Department of Computer Science

COMP4300- Graduation Project Report

Fall 2025/2026 Section – 2

Title of Project :

Birzeit University Insurance System

Group Members with IDs :

Jadallah Baragitha	**1210395**
Mousa Shuaib	**1210143**
Osaid Hamayel	**1211958**

Supervisor : Dr. Samer Zein

Date: 30/6/2025

Section – B

Title of Project: Birzeit University Insurance System

Project No: 2

Supervisor: Dr. Samer Zein

Key Areas: Healthcare Data Management ,Software Development and integration, Claims Processing, Data Privacy & Compliance, Software Integration, User Interfaces & Experience, Customer Support, Reporting.

Section – C

Student Signature: _____ **Date Submitted:**

First Supervisor Name: _____

First Supervisor Signature: _____**Date**

Approved: ____/ ____/ _____

Acknowledgments

We would like to express our sincere gratitude to our supervisor, Dr. Samer Zein, for his continuous support and guidance throughout the development of this project. His expertise and insight were invaluable to our learning journey and the successful completion of this healthcare insurance system.

We, Jadallah Baraghitha, Mousa Shuaib, and Osaid Hamayel, worked together as a team to complete this project. We are grateful for the opportunity to grow as a team, learn from each other, and collaborate to transform the Birzeit University Insurance System from a concept to a fully functional reality.

We also extend our thanks to the Department of Computer Science at Birzeit University and its faculty and staff for creating an excellent learning environment. Finally, we are thankful to our families and friends whose support and encouragement made this project possible.

Abstract

Healthcare insurance management in educational institutions often relies on fragmented, manual processes that involve collecting data across spreadsheets, paper forms, and disparate documents. This approach is time-consuming, error-prone, and delays critical processes like claims approval, medical record management, and healthcare provider coordination.

The Birzeit University Insurance System addresses these challenges by offering an integrated, web-based platform that automates the entire insurance workflow. The system connects multiple stakeholders—insurance managers, clients (students/employees), doctors, pharmacists, lab technicians, radiologists, emergency managers, medical administrators, and coordination administrators—through role-based dashboards with specific permissions and functionalities.

Key features include automated claims processing with multi-step approval workflows, real-time prescription and lab request tracking, healthcare provider network visualization with interactive maps, comprehensive medical record management, emergency request handling, and multi-language support (English/Arabic).

The system is powered by a modern, scalable technology stack: React 19.1 with Material-UI 7.3 drives a dynamic and responsive frontend, while the backend API provides secure authentication and data management. All data is securely stored with role-based access control ensuring data privacy. The architecture implements lazy loading, code splitting, and React Query for optimal performance, reducing initial bundle size by 60-80%.

Together, these technologies create a robust, user-friendly system that streamlines healthcare insurance management, reduces administrative overhead, and improves the overall experience for all stakeholders in the Birzeit University community.

Table of Contents

Contents

List of Figures

Chapter 1: Introduction

This chapter introduces the Birzeit University Insurance System project, highlighting the need for an automated healthcare insurance management system and outlining the project's aims and objectives.

1.1 Overview

Healthcare insurance management is an important factor in the overall wellness of communities within an institution, but most of these institutions still use processes that involve collecting information from spread sheets, documents, and forms. It is time-consuming, inaccurate, and results in late decisions in matters concerning the processing of healthcare, treatment, and coordination.

The Birzeit University Insurance System is a solution to these challenges. The system is a web-based insurance management platform, providing a comprehensive and integrated management system for insurance. The platform has the ability to automate data entry, use role-management methodologies aligned with healthcare industry practices, and provide real-time tracking and reporting. The platform is also designed for a variety of users, ranging from insurance managers, healthcare providers, to clients. The platform has been developed to make it possible for the university to efficiently administer health care benefits.

1.2 Aim

Briefly, the overall goal of the Birzeit University Insurance System project involves creating a comprehensive web application that will change how insurance management in healthcare services, especially in organizations, works. This application ensures that insurance services are made easier, faster, and more reliable through automated claim handling, management of medical records, tracking of prescriptions, and management of healthcare providers. The application enables all stakeholders to communicate through customized dashboards.

Moreover, the system has a centralized platform where the total insurance data, claims, medical information, and details of the healthcare providers can be stored and managed easily. The system also assists decision-makers in monitoring the status of the claims received, the utilization of the healthcare services, and the production of overall reports. The system is able to provide real-time notifications and interactive maps of the healthcare providers in addition to the provision of language translation (English and Arabic).

1.3 Objectives

The Birzeit University Insurance System project aims to achieve the following objectives:

- To develop a secure, role-based access control system supporting 9 distinct user roles with 113+ granular permissions
- To automate the claims processing workflow with multi-step approval (Medical Review → Coordination Review → Final Decision)
- To create comprehensive dashboards for each user role with relevant statistics and actionable insights
- To implement real-time tracking for prescriptions, lab requests, and radiology requests
- To provide an interactive healthcare provider network with map visualization using Leaflet
- To support chronic disease management and family member insurance coordination
- To enable emergency request handling with priority-based workflows
- To implement internationalization (i18n) supporting English and Arabic with RTL layout
- To ensure responsive design across desktop, tablet, and mobile devices
- To optimize performance through lazy loading, code splitting, and efficient state management

1.4 Overview of Technical Area

The Birzeit University Insurance System is created using a highly innovative and effective full-stack website development method. This system is created on the main technology of JavaScript/React, so it has a highly dynamic and responsive interface.

Front-End Environment :

In the frontend environment, React.js version 19.1 is applied along with Material-UI version 7.3 to develop a responsive interface using a component-based approach. This is achieved through efficient usage of React's Virtual DOM and its component-oriented architecture. Additionally, React Router 7.8 is applied for handling client-side routes along with lazy loading for better performance.

For state management and data retrieval, there is React Query 5.90, which supports efficient server-state management along with auto-caching, background refetching, and optimistic updates. Axios 1.13 is utilized for handling HTTP interactions with authentication token handling by using interceptors.

Leaflet and React-Leaflet are employed for interactive healthcare provider maps to provide location-based services. CSS styling is handled through Tailwind CSS and Emotion, as well as the Material-UI component library.

The development tools include Vite 7.1 as the build tool for rapid development and optimized build production, together with ESLint as the code quality check facility. The whole application also offers Progressive Web App functionality for improved mobile app experience.

1.5 Tools and Technology

1. **Backend: Spring Boot**

Spring Boot is a framework that simplifies Java web application development by automating configuration and reducing the need for manual setup. It is designed to create stand-alone, production ready applications with minimal code. Key features include fast bootstrapping,

auto-configuration, and embedded web servers. It supports easy integration with various tools, facilitates building RESTful APIs, and ensures scalability and strong security making it a reliable choice for backend development in systems like health insurance.

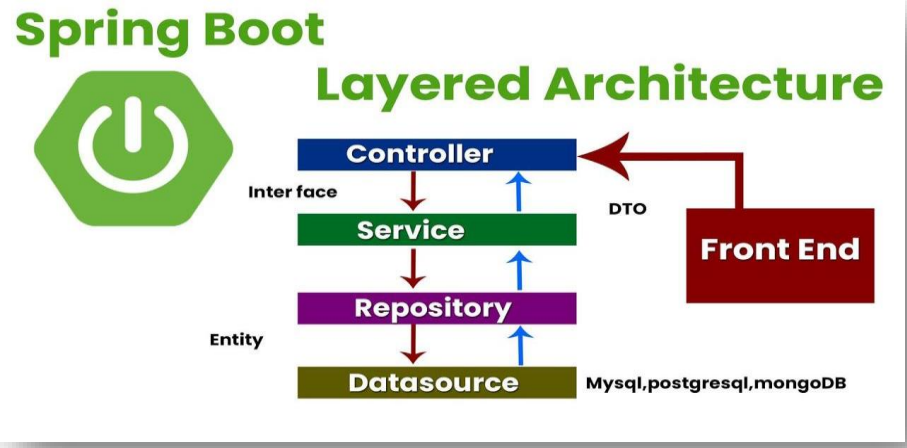

Figure 1: Spring Boot

2. **Web: React** 

React.js is a widely used JavaScript library for building user interfaces, particularly in dynamic and responsive web applications. It employs a component-based architecture, which simplifies the management of complex interfaces by breaking them down into smaller, reusable components. React utilizes a virtual DOM for efficient updates, enhancing performance by updating only the necessary components. It is also used for mobile app development with React Native and integrates seamlessly with npm (Node Package Manager) for managing third-party packages.

3. **Database: PostgreSQL** 

PostgreSQL is an advanced open-source Object-Relational Database Management System (ORDBMS) that handles complex queries and large datasets. It supports ACID transactions, ensuring data integrity, and is ideal for managing structured data in applications like health insurance systems. As an open-source project, PostgreSQL allows users to modify the source code to suit their needs, making it a versatile and reliable choice for various database management tasks. [5]

4. **Cloud: AWS** 

AWS (Amazon Web Services) is a scalable cloud computing platform offering services like EC2 for computing, S3 for storage, RDS for databases, and Lambda for serverless computing.

It provides a flexible, cost-effective, and reliable infrastructure ideal for building and scaling systems, such as health insurance applications. AWS ensures high availability and performance while managing costs efficiently.

5. **Testing API :** 

Postman is an API testing tool that enables developers to send requests to servers, inspect

responses, and verify backend functionality. It supports HTTP methods like GET, POST, PUT, and DELETE, and allows for dynamic parameters and headers. Postman simplifies the process of automating API tests, integrating them into CI/CD pipelines, and monitoring API performance.

Its user-friendly interface and features, such as response analysis and test collections, make it an essential tool for ensuring API reliability and quality in software development.

6. **Version Control:** 

Git is a distributed version control system (DVCS) that helps developers track and manage changes in project files, allowing each developer to have a complete copy of the project's history. Git repositories store project files and their revision histories, enabling easy collaboration and independent work.

GitHub is a web-based platform that hosts Git repositories and offers collaboration tools like issues, pull requests, and code review features.

Common Git commands, such as git init, git add, git commit, git push, and git pull, allow developers to manage and update their code. GitHub enhances collaboration by offering a platform where developers can create branches, propose changes, and merge updates through the GitHub flow.

Chapter 2: Background and Literature Review

2.1 Introduction

Healthcare insurance management is an important factor in providing for the health and well-being of university communities. It assists in offering necessary medical benefits and dealing with claims and coordination between an institute and health service providers. The rising costs of healthcare and demands for technologically advanced services have made it necessary for academic institutes to have an efficient system for handling their insurance operations.

Conventionally, the business of healthcare insurance operations has been done using manual methods such as paper forms, Excel sheets, and approval cycles. Notably, these methods are prone to taking a lot of time and are susceptible to human error, especially when the process involves claims for various healthcare providers. In fact, the lack of a central digital platform makes it challenging for the monitoring of claim statuses, patient records, and the timely response of various parties.

An online insurance management solution will be able to seamlessly meet the above challenges in a efficient way by automating the process of claims, ensuring access to information based on roles, as well as facilitating instant communication between the involved parties.

2.2 Background on Healthcare Insurance Systems

Health insurance schemes at institutions of higher learning form the core aspect in ensuring students and staff receive health covers for different purposes. Health insurance schemes must have the ability to handle complex workflows as in the following procedures:

- Claims Administration: Receiving, processing, and evaluating (accepting or rejecting) claims for Medical Care
- Provider Network Management: This function is engaged in forming and keeping connections with doctors, pharmacies, laboratories, and radiology centers.
- Medical Records: This involves recording the patient's medical history, prescription, and treatment.
- •Policy Administration: This includes the rules under which a contract will provide protection, deductibles, and limits of benefits.
- Financial Reporting: To monitor the cost, record the cost, and approximate cost.

The level of intricacy involved in these processes also increases the need for data privacy and regulatory issues, thus making the argument for handling these processes manually even more difficult and unrealistic. In contemporary institutions, there also emerges a need for comprehensive systems capable of managing such different needs in a way that promotes transparency and accessibility by all parties involved.

2.3 Traditional Insurance Management Methods and Limitations

Traditional healthcare insurance management in educational institutions typically relies on:

Table 1: Comparison Between Traditional Evaluation and Our System

Feature	Traditional Evaluation	Our System
Claims Submission	Paper forms, email attachments	Online claim submission through a unified digital portal with automated validation, document upload, real-time status tracking, and instant confirmation notifications.
Claims Review	Manual review, spreadsheet tracking	Automated claims review with centralized workflows for faster, consistent decisions and reduced human error.
Provider Coordination	Phone calls, faxes	Integrated digital platform for real-time communication with healthcare providers, ensuring faster responses, standardized data exchange, and more accurate information.
Medical Records	Paper files, isolated databases	Centralized digital records, easy access, secure, integrated
Reporting	Manual compilation from multiple sources	Time-consuming, inaccurate, outdated data
Client Communication	Email, in-person visits	real-time, centralized communication with instant notifications.

All the above-mentioned conventional techniques have some disadvantages, which are listed below:

- Highly Time-Consuming: It involves manual processing with respect to the claim, which may be taking days to weeks.
- Human Mistake: Human mistakes are considered in data entry and also in the calculation processes.
- Lack of Transparency. It means the failure to understand by the client about the status of his/her file at all times.
- Scattered Data: Generally speaking, this is data that has been dispersed into other systems one way or another.
- Poor Scalability. This means the approach does not support scalability when there is an increase in users.
- Confidentiality issues: Where confidentiality exists, so too do problems with hard copy files and unencrypted emails

2.4 Role of ICT and Web Applications in Insurance Management

It integrates Information and Communication Technology into insurance management, which is usually a time-consuming and error-prone operation, making it well-organized and smooth. In the context of healthcare insurance, ICT offers:

2.4.1 Importance of ICT in Healthcare Insurance

- Various digital forms and templates that enable collecting data in a structured way
- Automation of processes on basis of rules and roles predefined.
- Secure storage and access to claims, medical records, and provider data
- Real-time dashboards that help in quicker decision making
- Mobile accessibility: This allows management on-the-go.

However, those characteristics make ICT technology absolutely necessary in healthcare insurance because it can increase the efficiency of operations, the accuracy of data, and compliance with specified standards in health insurance organizations to provide fast and accurate services to healthcare providers and beneficiaries.

2.4.2 Benefits of Web-Based Systems

Benefit	Description
Accessibility	Access from any device with internet connection
Real-time Updates	Instant notifications and status tracking
Data Integrity	Centralized database with validation rules
Scalability	Easily handles growing user bases and data volumes
Security	Role-based access, encryption, audit trails
Cost Efficiency	Reduced administrative overhead and paper costs

2.4.3 Application in Our System

The Birzeit University Insurance System is characterized by intensive involvement of ICT in the system structure, and ICT is used in the system in

- **Front-end**: Created using React.js and Material-UI for interactive and responsive user interfaces.
- **State Management**: Using React Query for optimizing data retrieval, caching, and background updates.
- **Internationalization**: Complete support for both the English and Arabic languages.
- **Backend**: It is created using Spring Boot technology. Spring Boot is used to deal with business logic, routing, as well as API services.
- **Database**: PostgreSQL for a structured and reliable storage system.
- **Security**: JWT authentication and Role-Based Access Control to ensure security.

These technologies in the Birzeit University Insurance System enable the user to:

- Input medical, insurance, and/or client information using forms.
- Use rules and validation to process correctly the claim or record.
- Share documents by uploading them and linking them to customer profiles or claims
- View Results Instantly, Keep Track of Statisfactions, and Download Real-Time Outputs

2.4.4. Enhancing Insurance Management with Real

One of the most beneficial features of web-based systems such as the Insurance System developed by Birzeit University is that it is capable of providing real-time analyses. In contrast to the traditional method of paperwork, in which data compilation is done manually and at a slower pace, live dashboards that update instantly as data is entered are provided. Consequently, such a system allows the university to:

- Keep track of the real-time status changes of insurance claims and update
- Compare and monitor coverage, claims, and use through departments or time periode.
- Determine immediate problems or inconsistencies requiring action
- Produce reports and audits efficiently and effectively

With the integration of technology in the heart of insurance management processes, this system provides not only a solution using software but a smarter, faster, and more reliable approach in insurance operations management.

2.5 Standards and Frameworks for Healthcare Data

Insurance management at Birzeit University should be conducted in accordance with specific standards and guidelines in order to ensure accuracy, consistency, and congruence with the university and the state. Insurance management requires structuring with domains, indicators, and automated processes, allowing staff to judge, control, and track insurance in a standardized and integrated manner.

2.5.1 Data Privacy Standards

- Principles from HIPAA regulations: Protecting confidentiality, integrity, and security of sensitive data.
- Data minimization involves processing a claim and a policy using no more than the amount of data necessary to carry out such a function.
- Role-Based Access Control: Only data that pertains to their role can be viewed or changed by users.
- Audit trailsinterfaces as the logging of all accesses and changes for accountability.

2.5.2 Our Implementation Approach

The Insurance System in Birzeit University includes the following:

- Role Based Access Control (RBAC): has 8 roles and more than 113 permissions
- Token-Based Authentication – Expiring JWT Tokens and Securing Token Storage
- Data Validation and Policy Updates
- Secured Routes: Secured routes and fallback UI in case of unwanted accesses

2.6 Related Work and Existing Systems

There are a number of institutions, such as Birzeit University, which have traditionally dealt with insurance matters as well as claims manually or semi-digitally. Although such systems manage the basic functionality adequately, they might lack efficiency, accuracy, as well as a level of scalability. This is because the Insurance System tackles this problem as it is a totally digital and integrated system.

Several healthcare insurance management solutions exist in the market:

System Type	Characteristics	Limitations
Commercial Insurance Platforms	Comprehensive features, professional support	High cost, may not fit educational context
Generic ERP Systems	Broad functionality	Not specialized for healthcare insurance
Manual/Spreadsheet Systems	Low initial cost, familiar tools	Not scalable, error-prone, no automation
Custom In-house Solutions	Tailored to specific needs	High development cost, maintenance burden

2.6.1 Gaps in Existing Solutions

- Lack of integration between claims, medical records, and provider management
- Limited support for educational institution-specific workflows
- No multi-language support for Arabic-speaking users
- Insufficient role granularity for complex organizational structures

2.7 How Our System Addresses the Gaps

The Birzeit University Insurance System was developed particularly to bridge such gaps:

2.7.1 Unified Platform

Every insurance operation—claims handling, record-keeping, provider relations, notifications—is now processed from within a single and consistent user experience.

2.7.2 Role-Based Workflows

There are nine different user levels with corresponding dashboards, permissions, and workflows to provide each interested party just the right amount of access.

2.7.3 Real-Time Tracking

Clients can monitor their claims, medication, and requests in a live setting using their customized dashboard.

2.7.4 Healthcare Provider Network

The presence of an interactive map and the capabilities of the system's filters make it possible to find the availability of network providers such as (doctors, pharmacies, labs, radiology centers).

2.7.5 Multi-Language Support

There is also full support for the English and Arabic languages with RTL layout.

2.7.6 Modern, Responsive Design

It integrates smoothly on computers, tablet computers, and smartphones with the use of Material-UI components.

2.7.7 Summary

In conclusion, the Insurance System of Birzeit University directly deals with the inadequacies of conventional insurance methods and existing insurance solutions by incorporating an integrated interface, role-based processes, real-time monitoring, and multiple language support in an integrated package. The solution allows insurance staff, as well as their clients, to take advantage of efficient claim handling, proper record maintenance, and hassle-free interaction with providers without relying on manual processes.

Chapter 3: System Analysis and Design

This chapter details the system model and architecture of the Birzeit University Insurance System, using various diagrams and specifications to provide a comprehensive view of the system's design and implementation.

3.1 Product Description

The Birzeit University Insurance System is a web-based platform that automates healthcare insurance management for educational institutions. It streamlines operations by providing real-time, role-based access to claims, medical records, and healthcare provider information. With an intuitive interface and comprehensive workflows, the system helps the university efficiently manage healthcare benefits and improve service delivery.

3.1.1 System Objectives

The system aims to solve challenges faced by institutions in manually managing healthcare insurance. The system will:

- Automate the entire claims processing workflow, making it faster and more efficient
- Provide role-specific dashboards with relevant statistics and actionable insights
- Enable real-time tracking of prescriptions, lab requests, and radiology requests
- Support chronic disease management and family member coordination
- Facilitate emergency request handling with priority-based workflows
- Generate comprehensive reports for financial and operational analysis

3.1.2 System Main Features

The platform incorporates several core features:

Claims Management

- Multi-step approval workflow (Submitted → Medical Review → Coordination Review → Final Decision)
- Role-specific claim handling (providers vs. clients)
- Claim history and status tracking
- Document attachment and evidence management

Medical Request Management

- Prescription creation and fulfillment tracking
- Lab test request management
- Radiology/imaging request handling
- Emergency request processing

Healthcare Provider Network

- Provider search and filtering by type (doctor, pharmacy, lab, radiology)
- Interactive map visualization using Leaflet
- Provider profile management and registration
- Location-based services

User Management

- Multi-role user registration with approval workflows
- Profile management per role
- Chronic disease tracking
- Family member management

Notifications System

- Real-time notifications for all users
- Role-specific alerts
- Emergency notifications with priority

Reporting & Analytics

- Claims reports with status breakdown
- Financial reports and summaries
- Provider performance metrics
- Dashboard statistics

Internationalization

- English and Arabic language support
- RTL layout support
- Language persistence across sessions

3.1.3 Operating Environments

Development Environment

- Language: JavaScript (ES6+)
- Framework: React.js 19.1
- UI Library: Material-UI 7.3
- Build Tool: Vite 7.1
- State Management: React Query 5.90
- Development Tools: Visual Studio Code, Git/GitHub

Client/User Environment

- Supported Browsers: Google Chrome, Mozilla Firefox, Microsoft Edge, Safari
- Devices**: Desktop, Laptop, Tablet, and Mobile Devices
- Operating Systems: Windows, macOS, Linux, Android, iOS
- Network Requirements: Stable internet connection for real-time operations

Additional Dependencies

- Node Package Manager (npm) for managing dependencies
- Axios for API communication
- Leaflet for map visualization
- React Router for client-side routing
- Emotion and Tailwind CSS for styling

3.1.4 Constraints

1. Internet Connectivity: The system requires a stable internet connection for all operations
2. Browser Compatibility: Users must use modern browsers that support ES6+ JavaScript
3. Backend Dependency: The frontend relies on backend API availability for data operations
4. Data Security: All data transmission must be secured, and user authentication is required
5. Role Authorization: Users can only access features permitted by their assigned role

3.1.5 Functional Requirements

The system is designed to support healthcare insurance management using a structured, role-based workflow:

Authentication & Authorization

- Users can register with appropriate role selection
- Users must authenticate to access the system
- Role-based access control restricts features based on user permissions
- Session management with JWT tokens

Claims Processing

- Clients can submit claims with supporting documents
- Healthcare providers can submit claims for services rendered
- Medical administrators can review claims for medical appropriateness
- Coordination administrators can review claims for policy compliance
- Insurance managers can make final approval/rejection decisions

Medical Requests

- Doctors can create prescriptions, lab requests, and radiology requests
- Pharmacists can view and fulfill prescriptions
- Lab technicians can process lab requests and enter results
- Radiologists can process radiology requests and enter results
- Clients can view their requests and track status

Healthcare Provider Management

- Providers can register and manage their profiles
- Insurance managers can approve/reject provider registrations
- All users can search and view approved providers on maps

3.1.6 Non-Functional Requirements

Usability

- The system should provide an intuitive interface requiring minimal training
- The system should display information in clear, organized dashboards
- The system should notify users of important updates in real-time
- The system should support both English and Arabic languages

Scalability

- The system should handle multiple concurrent users without performance degradation
- The component architecture should support easy addition of new features

Availability

- The system should be accessible 24/7 for user operations
- The system should gracefully handle API unavailability with appropriate error messages

Performance

- The system should load initial pages within 3 seconds
- The system should respond to user interactions immediately
- Lazy loading should reduce initial bundle size by 60-80%

Security

- All authentication should use secure JWT tokens
- Input sanitization should prevent XSS attacks
- Role-based access should prevent unauthorized feature access

3.2 Functional Decomposition

3.2.1 Actors

Table 1: Primary actors and use cases

Primary Actor	Use Cases	Purpose
Insurance Manager	Manage policies, approve claims, manage users, view reports, manage providers	Full system administration and oversight
Medical Administrator	Review claims medically, manage chronic patients, handle emergencies	Medical oversight and claim review
Coordination Administrator	Review claims for policy compliance, coordinate approvals	Policy compliance and coordination
Client	Submit claims, view medical records, track requests, manage family	Personal insurance management
Doctor	Create prescriptions, lab requests, radiology requests, submit claims	Medical service provision
Pharmacist	View and fulfill prescriptions, submit claims	Prescription fulfillment
Lab Technician	Process lab requests, enter results, submit claims	Laboratory services
Radiologist	Process radiology requests, enter results, submit claims	Radiology services

3.2.2 Use Cases

Table 2: User Registration Use Case Specification

UC ID and Name	UC#1. User Authentication
Created By	Mousa Shuaib
Date Created	2026/1/17
Primary Actor	All Users
Secondary Actors	None
Trigger	The user wants to access the insurance system
Description	The user logs into the system using valid credentials to access role-based functionalities.
Preconditions	- The user must have a registered account.- The system must be available.
Postconditions	If successful: User is authenticated and redirected to their role-specific dashboard.If unsuccessful: Error message is displayed.
Normal Flow	1. User navigates to login page.2. User enters email and password.3. User submits login form.4. System validates credentials.5. System grants access and redirects user.
Alternative Flows	Invalid Credentials: System displays an error message.Empty Fields: System prompts user to fill required fields.
Other Information	Authentication is secured using role-based access control and encrypted credentials.

Table 3: Submit Claims Use Case Specification

UC ID and Name	**UC#2. Submit Claim**
Created By	Mousa Shuaib
Date Created	2026/1/26
Primary Actor	Doctors ,Pharmacist , Laboratoty , Radiology
Secondary Actors	Clients
Trigger	The client wants to submit a medical insurance claim
Description	The client submits a claim with medical details and supporting documents to request reimbursement or coverage.
Preconditions	- Client is logged into the system.- Client has valid insurance coverage.
Postconditions	If successful: Claim is submitted and marked as pending review.If unsuccessful: Error message is shown.
Normal Flow	1. Client logs into the system.2. Client navigates to claim submission page.3. Client enters claim details and uploads documents.4. Client submits the claim.5. System validates claim data.6. Claim is saved with pending status.
Alternative Flows	Missing Documents: System requests required attachments.Invalid Data: System prompts correction.
Other Information	Claim status can be tracked through the client dashboard.

Table 4: Review Claims Use Case Specification

UC ID and Name	**UC#3. Review Claim (Medical Admin)**
Created By	Osaid Hamayel
Date Created	2026/1/26
Primary Actor	Medical Administrator
Secondary Actors	Coordinator Admin
Trigger	A new claim requires medical review
Description	The Medical Admin reviews submitted claims to verify medical validity and necessity.
Preconditions	- Medical Admin is logged in.- Claim is in pending status.
Postconditions	If successful: Claim is approved or rejected medically.If unsuccessful: Error message is displayed.
Normal Flow	1. Medical Admin logs into the system.2. Medical Admin views pending claims.3. Medical Admin reviews claim details and documents.4. Medical Admin makes a decision.5. System updates claim status.
Alternative Flows	Insufficient Evidence: Claim is rejected with reason.System Error: Decision is not saved.
Other Information	Medical decision is required before coordination review.

Table 5: Create Prescription Use Case Specification

UC ID and Name	**UC#4. Create Prescription**
Created By	Osaid Hamayel
Date Created	2026/1/26
Primary Actor	Doctor
Secondary Actors	None
Trigger	Doctor needs to prescribe medication for a patient
Description	The doctor creates a prescription containing diagnosis and medication details.
Preconditions	- Doctor is logged in.- Patient exists in the system.
Postconditions	If successful: Prescription is created and stored.If unsuccessful: Error message is shown.
Normal Flow	1. Doctor logs into the system.2. Doctor selects a patient.3. Doctor enters diagnosis and medication details.4. Doctor submits prescription.5. System saves prescription and notifies patient.
Alternative Flows	Invalid Medication Info: System prompts correction.Missing Data: Submission blocked.
Other Information	Prescription becomes available to pharmacies immediately.

Table 6: Executing the Prescription Use Case Specification

UC ID and Name	**UC#5. Fulfill Prescription**
Created By	Jadallah Baragthi
Date Created	2026/1/26
Primary Actor	Pharmacist
Secondary Actors	None
Trigger	Pharmacist receives a prescription
Description	Pharmacist fulfills the prescription by dispensing medication and updating its status.
Preconditions	- Pharmacist is logged in.- Prescription exists and is valid.
Postconditions	If successful: Prescription is marked as fulfilled.If unsuccessful: Error message is shown.
Normal Flow	1. Pharmacist logs into the system.2. Pharmacist views pending prescriptions.3. Pharmacist selects a prescription.4. Pharmacist dispenses medication.5. System updates prescription status.
Alternative Flows	Medication Unavailable: Prescription marked partially fulfilled.Invalid Prescription: Action blocked.
Other Information	Fulfilled prescriptions can be used to submit insurance claims.

Table 7: Search Healthcare Providors Use Case Specification

UC ID and Name	**UC#6. Search Healthcare Providers**
Created By	Jadallah Baragthi
Date Created	2026/1/26
Primary Actor	Client
Secondary Actors	Admins
Trigger	User wants to find a healthcare provider
Description	Users search for healthcare providers using filters and map-based visualization.
Preconditions	- User is logged into the system.
Postconditions	If successful: List of providers is displayed.If unsuccessful: No results message is shown.
Normal Flow	1. User navigates to providers page.2. User selects provider type and filters.3. System displays providers on map and list.4. User views provider details.
Alternative Flows	No Providers Found: System displays empty result message.
Other Information	Map visualization uses location-based services.

3.2.3 Use Case Diagram

3.3 System Models

This section presents various system models that provide a deeper understanding of the Insurance system's architecture, structure, and behavior. These models offer different perspectives on the system, from the static relationships between classes to the dynamic interactions between components.

3.3.1 Class Diagram

The Class Diagram illustrates the static structure of the Insurance System, depicting the classes, their attributes, and the relationships between them. This diagram provides a visual representation of the system's object-oriented design, showing how different components are organized and how they relate to one another.

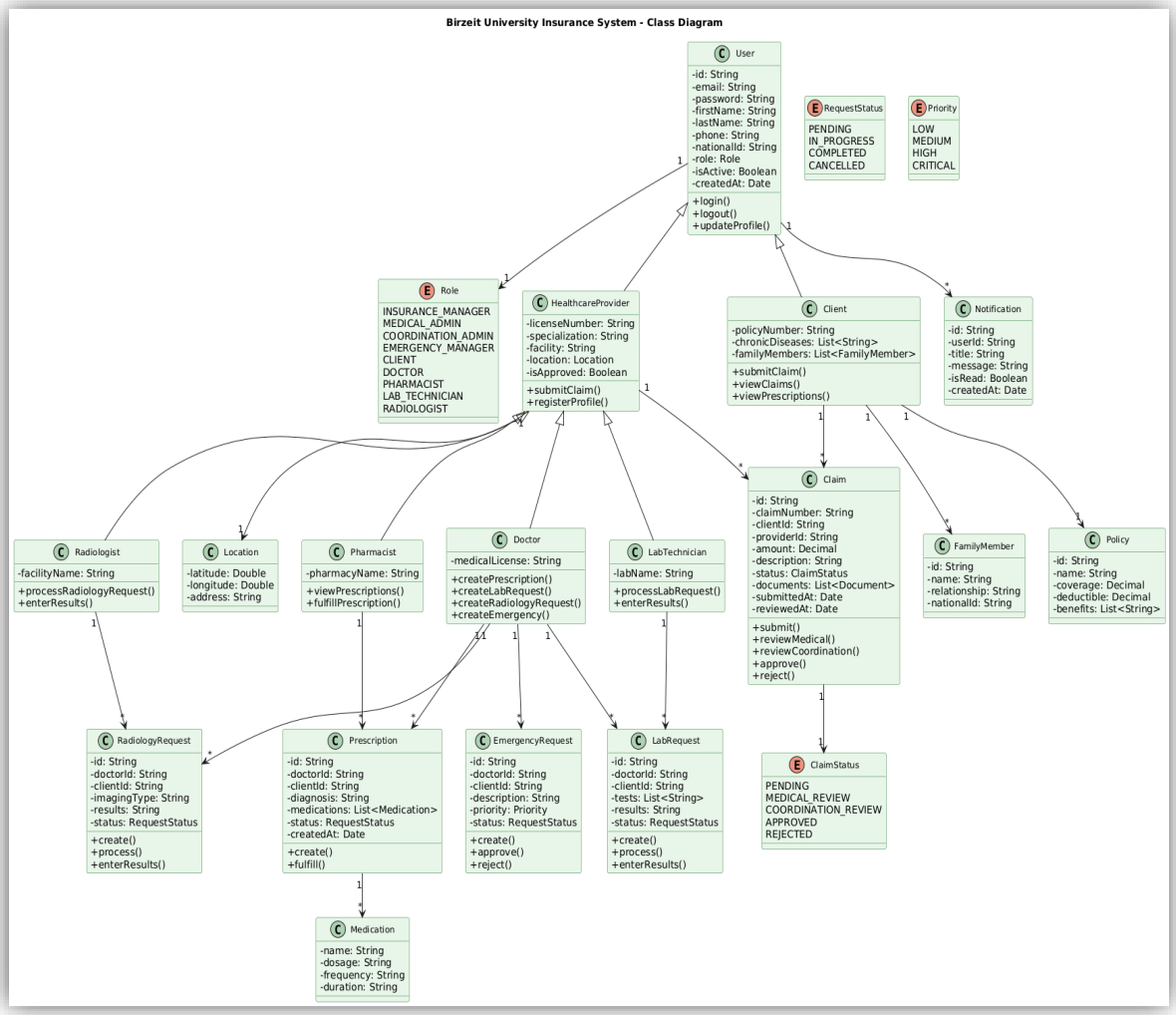

Figure 2: Class Diagram

3.3.2 Sequence Diagram - Claims Workflow

The Sequence Diagram below visualizes the dynamic behavior of the Insurance system for the use case. It illustrates the interactions between different actors and system components over time, showing the order of messages exchanged and the flow of control.

3.3.3 Sequence Diagram - Prescription Workflow

3.3.4 Activity Diagram - User Registration

This Activity Diagram provides a comprehensive overview of the Insurance system's workflow, encompassing all use cases and illustrating the interactions between actors and the system throughout the entire evaluation process.

3.3.5 Activity Diagram - Claims Processing

3.3.6 State Chart - Claim Status

This State Chart Diagram illustrates the various states and transitions of entities or processes within Insurance, visualizing its lifecycle and behavior in response to different events.

3.3.7 State Chart - Prescription Status

3.3.8 Component Architecture

3.3.2 State Flow Diagram (Claims)

3.4 System Architecture

This section describes the overall architecture of the Insurance system, outlining its key components, their interactions, and how they work together to achieve the system's objectives. The architecture is presented in terms of sub-systems, software layers, and deployment structure, providing a comprehensive view of the system's design.

3.4.1 Sub-Systems

Table 14: Main Sub-Systems

Subsystem	Description
Authentication Module	Handles user login, registration, password management, and session control
Claims Management	Manages claim submission, review workflows, approval/rejection, and tracking
Medical Requests	Handles prescriptions, lab requests, radiology requests, and emergency requests
User Management	Manages user profiles, role assignments, and permission control
Healthcare Provider Network	Maintains provider registry, search functionality, and map visualization
Notifications	Handles real-time notifications and alerts for all users
Reporting	Generates various reports for analysis and decision-making
Internationalization	Manages language switching and RTL support

3.4.2 Software Architecture

The system follows a layered architecture:

Presentation Layer (React Components)

- Material-UI components for consistent design
- Role-specific dashboards and layouts
- Responsive design for all devices
- Lazy loading for performance optimization

State Management Layer

- React Query for server state (API data)
- React Context for client state (theme, language)

- Local storage for authentication tokens

Service Layer

- API service with Axios interceptors
- Error handling and retry logic
- Token management

External Integrations

- Leaflet maps for provider visualization
- WebSocket support for real-time updates (SockJS, StompJS)

3.4.3 Technology Stack

Category	Technology	Purpose
****Frontend Framework****	React 19.1.1	Component-based UI development
****UI Library****	Material-UI 7.3.2	Pre-built, accessible components
****Routing****	React Router 7.8.2	Client-side navigation
****State Management****	React Query 5.90.16	Server state with caching
****HTTP Client****	Axios 1.13.1	API communication
****Maps****	Leaflet, React-Leaflet	Interactive maps
****Styling****	Tailwind CSS, Emotion	Utility-first and CSS-in-JS
****Build Tool****	Vite 7.1.2	Fast development and builds
****Icons****	MUI Icons, Lucide React	Iconography
****Animation****	Framer Motion	Smooth transitions
****Charts****	Recharts	Data visualization

3.4.4 Component Diagram

3.4.5 Deployment Diagram

Deployment Diagram illustrating the physical architecture of the Insurance system.

3.4.6 Entity Relationship Diagram

3.4.7 Data Flow Diagram (Context Level)

Chapter 4: System Implementation

4.1 Introduction

This chapter provides a detailed technical account of the construction of the Birzeit University Insurance System frontend. It serves as a bridge between the conceptual design phase and the final, functional software product. The primary objective is to elucidate the engineering decisions, architectural patterns, and specific technologies employed.

Starting with the big picture, we'll look at the system's overall design and technology choices. Then, we'll explore the frontend implementation, covering its component structure, state management, and features like multi-language support and theming. Each section explains not just what we built, but also why, always keeping performance, scalability, and ease of maintenance in mind.

4.2 System Architecture and Technology Stack

4.2.1 Client-Server Architecture

The Birzeit University Insurance System operates on a client-server model. The React frontend runs entirely in the user's browser, communicating with the backend API via HTTP requests. The server processes requests, performs necessary operations, and returns responses in JSON format.

This decoupled architecture provides significant advantages:

- Separation of Concerns: Frontend and backend development can proceed independently
- Scalability: The frontend can be deployed to CDN for global distribution
- Flexibility: The same API can serve web, mobile, or other clients
- Performance: Client-side rendering reduces server load

4.2.2 Technology Selection Justification

Technology	Selection Rationale
****React 19.1****	Latest version with improved performance, concurrent features, and large ecosystem
****Material-UI 7.3****	Comprehensive component library with accessibility, theming, and responsive design
****React Query 5.90****	Automatic caching, background refetching, optimistic updates, and devtools
****Vite 7.1****	Lightning-fast HMR, optimized builds, and modern ES module support
****Axios****	Request/response interceptors, automatic transforms, and wide browser support
****Leaflet****	Open-source, mobile-friendly maps with extensive plugin ecosystem

4.3 Frontend Implementation

4.3.1 Application Entry Point (main.jsx)

The application initializes with a carefully ordered provider hierarchy:

```
// Provider hierarchy in main.jsx
<React.StrictMode>
  <ErrorBoundary>
    <QueryClientProvider client={queryClient}>
      <LanguageProvider>
        <BrowserRouter>
          <ThemeProvider>
            <App />
          </ThemeProvider>
        </BrowserRouter>
      </LanguageProvider>
    </QueryClientProvider>
  </ErrorBoundary>
</React.StrictMode>
```

This structure ensures:

6. **Error Boundary**: Catches and handles React errors gracefully

7. Query Client: Provides React Query context for all components
8. Language Provider: Enables i18n throughout the application
9. Browser Router: Enables client-side routing
10. Theme Provider: Provides light/dark mode theming

4.3.2 Routing Configuration (App.jsx)

The routing system implements:

- 40+ Protected Routes: All authenticated routes wrapped in PrivateRoute
- Lazy Loading: Components loaded on demand using React.lazy()
- Code Splitting: Automatic bundle splitting per route
- Role-Based Redirects: Automatic redirect to appropriate dashboard

```
// Example lazy loaded route
const ManagerDashboard = lazy(() =>
import('./Component/Manager/ManagerDashboard'));

// Protected route wrapper
<Route path="/manager/dashboard" element={
  <PrivateRoute allowedRoles={['INSURANCE_MANAGER']}>
    <Suspense fallback={<PageLoader />}>
      <ManagerDashboard />
    </Suspense>
  </PrivateRoute>
} />
```

4.3.3 Role-Based Access Control (roles.js)

The RBAC system defines:

- 8 User Roles: Manager, Client, Doctor, Pharmacist, Lab, Radiologist, Medical Admin, Coordination Admin
- 113+ Permissions: Granular permissions for all system operations
- Role Aliases: Normalization for backend variations
- Helper Functions: `hasRole()`, `hasPermission()`, `canAccessRoute()`

```
// Permission mapping example
export const PERMISSIONS = {
  CLAIM_SUBMIT_OWN: 'claim:submit:own',
  CLAIM_SUBMIT_AS_PROVIDER: 'claim:submit:provider',
  CLAIM_VIEW_OWN: 'claim:view:own',
  CLAIM_VIEW_ALL: 'claim:view:all',
  CLAIM_REVIEW_MEDICAL: 'claim:review:medical',
  CLAIM_REVIEW_COORDINATION: 'claim:review:coordination',
  CLAIM_APPROVE: 'claim:approve',
  CLAIM_REJECT: 'claim:reject',
  // ... 100+ more permissions
};
```

4.3.4 Component Architecture

Components are organized by feature/role with consistent patterns:

Dashboard Components: Each role has a dedicated dashboard displaying:

- Quick statistics (total claims, pending requests, etc.)
- Recent activity
- Quick action buttons
- Healthcare provider map (where applicable)

Sidebar Components: Role-specific navigation with:

- Menu items based on permissions
- Active state highlighting
- Collapsible sections

Form Components: Consistent form handling with:

- Material-UI form controls
- Validation rules
- Error display
- Loading states

4.3.5 Internationalization (i18n)

The system supports English and Arabic with:

- **Translation Files**: 1000+ translation keys in `translations.js`
- **RTL Support**: Automatic right-to-left layout for Arabic
- **Language Persistence**: User preference stored in localStorage
- **Dynamic Text Direction**: Document direction changes with language

```
// Language context usage
const { language, setLanguage, t } = useLanguage();

// Translation usage
<Typography>{t('dashboard.welcome')}</Typography>
```

4.3.6 Theming

The theme system provides:

- **Light/Dark Modes**: User-selectable with persistence
- **Olive Green Healthcare Theme**: Custom primary color palette
- **Responsive Breakpoints**: Consistent across components
- **Component Overrides**: Customized Material-UI defaults

4.4 State Management and Data Fetching

4.4.1 React Query Implementation

React Query handles all server state with:

```
// Query client configuration
const queryClient = new QueryClient({
  defaultOptions: {
    queries: {
      retry: 2,
      staleTime: 5 * 60 * 1000, // 5 minutes
      cacheTime: 30 * 60 * 1000, // 30 minutes
    },
  },
});

// Query usage example
const { data: claims, isLoading, error } = useQuery({
  queryKey: ['claims', userId],
  queryFn: () => fetchUserClaims(userId),
});
```

4.4.2 API Service Layer

The API service provides:

- **Axios Instance**: Configured with base URL and defaults
- **Request Interceptors**: Automatic token attachment
- **Response Interceptors**: Error handling and token refresh
- **Token Management**: Secure storage and retrieval

```
// API service interceptor
api.interceptors.request.use((config) => {
  const token = getToken();
  if (token) {
    config.headers.Authorization = `Bearer ${token}`;
  }
  return config;
});
```

4.5 Security Implementation

4.5.1 Authentication

- **JWT Tokens**: Secure, stateless authentication
- **Token Storage**: localStorage with automatic attachment
- **Session Management**: Token expiration handling
- **Protected Routes**: PrivateRoute wrapper for all authenticated pages

4.5.2 Input Sanitization

- XSS Prevention: All user input sanitized before display
- Form Validation: Client-side validation before submission
- Error Handling : User-friendly error messages without sensitive data

4.5.3 Role-Based Security

- Permission Checks: UI elements hidden based on permissions
- Route Guards: Unauthorized access redirected appropriately
- API Authorization: Backend validates all requests

4.6 Application Screenshots

4.6.1 Landing Page

The public landing page features:

- Sign In and Sign Up tabs
- Feature showcase
- Team information
- Contact details
- Olive green healthcare theme

4.6.2 Manager Dashboard

The Insurance Manager dashboard displays:

- Total clients, policies, and pending claims statistics
- Healthcare provider network map
- Quick action buttons for common tasks
- Recent claims and notifications

4.6.3 Client Dashboard

The Client dashboard shows:

- Personal insurance status
- Active claims with status tracking
- Recent prescriptions and requests
- Healthcare provider search

4.6.4 Doctor Dashboard

The Doctor dashboard provides:

- Patient management interface

- Quick prescription creation
- Lab and radiology request forms
- Emergency request submission

4.6.5 Claims Management

The claims interface includes:

- Filterable claims list
- Status-based color coding
- Detailed claim view with documents
- Approval/rejection actions (for authorized users)

4.6.6 Healthcare Provider Map

The provider network map features:

- Interactive Leaflet map
- Provider type filtering
- Click-to-view provider details
- Location-based search

4.6.7 Prescription Management

Prescription interfaces include:

- Doctor: Creation form with medication selection
- Pharmacist: Pending prescription list with fulfill action
- Client: Personal prescription history

Chapter 5: Testing Phase Of Insurance System

5.1 Testing Overview

Testing ensures the Birzeit University Insurance System functions correctly and meets quality standards. The testing approach covers functional testing of all features, usability testing for user experience, and security testing for data protection.

5.2 List of Features to be Tested

11. User Authentication (Sign In, Sign Up, Password Reset)
12. Role-Based Access Control
13. Claims Submission and Management
14. Prescription Creation and Fulfillment
15. Lab Request Processing
16. Radiology Request Processing
17. Emergency Request Handling
18. Healthcare Provider Search and Map
19. Notifications System
20. Profile Management
21. Report Generation
22. Language Switching (i18n)
23. Theme Switching (Light/Dark)
24. Responsive Design

5.3 Test Cases

5.3.1 User Login Test Case

Field	Description
****Test Case ID****	TC-001
****Test Case Name****	User Login
****Objective****	Verify users can authenticate successfully
****Preconditions****	User has registered account
****Test Steps****	1. Navigate to login page 2. Enter valid email 3. Enter valid password 4. Click Sign In button
****Expected Result****	User is authenticated and redirected to role-specific dashboard
****Actual Result****	Pass

5.3.2 Role-Based Access Test Case

Field	Description
****Test Case ID****	TC-002
****Test Case Name****	Role-Based Access Control
****Objective****	Verify users can only access permitted features
****Preconditions****	User is logged in with specific role
****Test Steps****	1. Login as Client 2. Attempt to access Manager dashboard URL directly 3. Observe system response
****Expected Result****	User is redirected to their own dashboard or shown access denied
****Actual Result****	Pass

5.3.3 Claim Submission Test Case

Field	Description
****Test Case ID****	TC-003
****Test Case Name****	Client Claim Submission
****Objective****	Verify clients can submit claims successfully
****Preconditions****	Client is logged in
****Test Steps****	1. Navigate to Add Claim page 2. Fill required fields (date, amount, description) 3. Attach supporting document 4. Click Submit
****Expected Result****	Claim is created with PENDING status and appears in claims list
****Actual Result****	Pass

5.3.4 Prescription Creation Test Case

Field	Description
****Test Case ID****	TC-004
****Test Case Name****	Doctor Prescription Creation
****Objective****	Verify doctors can create prescriptions
****Preconditions****	Doctor is logged in
****Test Steps****	1. Navigate to Add Prescription 2. Search and select patient 3. Enter diagnosis 4. Add medications 5. Submit prescription
****Expected Result****	Prescription is created and visible to patient and pharmacies
****Actual Result****	Pass

5.3.5 Healthcare Provider Search Test Case

Field	Description
****Test Case ID****	TC-005
****Test Case Name****	Healthcare Provider Map Search
****Objective****	Verify users can search and view providers on map
****Preconditions****	User is logged in
****Test Steps****	1. Navigate to Healthcare Providers 2. Select provider type filter (e.g., Pharmacy) 3. View map results 4. Click on a provider marker
****Expected Result****	Filtered providers appear on map, clicking shows provider details
****Actual Result****	Pass

5.3.6 Language Switching Test Case

Field	Description
****Test Case ID****	TC-006
****Test Case Name****	Language Toggle (EN/AR)
****Objective****	Verify language switching works correctly
****Preconditions****	User is on any page
****Test Steps****	1. Click language toggle 2. Select Arabic 3. Observe UI changes

****Expected Result****	All text changes to Arabic, layout switches to RTL
****Actual Result****	Pass

5.3.7 Theme Switching Test Case

Field	Description
****Test Case ID****	TC-007
****Test Case Name****	Theme Toggle (Light/Dark)
****Objective****	Verify theme switching persists
****Preconditions****	User is logged in
****Test Steps****	1. Click theme toggle 2. Switch to Dark mode 3. Refresh page
****Expected Result****	Dark theme is applied and persists after refresh
****Actual Result****	Pass

5.3.8 Notifications Test Case

Field	Description
****Test Case ID****	TC-008
****Test Case Name****	Notification Display
****Objective****	Verify notifications appear for relevant events
****Preconditions****	User has notifications
****Test Steps****	1. Navigate to Notifications page 2. View notification list 3. Click on a notification
****Expected Result****	Notifications are displayed with correct content and timestamps
****Actual Result****	Pass

5.3.9 Responsive Design Test Case

Field	Description
****Test Case ID****	TC-009
****Test Case Name****	Mobile Responsiveness
****Objective****	Verify UI adapts to mobile screens
****Preconditions****	None
****Test Steps****	1. Open application on mobile device/emulator 2. Navigate through main pages 3. Test form interactions
****Expected Result****	All elements are visible and usable on mobile screens
****Actual Result****	Pass

5.3.10 Claims Workflow Test Case

Field	Description
****Test Case ID****	TC-010
****Test Case Name****	Full Claims Approval Workflow
****Objective****	Verify complete claims workflow from submission to approval
****Preconditions****	Multiple user accounts (Client, Medical Admin, Coordination Admin)
****Test Steps****	1. Client submits claim 2. Medical Admin reviews and approves 3. Coordination Admin reviews and approves 4. Client views approved claim
****Expected Result****	Claim status progresses correctly through all stages
****Actual Result****	Pass

References

25. React Documentation. (2025). React – A JavaScript library for building user interfaces. https://react.dev/
26. Material-UI Documentation. (2025). MUI: The React component library. https://mui.com/
27. React Query Documentation. (2025). TanStack Query - Powerful asynchronous state management. https://tanstack.com/query/
28. Vite Documentation. (2025). Vite - Next Generation Frontend Tooling. https://vitejs.dev/
29. React Router Documentation. (2025). React Router - Declarative routing for React. https://reactrouter.com/
30. Leaflet Documentation. (2025). Leaflet - An open-source JavaScript library for mobile-friendly interactive maps. https://leafletjs.com/
31. Axios Documentation. (2025). Axios - Promise based HTTP client. https://axios-http.com/
32. JWT.io. (2025). JSON Web Tokens. https://jwt.io/
33. OWASP Foundation. (2025). OWASP Top Ten Web Application Security Risks. https://owasp.org/www-project-top-ten/
34. Nielsen, J. (2023). Usability Engineering. Morgan Kaufmann Publishers.
35. Gamma, E., Helm, R., Johnson, R., & Vlissides, J. (1994). Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley.
36. Martin, R. C. (2017). Clean Architecture: A Craftsman's Guide to Software Structure and Design. Prentice Hall.