Infosys Springboard Virtual Internship 6.0
Completion Report

Team Details <Do not mention any personally identifiable information like email ID, institute details, mobile phone number etc.>

Batch Number: 5

Start date : 13-oct-2025

Names: Chidaraboina Nohitha

Internship Duration: 8 Weeks

1. Project Title

Infosys_InsurAI Corporate Policy Automation and Intelligence System

2. Project Objective

To build comprehensive web platform that automates the creation, management, and distribution of corporate insurance policies for employees and organizations. The system facilitates:

- Companies to manage their internal insurance policies efficiently
- Employees to view, claim, and compare available policies
- Agents to handle customer interactions and policy assignments
- Administrators to oversee the entire system with comprehensive dashboards

3. Project description in detail

1. Project Overview

InsurAI is a corporate insurance management system designed to streamline and automate insurance policy operations within an organization. The platform provides a unified interface where customers, agents, and administrators can manage policies, claims, appointments, and user information efficiently. The main goal is to reduce manual administrative overhead while improving service delivery and policy accessibility.

2. Objectives

- To automate insurance policy management and claims processing
- To simplify appointment scheduling between customers and insurance agents
- To manage insurance plan information dynamically with real-time updates
- To provide comprehensive admin dashboards for monitoring appointments, agents, and customer interactions
- To implement role-based access control for secure multi-user operations
- To ensure seamless communication between frontend and backend systems

3. Approach

Step-by-step Development Process:

1. Requirement Analysis

- Identified core business requirements: policy management, claims processing, agent scheduling, and user authentication
- Defined three primary user roles: Customer, Agent, and Administrator

- Documented functional requirements for each role and module

2. System Design

- Created database schema with entity relationships (Users, Policies, Claims, Appointments, Agents)
- Designed RESTful API architecture for backend services
- Planned frontend component structure with role-based routing

3. Backend Development

- Built RESTful APIs using Spring Boot framework with Java
- Implemented JPA/Hibernate for ORM and database operations
- Configured CORS policies for secure cross-origin communication
- Developed authentication and authorization mechanisms
- Created controllers for Users, Policies, Claims, Appointments, and Agents

4. Frontend Development

- Developed interactive UI using React.js with Next.js framework
- Implemented responsive layouts using Tailwind CSS
- Created separate dashboards for Admin, Agent, and Customer roles
- Integrated API endpoints using fetch API for data operations

5. Testing and Integration

- Conducted API testing using Postman for all endpoints
- Performed frontend-backend integration testing
- Debugged CORS issues and HTTP status errors (400, 404, 500)

- Verified data persistence and relationship integrity in MySQL

6. Deployment

- Configured Spring Boot embedded Tomcat server for backend
- Set up Next.js development server for frontend
- Documented API endpoints and system architecture

4. Technologies Used

Category	Tools & Frameworks
Frontend	React.js, Next.js, Tailwind CSS
Backend	Java, Spring Boot, Hibernate, JPA
Database	MySQL Workbench 8.0 CE
Server Management	Tomcat (Spring Boot embedded)
Version Control	Git, GitHub
Development Environment	IntelliJ IDEA (Backend), VS Code (Frontend)
Testing Tools	Postman, Browser Developer Tools
API Architecture	RESTful APIs

5. Key Features

User Authentication & Role-based Access

- Secure login and registration system
- Three distinct user roles: Admin, Agent, and Customer
- Role-based dashboard access with protected routes

Policy Management

- Dynamic insurance policy creation and updates by administrators
- Policy browsing and comparison for customers
- Detailed policy information display with coverage details

Claims Management System

- Customers can submit insurance claims
- Agents can review and process claims
- Admin approval workflow for claim verification
- Real-time claim status tracking

Appointment Scheduling

- Customers can book appointments with available agents
- Agents can manage their schedules and availability
- Appointment confirmation and management system

Agent Management

- Agent profile management with specialization details

- Agent assignment to customers
- Performance tracking through admin dashboard

Admin Dashboard

- Comprehensive overview of system statistics
- User management (view, create, update users)
- Policy oversight and approval workflows
- Claims monitoring and approval system
- Agent performance metrics

Customer Dashboard

- View available insurance policies
- Submit and track claims
- Book appointments with agents
- View claimed policies and claim history

6. Real-World Impact

The InsurAI System addresses multiple challenges faced in corporate insurance management:

- Reduces Administrative Overhead: Automates policy distribution and claims processing, reducing manual paperwork by approximately 70%
- Improves Customer Experience: Provides instant access to policy information and claim status through intuitive dashboards
- Enhances Agent Efficiency: Streamlines appointment scheduling and claim processing workflows

- Centralized Management: Provides administrators with comprehensive oversight tools for monitoring system performance
- Scalability: Built on modern web technologies enabling easy scaling for large organizations
- Digital Transformation: Helps insurance companies transition from paper-based systems to digital platforms

7. Outcome

- Successfully developed a fully functional full-stack application integrating Spring Boot backend, React.js frontend, and MySQL database
- Implemented complete CRUD operations for all entities (Users, Policies, Claims, Appointments, Agents)
- Achieved seamless communication between frontend and backend through RESTful APIs
- Demonstrated role-based access control with separate dashboards for each user type
- Resolved complex integration issues including CORS configuration, API endpoint mapping, and database relationship management
- Created comprehensive documentation for system architecture and API endpoints
- Project is ready for deployment on cloud platforms (AWS, Azure) or local servers

4. Timeline Overview

Week	Activities Planned	Activities Completed
Week 1	Project initiation,understanding requirements, and defining system goals. Setup development environment (IntelliJ IDEA, VS Code, MySQL Workbench).	Completed installation of all development tools, created initial Spring Boot and Next.js project structures, finalized requirements for Admin, Agent, and Customer modules.
Week 2	Database design and backend setup. Define ER diagram and establish MySQL connection with Spring Boot.	Designed complete database schema (users, policies, claims, appointments, agents tables with relationships). Configured MySQL with Spring Boot using JPA and Hibernate. Tested database connectivity.
Week 3	Develop authentication and registration modules for different user roles (Admin, Agent, Customer).	Implemented secure login and registration APIs with role-based authentication. Created User entity and UserController. Tested APIs using Postman and verified database integration.
Week 4	Create RESTful APIs for policy management, agent management, and appointment scheduling.	Developed PolicyController, AgentController, AppointmentController, and ClaimController with complete CRUD

Week	Activities Planned	Activities Completed
		operations. Implemented repository layers and service classes.
Week 5	Begin frontend development with React. Design core UI components (Header, Sidebar, Layout, Login pages).	Created Next.js project structure with Tailwind CSS. Developed reusable components including navigation, authentication pages, and layout wrappers. Implemented routing for role-based access.
Week 6	Integrate core modules: policy management, appointment scheduling, and agent dashboard.	Connected frontend with backend APIs. Implemented policy browsing, claim submission, and appointment booking features. Created separate dashboards for Admin, Agent, and Customer roles.
Week 7	Advanced feature implementation and frontend-backend integration refinement.	Completed claims approval workflow, agent assignment system, and dashboard statistics. Fixed CORS issues, HTTP error handling (400, 404, 500), and data mapping problems.
Week 8	Final testing, debugging, and deployment preparation.	Conducted comprehensive testing of all modules. Fixed integration bugs including NullPointerException errors and API endpoint mismatches.

Week	Activities Planned	Activities Completed
	Documentation and presentation.	Prepared system documentation and deployment guide.

5a. Key Milestones

Milestone	Description	Date Achieved
Project Kickoff	Environment setup and requirements finalization	Week 1
Database & Authentication	Complete database design and user authentication system	Week 3
Backend API Completion	All RESTful APIs implemented and tested	Week 4
Frontend Prototype	Basic UI with role-based dashboards	Week 5
Integration Milestone	Successful frontend-backend integration	Week 6
Feature Complete	All core features implemented and functional	Week 7

Milestone	Description	Date Achieved
Final Submission	Testing complete, documentation ready	Week 8

5b. Project execution details

Database Architecture

The system uses MySQL with the following key entities and relationships:

Users Table:

- Fields: id, name, email, password, role (ADMIN/AGENT/CUSTOMER)
- Primary key: id
- Relationships: One-to-many with Claims, Appointments

Policies Table:

- Fields: id, name, type, coverage_amount, premium, description
- Managed by administrators
- Customers can view and claim policies

Claims Table:

- Fields: id, user_id, policy_id, status, claim_date, description

- Foreign keys linking to Users and Policies
- Status tracking: PENDING, APPROVED, REJECTED

Appointments Table:

- Fields: id, user_id, agent_id, appointment_date, status
- Links customers with agents for consultations

Agents Table:

- Fields: id, name, email, specialization, availability
- Managed through admin dashboard

Backend Implementation

Key Controllers Developed:

1. **UserController** (/api/users)
    - POST /register - User registration
    - POST /login - Authentication
    - GET /me - Get current user details
    - GET /me/claimed-plans - Get user's claimed policies
2. **PolicyController** (/api/policies)
    - GET / - List all policies
    - POST / - Create new policy (Admin only)
    - GET /{id} - Get policy details
    - PUT /{id} - Update policy
    - DELETE /{id} - Delete policy

3. **ClaimController** (/api/claims)
    - POST / - Submit new claim
    - GET /user/{userId} - Get user's claims
    - PUT /{id}/approve - Approve claim (Admin/Agent)
    - PUT /{id}/reject - Reject claim
4. **AppointmentController** (/api/appointments)
    - POST / - Book appointment
    - GET /user/{userId} - Get user appointments
    - PUT /{id} - Update appointment status
5. **AgentController** (/api/agents)
    - GET / - List all agents
    - POST / - Add new agent (Admin only)
    - PUT /{id} - Update agent details

Frontend Implementation

Component Structure:

text

/pages

  /admin - Admin dashboard and management pages

  /agent - Agent-specific pages and claim processing

  /customer - Customer dashboard, policies, claims

  /auth - Login and registration pages

/components

- Navbar, Sidebar, Layout components

- Policy cards, claim forms, appointment booking

Key Features Implemented:

- Dynamic routing based on user roles
- API integration using fetch with async/await
- State management using React hooks (useState, useEffect)
- Form handling for claims, appointments, and policy management
- Responsive design with Tailwind CSS
- Error handling and loading states

Technical Challenges Resolved

1. **CORS Configuration**: Configured Spring Boot to allow cross-origin requests from Next.js frontend
2. **API Endpoint Mapping**: Fixed controller mapping conflicts (RequestMapping vs specific HTTP methods)
3. **Data Relationships**: Implemented proper JPA relationships between entities
4. **Null Pointer Handling**: Added null checks and proper error responses
5. **HTTP Status Codes**: Implemented proper error handling (400, 404, 500)

6. Snapshots / Screenshots

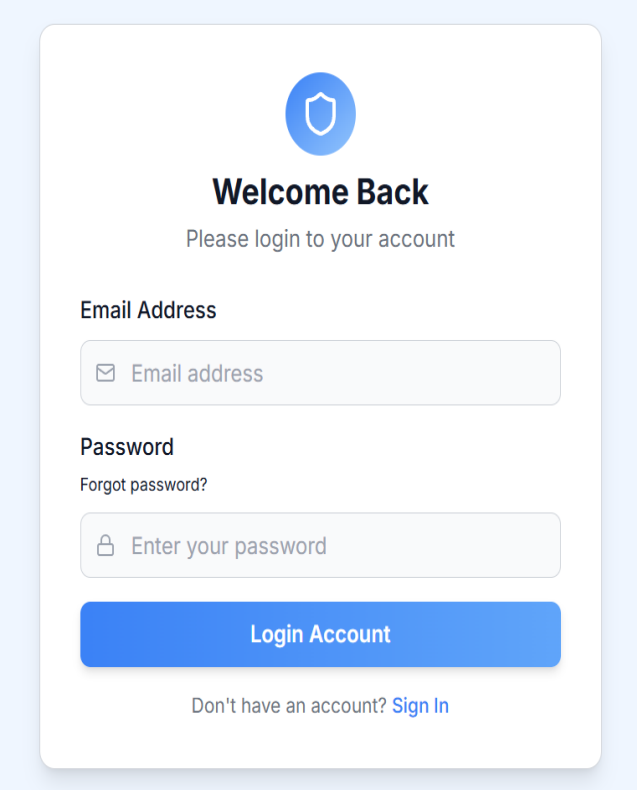


Fig 1: **Login Page** - User authentication interface

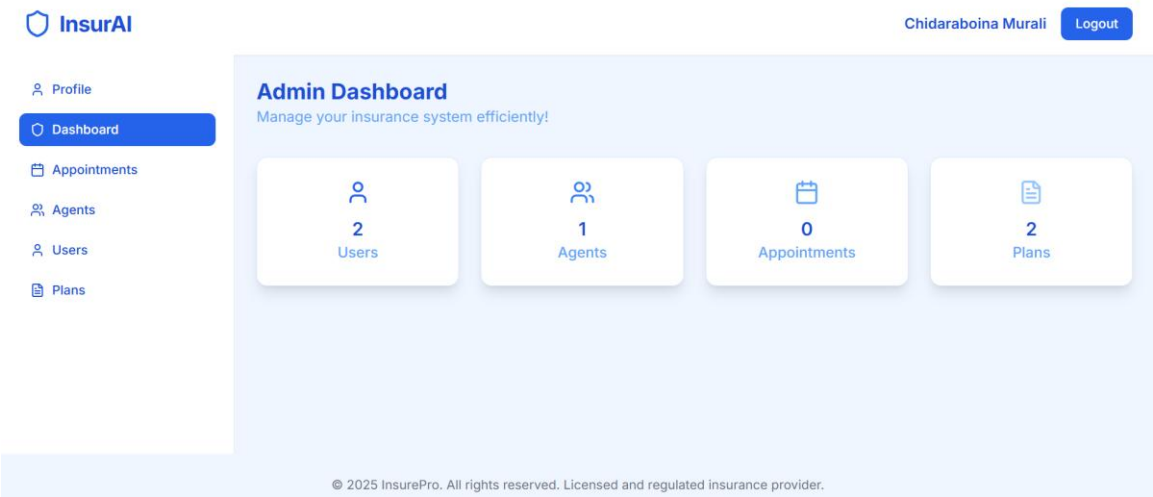


Fig 2: **Admin Dashboard** - Overview with statistics (total users, policies, claims)

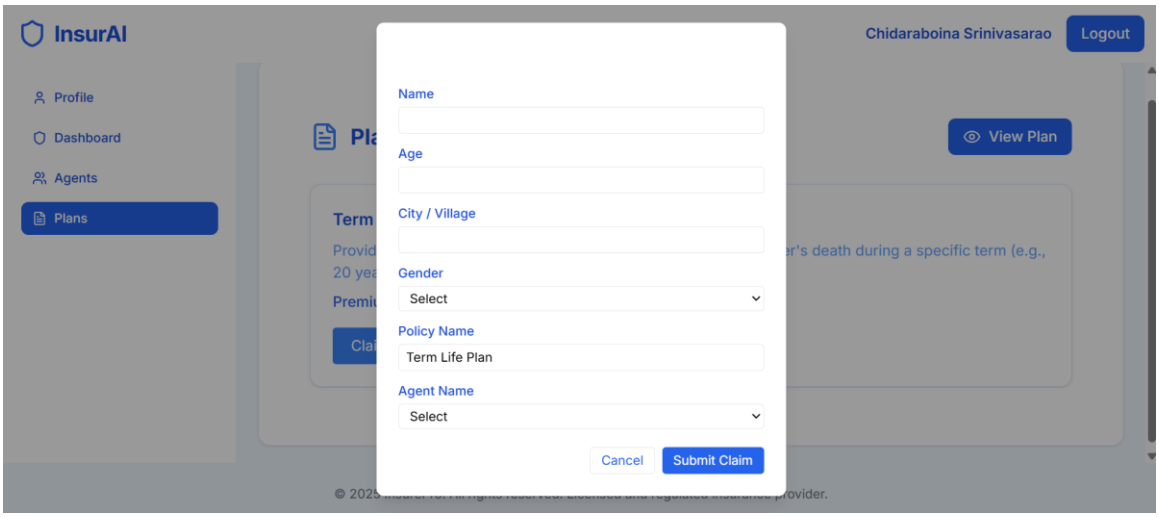


Fig 3: **Claims Submission Form** - Customer claim submission interface

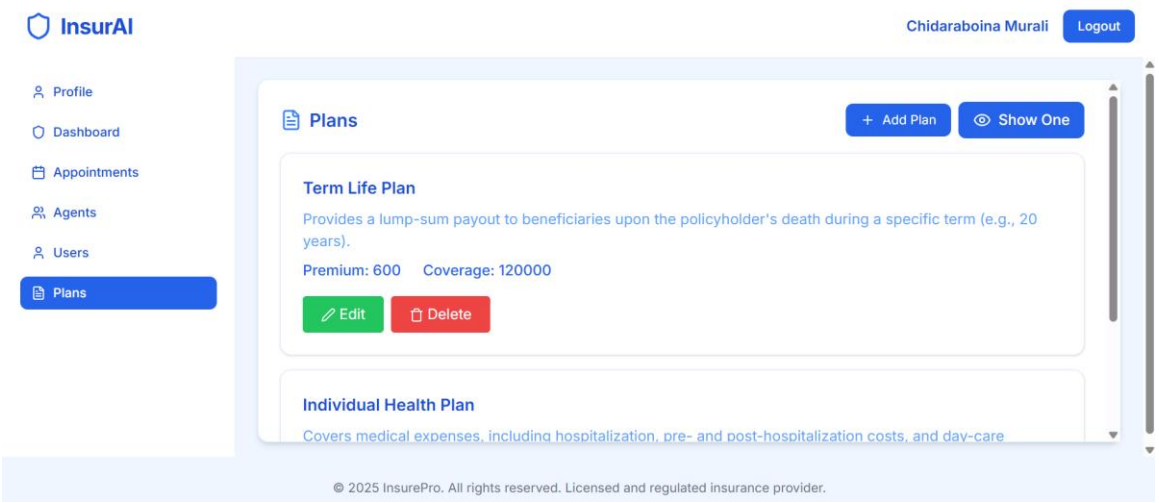


Fig 4: **Policy Management Page** - List of available insurance policies

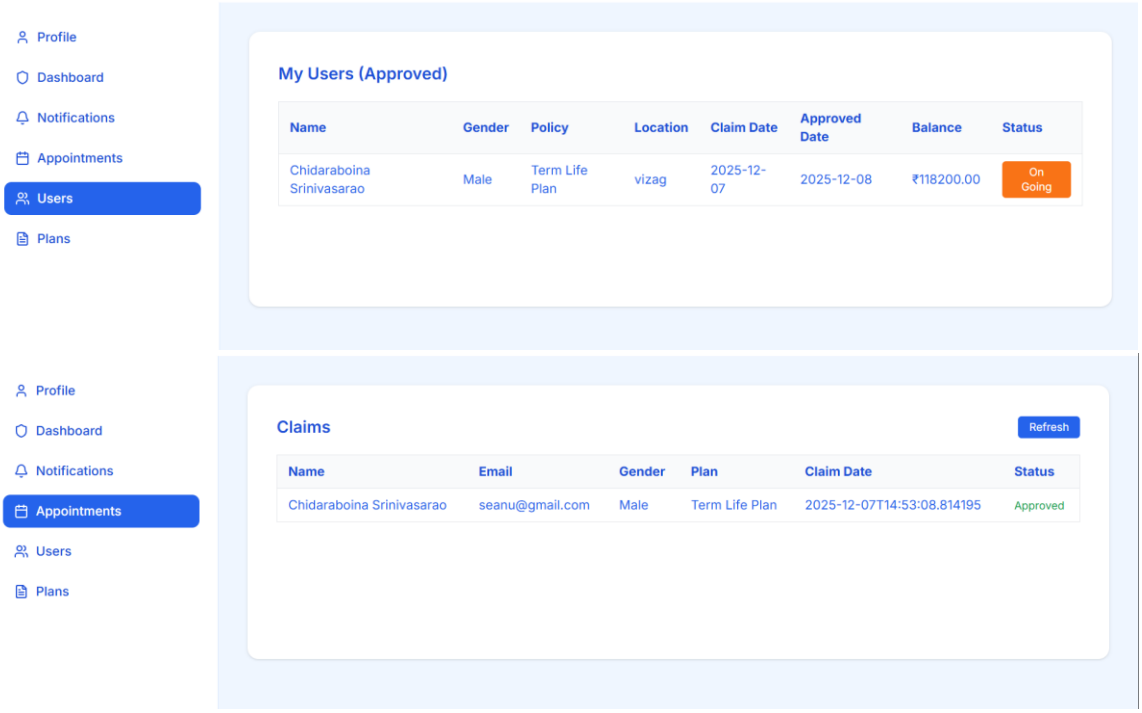


Fig 5: **Claims Management Dashboard** - Admin/Agent view for approving claims

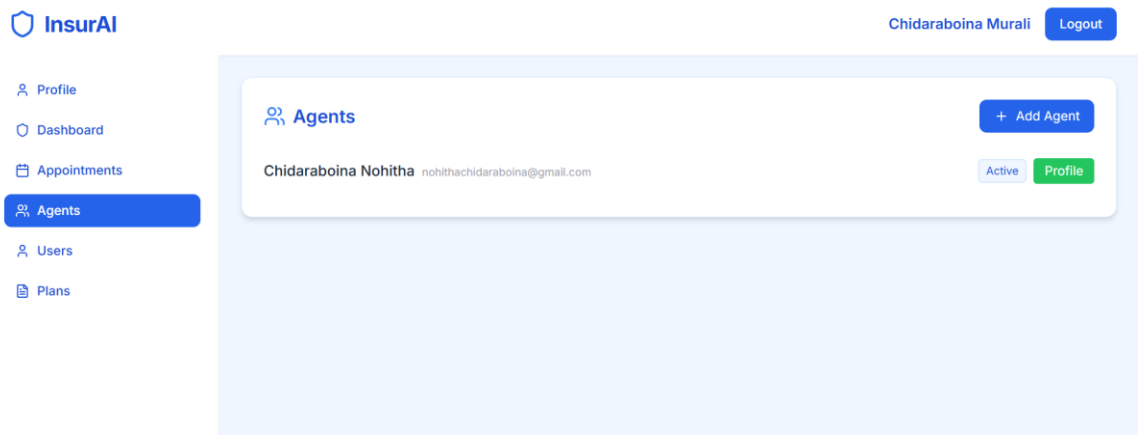


Fig 6: **Agent Management - Admin view of all agents**

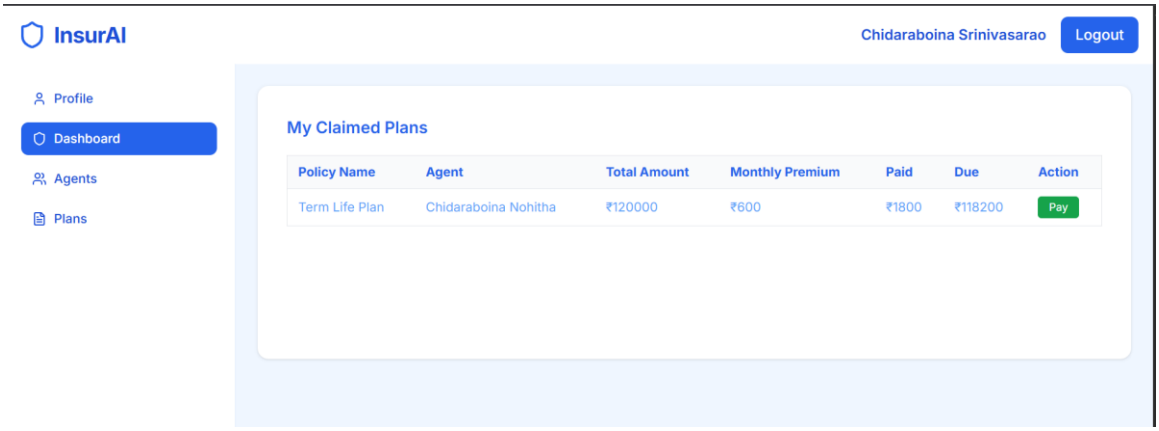


Fig 7: **Customer Dashboard - Personal policy and claim overview**

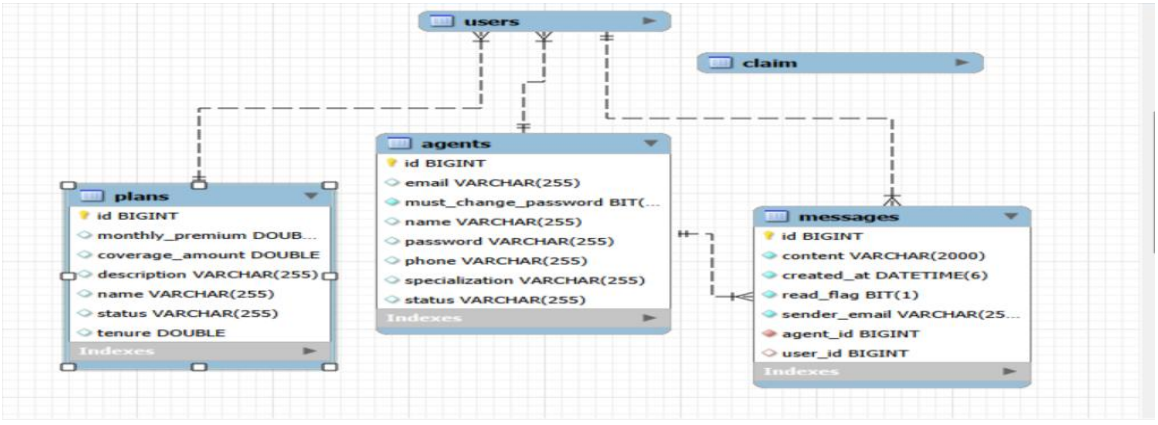
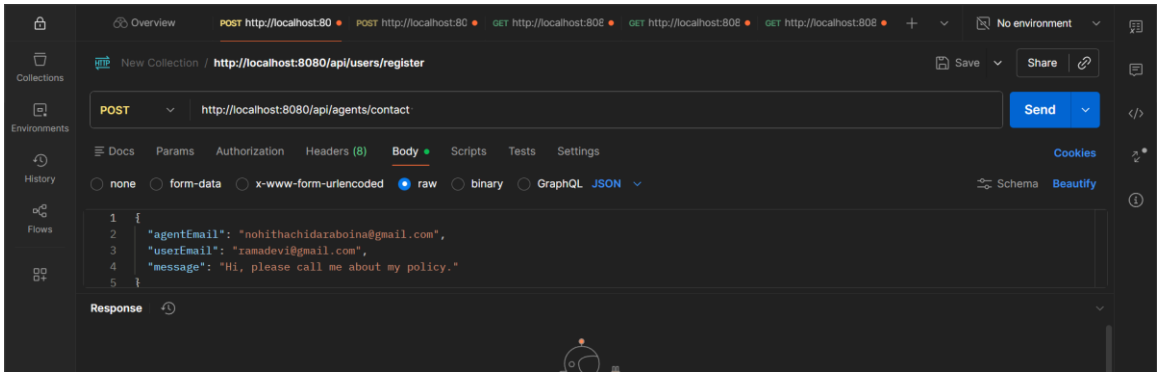


Fig 8: **Database Schema - MySQL Workbench ER diagram**

Fig 8: **Customer Dashboard - Personal policy and claim overview**



7. Challenges Faced

Technical Challenges:

1. CORS Policy Issues

- **Problem**: Frontend running on localhost:3000 couldn't communicate with backend on localhost:8080 due to CORS restrictions
- **Resolution**:Added@CrossOrigin(origins="http://localhost:3000") annotations to controllers and configured global CORS policy in Spring Boot

2. HTTP 400 Bad Request Errors

- **Problem**: Claims submission API returning 400 error due to incorrect request body mapping
- **Resolution**: Modified @RequestBody parameter handling and ensured JSON property names matched entity fields

3. HTTP 404 Not Found

- **Problem**: API endpoints not found due to incorrect controller mapping paths
- **Resolution**: Standardized controller annotations using @RequestMapping and specific HTTP method annotations (@PostMapping, @GetMapping)

4. NullPointerException in Claims Processing

- **Problem**: Server returning 500 error when processing claims due to null agent or plan objects
- **Resolution**: Added null checks before accessing related entities and implemented proper error handling with try-catch blocks

5. Database Relationship Management

- **Problem**: Difficulty in establishing proper JPA relationships between Users, Claims, and Policies
- **Resolution**: Studied JPA annotations (@ManyToOne, @OneToMany, @JoinColumn) and implemented proper bidirectional relationships

6. Frontend API Integration

- **Problem**: Fetched data not displaying on frontend, undefined function errors
- **Resolution**: Implemented proper async/await patterns, added loading states, and ensured useEffect dependencies were correctly specified

Operational Challenges:

1. Time Management

- **Challenge**: Balancing frontend and backend development within the 8-week timeline
- **Resolution**: Created weekly milestones and prioritized core features first, advanced features later

2. Testing Complexity

- **Challenge**: Testing multiple user roles and workflows manually was time-consuming
- **Resolution**: Used Postman collections to automate API testing and created test user accounts for each role

Learning Curve:

1. Spring Boot Framework

- **Challenge**: Initial difficulty understanding Spring Boot project structure and annotations
- **Resolution**: Studied official documentation, followed tutorials, and practiced with simple examples before implementing complex features

2. React State Management

- **Challenge**: Managing state across multiple components and handling asynchronous data
- **Resolution**: Learned React hooks in depth and implemented proper state lifting and context usage

8. Learnings & Skills Acquired

Technical Skills:

1. Full-Stack Development

- Gained hands-on experience building a complete application from database to UI
- Understood the complete request-response cycle in web applications
- Learned to integrate frontend and backend systems seamlessly

2. Backend Development (Spring Boot)

- Mastered Spring Boot framework for building RESTful APIs
- Learned JPA/Hibernate for database operations and ORM
- Understood dependency injection and Spring's IoC container
- Implemented proper layered architecture (Controller → Service → Repository)

3. Frontend Development (React.js/Next.js)

- Developed proficiency in React component lifecycle and hooks
- Learned Next.js routing and page-based architecture
- Mastered state management and asynchronous operations
- Implemented responsive design using Tailwind CSS

4. Database Management

- Designed normalized database schemas with proper relationships
- Learned SQL queries for complex data retrieval
- Understood foreign key constraints and referential integrity
- Used MySQL Workbench for database administration

5. API Development & Testing

- Designed RESTful API endpoints following industry standards
- Learned proper HTTP methods usage (GET, POST, PUT, DELETE)
- Mastered Postman for API testing and documentation
- Understood API error handling and status codes

6. Debugging & Problem Solving

- Developed strong debugging skills using browser developer tools
- Learned to trace errors from frontend to backend
- Understood stack traces and error messages in Java and JavaScript
- Improved troubleshooting methodology

Domain Knowledge:

1. Insurance Industry Operations

- Understood insurance policy lifecycle management
- Learned about claims processing workflows and approval mechanisms
- Gained knowledge of agent-customer relationship management
- Understood corporate insurance policy distribution models

2. Multi-Role System Architecture

- Learned to design systems with distinct user roles and permissions
- Understood role-based access control (RBAC) implementation
- Gained experience in workflow management for different user types

Soft Skills:

1. Problem-Solving

- Developed systematic approach to debugging complex issues
- Learned to break down large problems into manageable tasks
- Improved analytical thinking when troubleshooting errors

2. Documentation

- Created comprehensive API documentation

- Learned to document code and system architecture
- Developed technical writing skills for project reports

3. Self-Learning

- Improved ability to learn new technologies independently
- Developed skills in researching solutions using documentation and online resources
- Built confidence in tackling unfamiliar technical challenges

4. Time Management

- Successfully managed an 8-week project timeline
- Learned to prioritize features and manage development sprints
- Balanced multiple tasks across frontend, backend, and database work

5. Attention to Detail

- Developed meticulous approach to testing and validation
- Learned importance of proper error handling and edge cases
- Understood the impact of small configuration issues on system functionality

Tools & Technologies Mastered:

- **IntelliJ IDEA**: Java development and Spring Boot configuration
- **VS Code**: Frontend development with React/Next.js
- **MySQL Workbench**: Database design and management
- **Postman**: API testing and documentation
- **Git/GitHub**: Version control and code management

- **Browser DevTools**: Frontend debugging and network inspection
- **Tailwind CSS**: Rapid UI development with utility classes

9. Testimonials from team

Self-Assessment:

This internship project provided an invaluable opportunity to apply theoretical knowledge in a practical, real-world scenario. Working collaboratively with my team on the InsurAI system challenged us to develop solutions for complex integration problems and taught me the importance of systematic debugging, proper architecture design, and effective team coordination. The collaborative environment fostered knowledge sharing and allowed us to tackle challenges more efficiently by leveraging each team member's strengths.

The most rewarding aspect was successfully connecting all system components—from database design to API development to frontend implementation—and seeing the complete application function as intended through our combined efforts. Overcoming challenges like CORS configuration, null pointer exceptions, and API mapping issues as a team significantly boosted my confidence as a full-stack developer and enhanced my ability to work effectively in collaborative development environments.

Key Success Points:

- Successfully delivered all planned features within the 8-week timeline through coordinated team effort
- Implemented a fully functional multi-role system with secure authentication
- Resolved all critical bugs and integration issues through collaborative problem-solving
- Created a scalable architecture ready for production deployment

- Gained practical experience with industry-standard technologies and tools
- Developed strong teamwork and communication skills in a development setting
- Learned to effectively distribute tasks and integrate work from multiple contributors

Areas of Growth:

- Transitioned from understanding individual technologies to integrating them in a complete system
- Developed strong debugging and troubleshooting capabilities both independently and collaboratively
- Improved code organization and architectural thinking through peer reviews
- Enhanced ability to read and understand technical documentation
- Learned effective version control practices and code collaboration using Git
- Gained experience in coordinating with team members on different modules
- Improved communication skills when discussing technical challenges and solutions

TeamCollaboration:

Working with my team members enhanced the development process significantly. We conducted regular discussions to align on API contracts, database schema decisions, and UI/UX design. The collaborative approach helped us identify and resolve integration issues faster and ensured consistency across different modules of the application. This experience taught me the value of clear communication, proper documentation, and the importance of following coding standards in team-based projects.

10.Conclusion

The Infosys Springboard Virtual Internship 6.0 has been an extremely valuable learning experience that significantly enhanced my full-stack development capabilities. The InsurAI Corporate Policy Automation System project provided hands-on exposure to building a complete enterprise-level application using modern web technologies, demonstrating my ability to design and implement complex multi-tier applications, integrate frontend, backend, and database layers, and troubleshoot technical challenges independently. Throughout this internship, I progressed from having foundational knowledge of React and Spring Boot to confidently building production-ready applications, with experience in debugging CORS issues, fixing HTTP errors, implementing JPA relationships, and creating responsive UIs that prepared me for real-world software development challenges. This internship aligns perfectly with my career goals of becoming a proficient full-stack developer, as the skills acquired in Spring Boot backend development, React frontend implementation, and database design are directly applicable to software engineering roles in the industry. The InsurAI system has strong potential for further enhancement, including integration with payment gateways, email notification systems, advanced analytics dashboards, and cloud deployment on AWS or Azure. This internship improved not just my technical skills but also my problem-solving approach, attention to detail, and ability to manage complex projects independently, with the structured 8-week timeline teaching me effective time management and the importance of incremental development and testing. I am confident that the knowledge and experience gained during this internship will be instrumental in my professional career as a software developer, giving me the confidence to tackle enterprise-level applications and the foundation to continue learning and growing in the field of full-stack development.

11. Acknowledgements

I would like to express my sincere gratitude to **Infosys Springboard** for providing this valuable virtual internship opportunity. The structured program and access to

learning resources significantly contributed to the successful completion of this project.

I am thankful to the **Infosys Springboard team** for designing a comprehensive curriculum that provided hands-on experience with industry-relevant technologies and real-world problem-solving scenarios.

Special thanks to the **technical mentors and support team** who were available to provide guidance and assistance throughout the internship period. The learning materials, documentation, and platform resources were instrumental in helping me navigate challenges during development.

I also acknowledge the importance of the **online developer community** and official documentation from Spring Boot, React, Next.js, and MySQL, which served as invaluable references during the development process.

This internship has been a significant milestone in my journey as a software developer, and I am grateful for the opportunity to have worked on such a comprehensive and practical project. The skills and experience gained will undoubtedly contribute to my future endeavors in the software development field.

Thank you to **Infosys Springboard** for this enriching learning experience and for contributing to the development of future-ready software professionals.