Software Requirements Specification

for

Medical laboratory Management System (MedLab)

Version 2.0

Prepared by Team 18

(March – 2024)

Name	PRN
Naresh Lokhande	240340320064
Pradeep Bahukhandi	240340320070
Pavitra Patil	240340320069
Sayali Golvankar	240340320092
Abhijit Jogade	240340520004

Centre for Development of Advanced Computing

Raintree Marg, Near Bharati Vidyapeeth, Opp. Kharghar Railway Station, Sector 7,

CBD Belapur, Navi Mumbai - 400 614 - Maharashtra (India)

Phone: +91-22-27565303

Fax: +91-22-2756-0004

INDEX

Table of Contents

Revision History

Name	Date	Reason For Changes	Version

1. Introduction

1.1 Purpose

This SRS document provides a detailed description of the Medical Laboratory Management System (MLMS), including functional and non-functional requirements, system features, external interface requirements, and overall system architecture. The goal is to provide a clear understanding of the system to be developed using ASP.NET API and Spring Boot frameworks.

1.2 Project Scope

MLMS is a web-based application designed to manage the operations of medical laboratories. The system will support various user roles, including Admin, Lab Assistants, and Users. It will handle tasks such as user authentication, appointment scheduling, lab test management, and reporting.

1.3 Definitions, Acronyms, and Abbreviations

- **MLMS**: Medical Laboratory Management System
- **ASP.NET Web API: A** framework developed by Microsoft for building HTTP services that can be consumed by a variety of clients.Spring Boot: A framework for building Java-based applications
- **Entity Framework**: An ORM framework for .NET applications
- **JPA**: Java Persistence API

1.4 References

- https://spring.io/
- https://www.javascript.com/
- https://developer.mozilla.org/en-US/docs/Web/JavaScript
- https://en.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93controller
- Spring Boot + React: JWT Authentication with Spring Security - BezKoder
- ASP.NET Core Documentation
- Bootstrap Documentation
- MySQL Documentation
- Entity Framework Core
- Microsoft Azure
- Stripe API Documentation
- https://dotnet.microsoft.com/en-us/apps/aspnet/apis

2. Overall Description

2.1 Product Perspective

MLMS is a standalone web application that integrates with existing laboratory systems to provide a centralized platform for managing lab operations.

2.2 Product Features

- User authentication and authorization
- Appointment scheduling and management
- Lab test management
- Notification system
- Reporting and analytics
- User management

2.3 User Classes and Characteristics

1. Admin

- **Characteristics**: Oversees the system, manages user accounts, and ensures platform operation.
- **Requirements**:
    - User Management: Create, edit, delete accounts, and assign roles.
    - System Management: Manage states, cities, tests, prices, and generate reports.

2. Lab Assistants

- **Characteristics**: Manages appointments and performs tests.
- **Requirements**:
    - Appointment Management: View and manage assigned appointments.
    - Test Management: Update test results and manage records.

3. Users

- **Characteristics**: Creates and manages appointments.
- **Requirements**:
    - Appointment Creation: Register, log in, select state and city, choose tests, create appointments.

- Appointment Management: View and track appointments.

2.4 Operating Environment

- **ASP.NET Web API:** Deployed on Windows Server with IIS or can be hosted in cloud environments and containerized solutions.
- **Spring Boot**: Deployed on Linux/Windows with Tomcat.

The operating environment for the MedLab consists of the following hardware and software components:

- **Hardware:**
  - A machine with at least 8GB of RAM and a fast processor, such as Intel Core i5 or higher, to ensure smooth and efficient execution of the project.
- **Software:**
  - **Frontend Development:** ReactJS for building the user interface.
  - **Backend Development:** ASP.NET Web API for server-side logic and HTTP services..
  - **Database Management:** MySQL for storing and managing data.
  - **Authentication:** JWT for secure user authentication.
  - **Deployment:** AWS for cloud deployment and scalability.
- **Other Applications:**
  - Code editor (e.g., Visual Studio Code, JetBrains Rider)
  - Git version control software
  - Command line interface (CLI) or terminal
  - A web browser for testing and interacting with the application
  - Postman for testing APIs
  - MySQL Workbench or another database management tool
  - An AWS account for deployment

2.5 Design and Implementation Constraints

- Must comply with data protection regulations (e.g., GDPR, HIPAA).
- Ensure compatibility with various web browsers.
- Use secure coding practices to prevent vulnerabilities.

2.6 Assumptions and Dependencies

- Users have access to the internet.
- Laboratory equipment can integrate with the system.
- Users are familiar with basic web navigation.

2.5 Design and Implementation Constraints

- **User Interface Language:** The user interface is available only in English; no other language options are provided.
- **System Load:** High traffic and parallel operations may affect system performance and responsiveness.
- **Security Protocols:** Limited to HTTP for secure data transmission.
- **Integration Challenges:** Potential compatibility issues when integrating frontend with backend systems.

2.6 User Documentation

User documentation will include an in-app Help menu that provides detailed instructions on using the application, including how to rent or list cars, manage bookings, and navigate the interface. The comprehensive documentation will also cover system functionalities, user roles, and technical requirements.

2.7 Assumptions and Dependencies

- **Assumptions:**
    - Users have a valid email address for notifications and account verification.
    - Users are familiar with basic car rental concepts and processes.

- Car owners have proper vehicle documentation and meet platform standards for car listings.

- **Dependencies:**
  - A functional email service is required for sending notifications and alerts.
  - Integration with payment gateways for processing transactions.
  - Reliable internet connection is necessary for accessing the application and receiving updates.
  - AWS infrastructure for hosting and scaling the application.
  - ReactJS, ASP.NET WEB API, and MySQL components are essential for the operation of the application.

3. System Features

3.1 User Login

3.1.1 Description and Priority

- **Description**: The User Login feature allows users to access the system by providing their credentials (username and password). This feature is crucial for maintaining system security by ensuring that only authorized users can access specific functionalities and data.
- **Priority**: High

3.1.2 Stimulus/Response Sequences

1. **Stimulus**: The user navigates to the login page and enters their username and password.
    - **Response**: The system validates the credentials. If valid, the user is redirected to their dashboard. If invalid, an error message is displayed.

3.1.3 Functional Requirements

1. **FR-1**: The system shall provide a login page where users can enter their username and password.
2. **FR-2**: The system shall validate the user credentials against stored data.
3. **FR-3**: The system shall redirect the user to the appropriate dashboard upon successful login.
4. **FR-4**: The system shall display an error message if the credentials are invalid.

3.2 User Registration

3.2.1 Description and Priority

- **Description**: The User Registration feature allows new users to create an account by providing their details. This includes selecting a state and city, and creating a username and password. It ensures users can access the system and use its features based on their role.
- **Priority**: High

3.2.2 Stimulus/Response Sequences

1. **Stimulus**: The user navigates to the registration page and fills out the registration form.
    - **Response**: The system validates the input and creates a new user account if all information is correct. The user is then redirected to the login page.
2. **Stimulus**: The user selects a state from a dropdown menu.
    - **Response**: The system dynamically updates the city dropdown based on the selected state.

3.2.3 Functional Requirements

1. **FR-1**: The system shall provide a registration page with fields for username, password.
2. **FR-2**: The system shall validate the input fields and check for duplicates (e.g., existing usernames).
3. **FR-3**: The system shall dynamically update the city dropdown based on the selected state.
4. **FR-4**: The system shall create a new user account and store the user details in the database upon successful registration.
5. **FR-5**: The system shall redirect the user to the login page after successful registration.

3.3 Appointment Management

3.3.1 Description and Priority

- **Description**: The Appointment Management feature allows users to create, view, and manage laboratory appointments. It assigns appointments to lab assistants based on the department and provides appointment details to both lab assistants and administrators.
- **Priority**: High

3.3.2 Stimulus/Response Sequences

1. **Stimulus**: The user creates a new appointment by filling out the appointment form.
    - **Response**: The system assigns the appointment to the relevant lab assistant based on the department and sends notifications to both the lab assistant and the user.
2. **Stimulus**: The user or lab assistant views the list of appointments.
    - **Response**: The system displays the appointments in a list, showing relevant details such as appointment date, time, and department.

3.3.3 Functional Requirements

1. **FR-1**: The system shall provide a form for users to create new appointments, including selecting a department and entering details.

2. **FR-2**: The system shall automatically assign the appointment to a lab assistant based on the selected department.
3. **FR-4**: The system shall allow users and lab assistants to view a list of appointments with relevant details.

3.4 Lab Assistant Dashboard

3.4.1 Description and Priority

- **Description**: The Lab Assistant Dashboard provides lab assistants with a personalized view of their appointments, allowing them to manage and track their assigned tasks. It helps them stay organized and informed about their responsibilities.
- **Priority**: Medium

3.4.2 Stimulus/Response Sequences

1. **Stimulus**: The lab assistant logs into the system and accesses their dashboard.
    - **Response**: The system displays a dashboard with a list of assigned appointments, including details such as appointment time, department, and patient information.
2. **Stimulus**: The lab assistant updates the status of an appointment.
    - **Response**: The system updates the appointment status and reflects the change on the dashboard.

3.4.3 Functional Requirements

1. **FR-1**: The system shall provide a dashboard view for lab assistants showing their assigned appointments.
2. **FR-2**: The system shall allow lab assistants to update the status with the help of callback.
3. **FR-3**: The system shall display relevant appointment details on the dashboard.

3.5 Admin Role Management

3.5.1 Description and Priority

- **Description**: The Admin Role Management feature allows administrators to manage user roles and permissions within the system. It ensures proper access control and helps maintain the integrity of the system.
- **Priority**: High

3.5.2 Stimulus/Response Sequences

1. **Stimulus**: The administrator accesses the user management section.

- **Response**: The system displays a list of users with options to create, edit, or delete user accounts and assign roles.

2. **Stimulus**: The administrator assigns a new role to a user.
    - **Response**: The system updates the user's permissions based on the assigned role and confirms the change.

3.5.3 Functional Requirements

1. **FR-1**: The system shall provide an interface for administrators to manage user accounts, including creating, editing, and deleting users.
2. **FR-2**: The system shall allow administrators to assign and manage roles and permissions for users.
3. **FR-3**: The system shall ensure that changes to user roles and permissions are reflected throughout the system.
4. **FR**-4: The system shall allow administrator to add and update locations.

3.7 Data Security and Privacy

3.7.1 Description and Priority

- **Description**: The Data Security and Privacy feature ensures that user data and transactions are secure and comply with privacy regulations. It protects sensitive information and maintains system confidentiality.
- **Priority**: High

3.7.2 Stimulus/Response Sequences

1. **Stimulus**: The user submits sensitive data (e.g., personal information, payment details).
    - **Response**: The system encrypts the data during transmission and storage to ensure confidentiality and security.
2. **Stimulus**: An unauthorized access attempt is detected.
    - **Response**: The system logs the attempt and alerts administrators of the potential security breach.

3.7.3 Functional Requirements

1. **FR-1**: The system shall implement encryption for data transmission and storage to protect user information.
2. **FR-2**: The system shall adhere to privacy regulations and data protection standards.
3. **FR-3**: The system shall log and alert administrators of unauthorized access attempts.

4. External Interface Requirements

4.1 User Interfaces

- Responsive web design compatible with various devices.

4.2 Hardware Interfaces

- Integration with lab equipment for test results (future scope).
- **Server Side:**
    - The web application will be hosted on a web server that listens on the standard web port, port 5221 (HTTP) or port 7093 (HTTPS).
- **Client Side:**
    - Monitor Screen: The system will display information to users via the monitor screen, providing a visual representation of car rental details and management tools.
    - Mouse: The system will interact with mouse movements and clicks to enable data input, activate command buttons, and select options from menus.
    - Keyboard: The system will process keystrokes from the keyboard for data entry and navigation within the application.

4.3 Software Interfaces

- API integration for notifications (email, SMS).
- Database connectivity (SQL Server, MySQL).
- **Server Side:**
    - - An Apache or Nginx web server will handle requests from clients and interact with the application backend.
    - - A MySQL database will be used to store and manage data related to car rentals, customer information, and system configurations.
- Client Side:
    - The system will require an operating system capable of running a modern web browser that supports JavaScript and HTML5 for full functionality.

4.4 Communication Interfaces

Protocols:

- The system will use HTTP or HTTPS protocols to facilitate secure communication between the client and server. HTTPS will be used to ensure encrypted data transmission and enhance security.

5. Other Nonfunctional Requirements

5.1 Performance Requirements

The performance requirements of the project should outline the expected speed, reliability, scalability, and efficiency of the system. This may include the maximum response time for user actions, the expected uptime percentage, the ability to handle increasing numbers of users and submissions, and the resource usage of the system. These requirements should be realistic and achievable, taking into account factors such as hardware limitations and network bandwidth. They should also be measurable and verifiable, allowing the system to be tested and evaluated against the defined standards.

5.2 Safety Requirements

The system should be reliable and have minimal downtime to ensure that students can submit assignments and receive feedback in a timely manner. The user interface should be intuitive and user-friendly to prevent errors or misunderstandings while using the app. The app should have robust error handling mechanisms to deal with unexpected situations and prevent any harm to the user or loss of data.

5.3 Security Requirements

The project would include measures to protect the confidentiality, integrity, and availability of sensitive information such as students' assignments, grades, and personal information. Some of the security requirements are:

- **Authentication**: A secure authentication system that ensures only authorized users can access the system.
- **Authorization**: An authorization mechanism to determine what actions a user can perform within the system based on their role and permissions.
- **Encryption**: Data in transit and at rest should be encrypted to protect sensitive information from being intercepted or accessed by unauthorized parties.
- **Access Control**: A mechanism to control access to sensitive information within the system, including the ability to set permissions for different users and roles.

5.4 Software Quality Attributes

The software quality attributes for the project include the following:

- **Usability**: The user interface should be intuitive, easy to navigate and user-friendly, allowing users to easily submit and review assignments.
- **Reliability**: The system should be reliable and provide accurate results, ensuring that assignments are submitted and reviewed accurately and efficiently.

- **Performance**: The system should perform efficiently and respond quickly to user requests, allowing for fast and seamless assignment submissions and reviews.
- **Scalability**: The system should be scalable, allowing for an increasing number of users and assignments as the demand grows.
- **Maintainability**: The system should be maintainable, with easy-to-update code and well-documented processes, ensuring that updates and improvements can be made quickly and efficiently.
- **Security**: The system should have strong security features, protecting sensitive information and ensuring the privacy and confidentiality of user data.
- **Compliance**: The system should comply with relevant regulations and standards, ensuring that it operates within the bounds of the law and industry best practices.

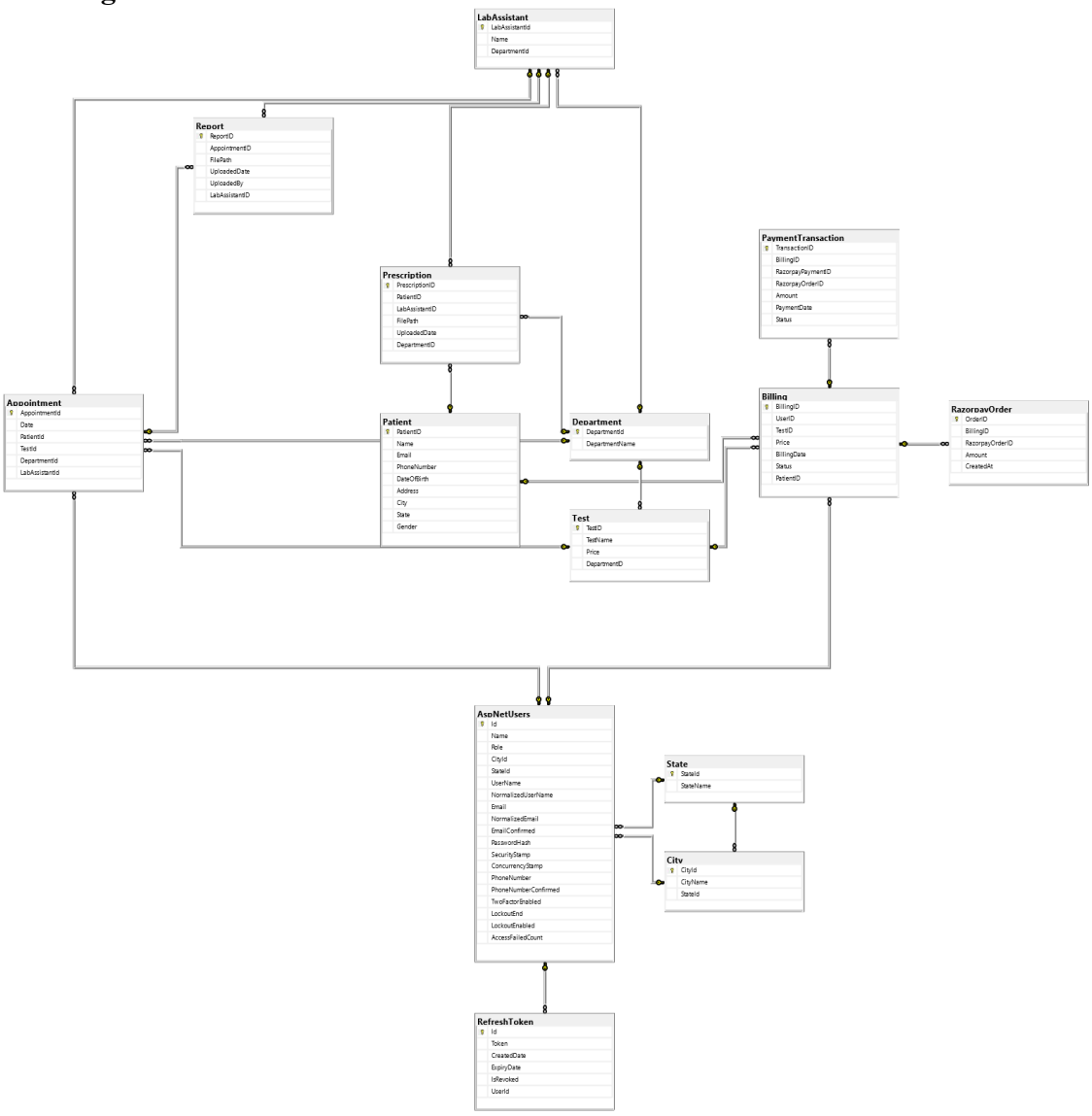
13. Appendices

Appendix A : Glossary

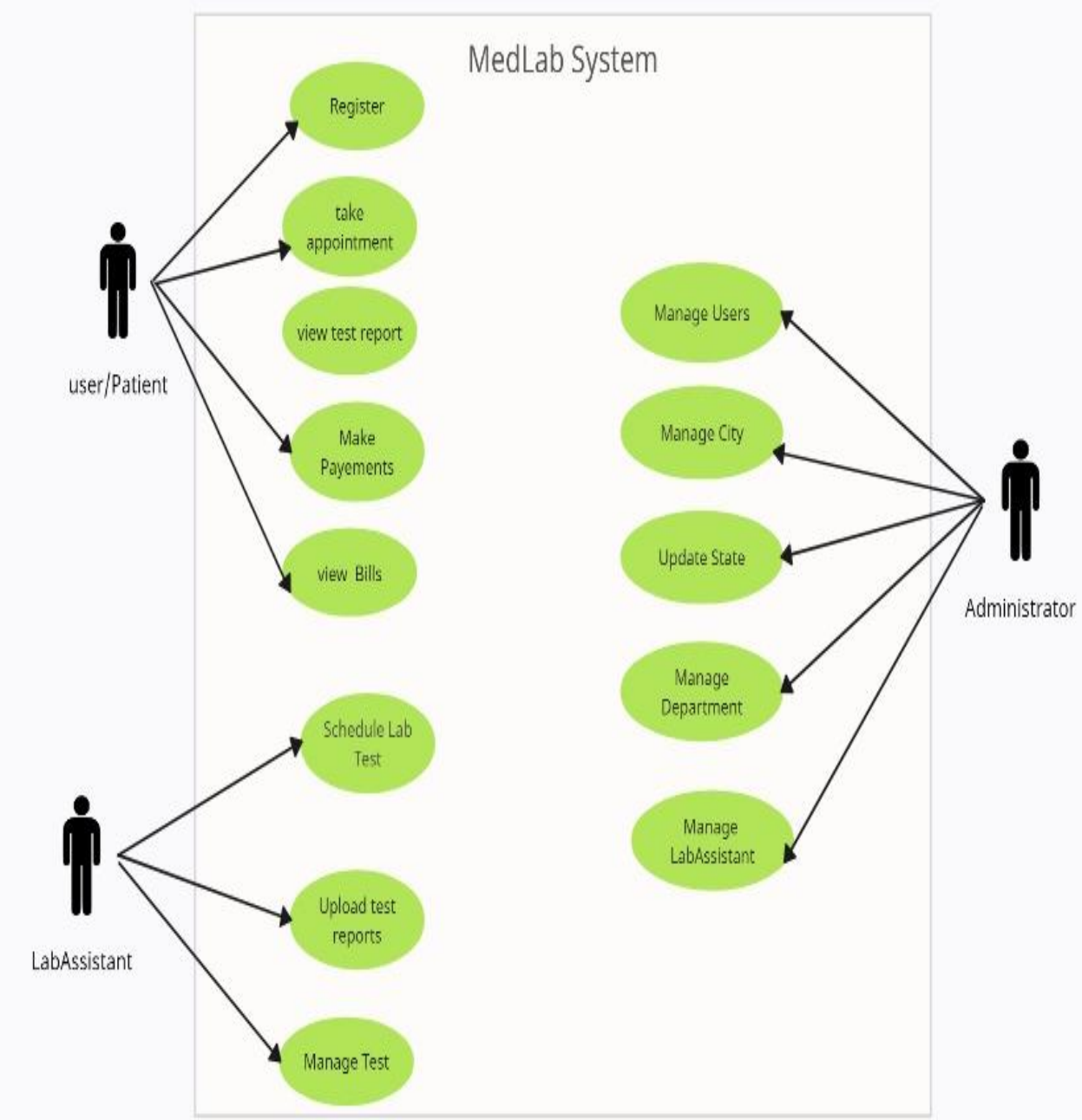
SR. NO	ABBREVIATION	FULL FORM
1.	API	Application Programming Interface
2.	AWS	Amazon Web Service
3.	CLI	Command line Argument
4.	GB	Gigabyte
5.	HTML	Hypertext Markup Language
6.	HTTP / HTTPS	Hypertext Transfer Protocol / Hypertext Transfer Protocol Secure
7.	ID	Identification
8.	JS	JavaScript
9.	JWT	Java Web Token
10.	OS	Operating System
11.	RAM	Random Access Memory
12.	SQL	Structured Query Language
13.	SRS	Software Requirement Specification
14.	URL	Uniform Resource Locator

Appendix B : Analysis Models

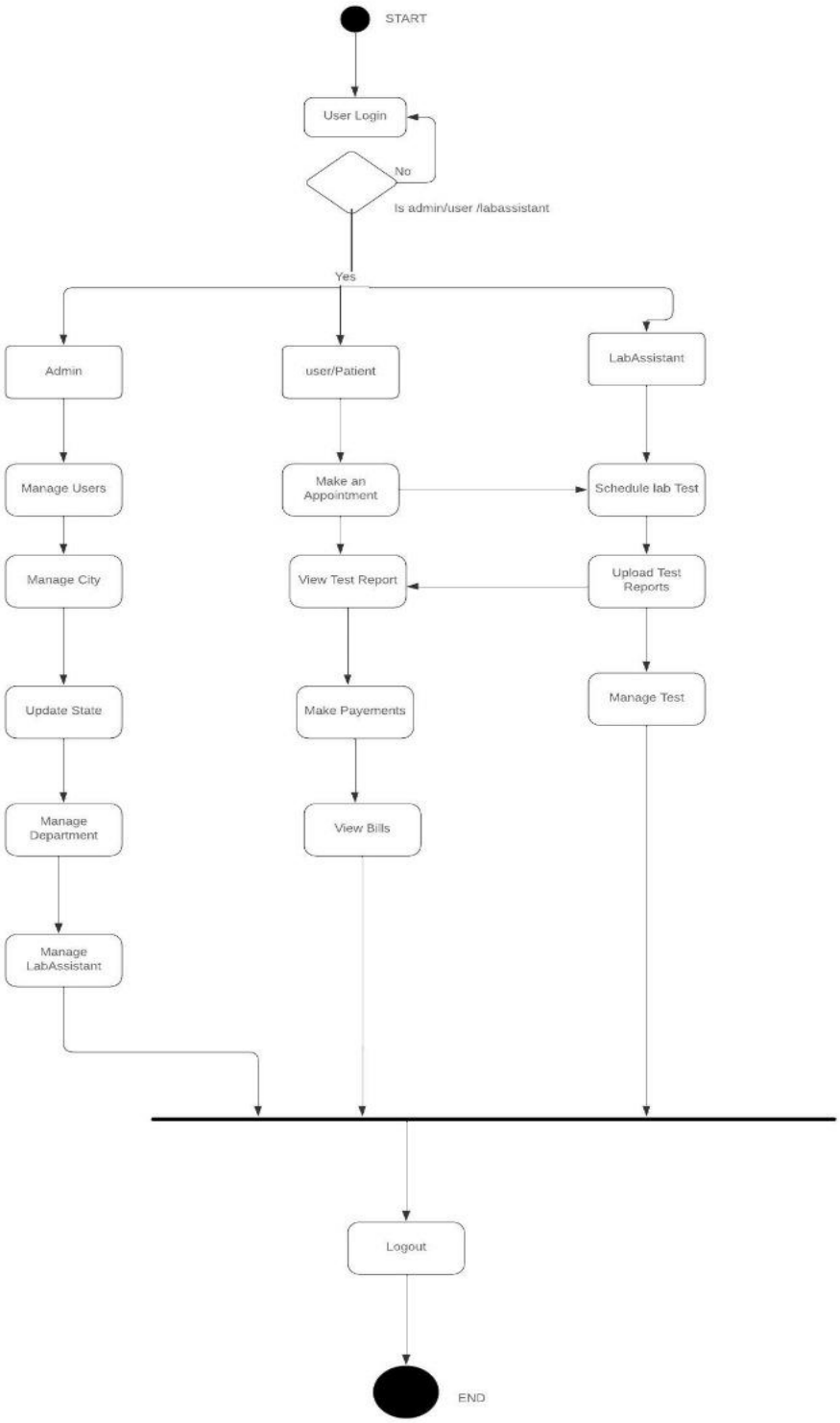
- **ER Diagram**



- **Use Case Diagram**



- **Activity Diagram**



- **Class Diagram**