
React js Application
User Interface & Component Management
API Gateway / Nginx
Request Routing & Load Balancing
Express js + Node js Server
RESTful API & Business Logic
Authentication
JIYT / Sesson
Middieware
Validation
Error Handler
Logging
Core Modules
User Management
Authentication & Roles
Inventory
Stock Management
Laboratory
Lab Tac
Tex Records &
Billing
Inselce Generation
Payments
Payment Procsssling
Supplier
Vendor
andor Management
MongoDB
Expresss.js
React.js
Node.js
Technology Stack (MERN)
High Efficiency
Optimized performance for fast data processing
Data Accuracy
Reliable data validation and integrity checks
Scalable Architecture
Modular design for easy expansion

Diagram flow:

- **Client UI (React.js) → Application Server (Node.js + Express) → Database Server (MongoDB)**
- APIs for each module (User, Lab, Inventory, Billing, Supplier) interact with the database.
- External integrations: **Payment Gateway** (PayHere/Stripe) + **Email/SMS notifications**.

07.Network Design

Architecture Overview

The Medical Center System follows a three-layer MERN architecture for secure, scalable, and efficient operations.
It consists of:

- Frontend (React.js) – User interface layer
- Backend (Node.js + Express.js) – Application logic and API gateway
- Database (MongoDB) – Data storage and retrieval layer

Layer Summary

Layer	**Technology Used**	**Key Features**	**Role in Network Design**
Frontend (Client Layer)	React.js	- Responsive UI (hooks, components) - Real-time updates via Context API or Redux - Lazy loading and routing - API communication using Axios/Fetch	- Runs on user devices (browser/mobile) - Communicates with backend via secure HTTPS REST APIs - Hosted in the DMZ or served by Node.js backend - Provides user-friendly interface for all system

			modules (Supplier, Inventory, Lab, Billing)
Backend (Application Layer)	Node.js + Express.js	- RESTful APIs for CRUD operations - JWT-based authentication & role access control - Handles business logic (requests, invoices, lab results) - Scalable and asynchronous event handling - Logging (Winston/Morgan)	- Runs on internal application servers - Connects frontend and database - Secured behind firewall and reverse proxy - Routes client requests to database and other services
Database (Data Layer)	MongoDB	- Document-based schema for flexibility- Data validation and indexing- Role-based access control- Backup and recovery scripts- Supports aggregation and reporting	- Hosted on private database VLAN- Stores all data related to users, suppliers, inventory, and invoices- Only accessible from backend layer- Provides persistent data storage and high availability (replica set)

Key Components

- **Reverse Proxy (Nginx)** – Handles HTTPS, load balancing, caching, and routing API requests.
- **Firewall / WAF** – Protects against unauthorized access and network attacks.
- **Backend Cluster** – Node.js servers handle business logic, user requests, and authentication.
- **Database Server** – MongoDB primary and replica nodes in a private VLAN, isolated from internet.
- **Monitoring Tools** – ELK stack or Prometheus for performance tracking and security auditing.
- **Backup Server** – Regular snapshots of MongoDB stored on encrypted storage or cloud backup.

Security & Performance

- **HTTPS/TLS encryption** for all client-server communication.
- **JWT authentication** for secure session handling.
- **Role-based access control** for different system modules (Supplier, Inventory, Lab, Billing).
- **Firewall segmentation:**
    - Internet → DMZ (React frontend / Reverse Proxy)
    - DMZ → Application (Node.js APIs)
    - Application → Database (MongoDB)
- **Data backups** and **replica sets** for high availability.
- **Scalable design:** Load balancer distributes traffic to multiple Node.js instances.

08.Commercialization

Target Users / Market

Our Medical Center Management System is designed for the people who keep healthcare centers running every day.
It will mainly be used by **supplier managers, inventory managers, laboratory managers, billing managers, and administrative staff**.
Each of them can use the system to handle their specific tasks — such as managing supplies, monitoring stock levels, recording lab results, or generating invoices — all in one secure and easy-to-use web platform.
The system is ideal for **private hospitals, clinics, laboratories, and medical suppliers** that want to move away from time-consuming manual processes and adopt a digital solution to manage their daily operations efficiently.
Scalability

The system is built using the **MERN stack (MongoDB, Express.js, React.js, Node.js)**, which makes it easy to scale as the number of users and data grows.
It can be expanded by:

- Adding more servers or databases to handle increased demand.
- Hosting it on the cloud to support multiple branches or hospitals.
- Developing a mobile version for quick access by staff and management.
- Adding new modules like **pharmacy, appointments, or patient management** in the future.

This flexibility ensures the system can grow along with the healthcare institution's needs.

Practical Benefits

Hospitals and clinics will benefit from this system because it makes daily work faster, easier, and more accurate.
It helps to:

- **Save time** by automating manual tasks like report generation and invoice approval.
- **Reduce errors** by using smart validation and real-time updates.
- **Improve communication** between departments, leading to smoother workflows.
- **Enhance security** by keeping sensitive data safe with login controls and encryption.
- **Provide better service** to patients and staff by speeding up request handling and decision-making.

Business Model

- **Subscription Model:**
  Hospitals and clinics can pay a monthly or yearly subscription based on how many users or modules they need, such as Inventory, Laboratory, or Billing.
- **License Purchase:**
  Larger medical organizations can buy a one-time license to use the system permanently, with an option for annual updates and support.
- **Cloud Service (SaaS):**
  Smaller clinics can use the system online without worrying about installation or maintenance. The system can be hosted on cloud platforms like AWS or Azure and accessed through a secure web login.
- **Customization & Support Services:**
  Revenue can also come from offering extra features, on-site training, and ongoing technical support to meet the specific needs of each client.

09.Github

https://github.com/navo2002-abey/allinone

https://github.com/navo2002-abey/test1