InsureOne

Insurance Premium Payment & Claims Portal

CAPSTONE PROJECT DOCUMENTATION

Contributors:
Adeeb Ahmed Khan
Krishi Y Antad
M Ranjitha
Lekhana K

Project Type: Full-Stack Web Application

Technology Stack: React + ASP.NET Core + Oracle DB

December 2025

Table of Contents

- **1.** Introduction & Problem Statement
- **2.** Project Objectives
- **3.** Project Scope
- **4.** Technology Stack
- **5.** System Architecture
- **6.** Database Design & ER Diagram
- **7.** API Documentation
- **8.** User Flow Diagram
- **9.** Sequence Diagram - User Operations
- **10.** Sequence Diagram - Admin Operations
- **11.** Features & Modules
- **12.** Security Implementation
- **13.** Testing Strategy
- **14.** Deployment & Docker Setup
- **15.** Application Screenshots
- **16.** Team Contributions
- **17.** Future Enhancements
- **18.** Conclusion

1. Introduction & Problem Statement

1.1 Project Overview

InsureOne is a modern, full-stack insurance client servicing platform designed to revolutionize how customers interact with their insurance policies and claims. The platform provides a seamless digital experience for managing insurance portfolios, filing claims, and tracking claim statuses in real-time.

1.2 Problem Statement

The traditional insurance industry faces significant challenges in customer service delivery. Policyholders often struggle with:

- **Lack of Transparency:** Customers have limited visibility into their policy details and claim statuses
- **Inefficient Processes:** Paper-based claim submissions lead to delays and errors
- **Poor Accessibility:** Limited access to policy information outside business hours
- **Manual Administration:** Agents spend excessive time on routine tasks instead of customer service
- **Communication Gaps:** Lack of real-time updates on policy renewals and claim processing

1.3 Proposed Solution

InsureOne addresses these challenges by providing a comprehensive digital platform that enables:

- 24/7 access to policy information and documents
- Online claim submission with real-time status tracking
- Automated policy renewal notifications and management
- Role-based dashboards for clients and administrators

- Secure authentication and data protection

1.4 Target Users

User Type	Description	Primary Actions
Clients	Insurance policyholders who need to manage their policies	View policies, file claims, track claim status, renew policies
Administrators	Insurance company staff managing client portfolios	View all users, manage policies, approve/reject claims

2. Project Objectives

2.1 Primary Objectives

1. **Digital Transformation:** Create a fully digital platform for insurance policy and claims management
2. **User Experience:** Design an intuitive, responsive interface accessible on all devices
3. **Process Automation:** Automate routine tasks like policy expiry notifications and claim status updates
4. **Security:** Implement robust authentication and authorization mechanisms
5. **Scalability:** Build a containerized architecture that can scale with business growth

2.2 Technical Objectives

- Implement RESTful API architecture following best practices
- Utilize JWT-based authentication for secure API access
- Develop responsive UI using modern React ecosystem
- Configure Docker containerization for consistent deployment
- Implement comprehensive error handling and logging
- Write unit tests for critical business logic

2.3 Business Objectives

- Reduce claim processing time
- Improve customer satisfaction through self-service capabilities
- Decrease in administrative overhead for routine tasks
- Enable real-time business analytics and reporting

3. Project Scope

3.1 In Scope

User Management

- User registration with email verification
- Secure login with JWT authentication
- Role-based access control (Client/Admin)
- Password security requirements

Policy Management

- View all policies with detailed information
- Add new insurance policies
- Policy renewal functionality

- Policy cancellation with confirmation
- Policy status tracking (Active, Lapsed, Cancelled)
- Expiry notifications (30 days before)

Claims Management

- File new claims against active policies
- Edit pending claims
- Withdraw submitted claims
- Track claim status (Submitted, Under Review, Approved, Rejected, Withdrawn)
- Claims history view

Admin Features

- View all registered users
- Access user-specific policies and claims
- Update claim statuses
- Delete policies and claims

3.2 Out of Scope

- Payment processing and premium collection
- Document upload and verification
- Multi-language support
- Mobile native applications
- Email/SMS notifications
- Reporting and analytics dashboard

4. Technology Stack

4.1 Frontend Technologies

React 18 UI Library
Vite Build Tool
Tailwind CSS Styling
React Router Navigation
Redux Toolkit State Management
Axios HTTP Client
Lucide React Icons
React Toastify Notifications
JWT Decode Token Parsing

4.2 Backend Technologies

ASP.NET Core 8 Web API Framework
Entity Framework Core ORM
ASP.NET Identity Authentication

JWT Bearer Token Auth
Swagger/OpenAPI API Documentation
xUnit + Moq Unit Testing

4.3 Database & Infrastructure

Oracle XE 21 Database
Docker Containerization
Docker Compose Orchestration
Nginx Web Server

4.4 Development Tools

Tool	Purpose
Visual Studio / VS Code	IDE for development
Git	Version control

Bruno	API testing
SQL Developer	Database management
Podman Desktop	Container management

5. System Architecture

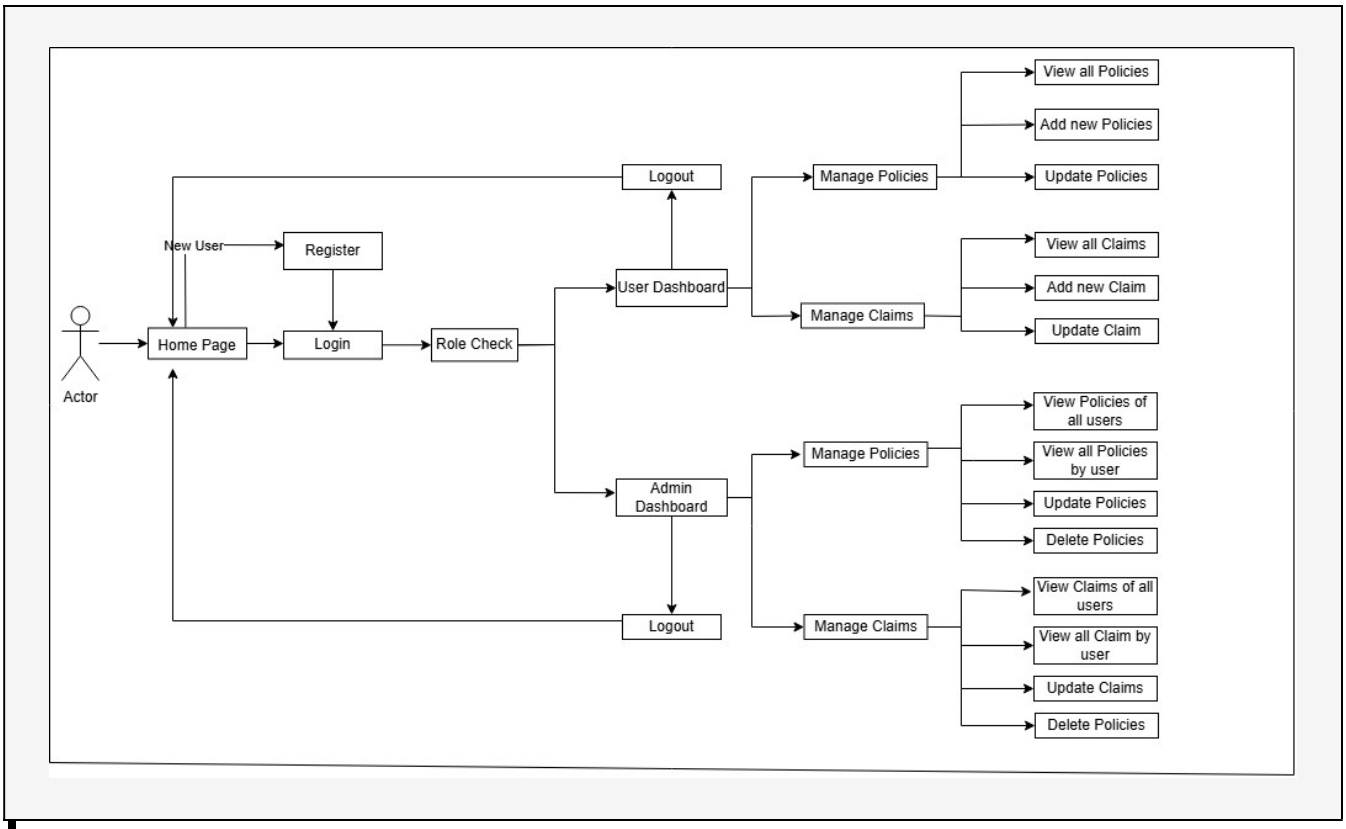
5.1 High-Level Architecture

InsureOne follows a modern three-tier architecture pattern with clear separation of concerns:

Architecture Layers

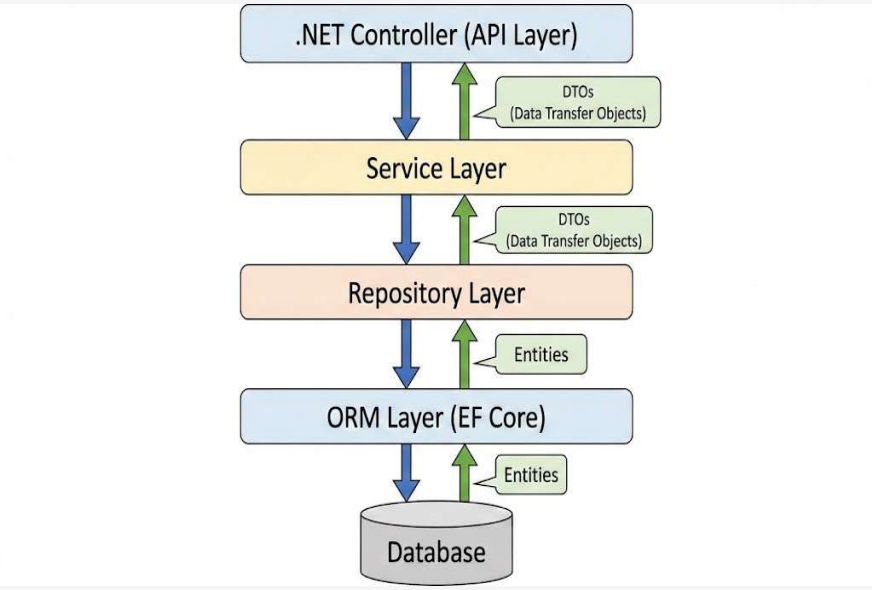
- **Presentation Layer (Frontend):** React SPA served via Nginx
- **Business Logic Layer (Backend):** ASP.NET Core Web API
- **Data Access Layer:** Entity Framework Core with Oracle Database

5.2 User Flow Diagram



5.3 Backend Architecture Pattern

The backend follows the **Repository Pattern** with Service Layer for clean separation:



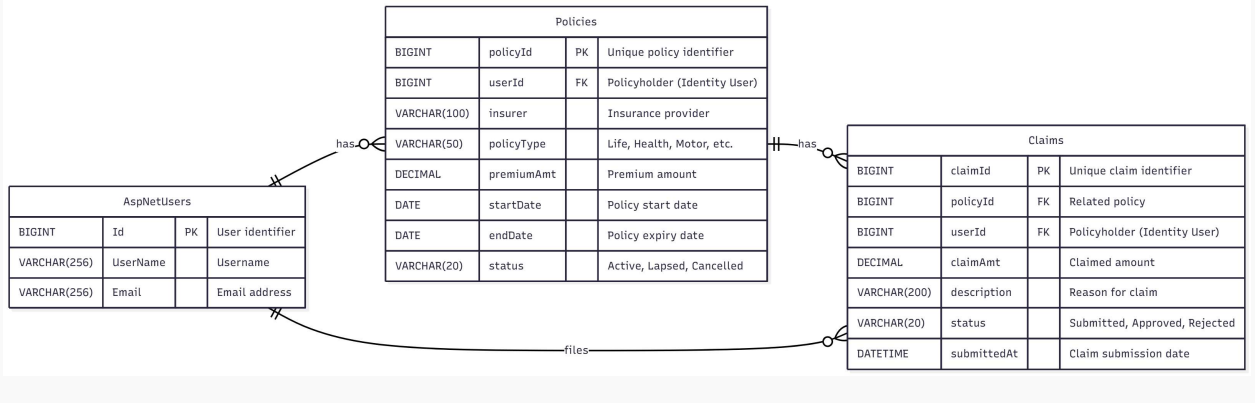
6. Database Design & ER Diagram

6.1 Entity Relationship Overview

The database schema consists of three main entities with the following relationships:

Relationship	Description	Cardinality
User → Policies	A user can have multiple insurance policies	One-to-Many (1:N)
Policy → Claims	A policy can have multiple claims filed against it	One-to-Many (1:N)
User → Claims	A user can file multiple claims	One-to-Many (1:N)

6.2 ER Diagram



6.3 Table Definitions

6.3.1 ApplicationUser (ASPNETUSERS)

Column	Data Type	Constraints	Description
Id	NVARCHAR2(450)	PRIMARY KEY	Unique user identifier (GUID)
UserName	NVARCHAR2(256)	UNIQUE, NOT NULL	User's login name (email)
Email	NVARCHAR2(256)	UNIQUE, NOT NULL	User's email address

6.3.2 Policy (POLICIES)

Column	Data Type	Constraints	Description
PolicyId	NUMBER	PRIMARY KEY, AUTO INCREMENT	Unique policy identifier
UserId	NVARCHAR2(450)	FOREIGN KEY	Reference to user

Insurer	NVARCHAR2(256)	NOT NULL	Insurance company name
PolicyType	NVARCHAR2(100)	NOT NULL	Type of insurance
PremiumAmt	NUMBER(18,2)	NOT NULL	Premium amount
StartDate	DATE	NOT NULL	Policy start date
EndDate	DATE	NOT NULL	Policy end date
Status	NVARCHAR2(50)	NOT NULL	Active/Lapsed/Cancelled

6.3.3 Claim (CLAIMS)

Column	Data Type	Constraints	Description
ClaimId	NUMBER	PRIMARY KEY, AUTO INCREMENT	Unique claim identifier

PolicyId	NUMBER	FOREIGN KEY	Reference to policy
UserId	NVARCHAR2(450)	FOREIGN KEY	Reference to user
ClaimAmt	NUMBER(18,2)	NOT NULL	Claimed amount
Description	NVARCHAR2(2000)	NOT NULL	Claim description
Status	NVARCHAR2(50)	NOT NULL	Submitted/Under Review/Approved/Rejected/Withdrawn
SubmittedAt	TIMESTAMP	NOT NULL	Submission timestamp

7. API Documentation

7.1 Base URL

```
Development: http://localhost:8080/api
Docker: http://localhost:8080/api
```

7.2 Authentication Endpoints

```
POST /api/auth/register
```

Description: Register a new user account

Request Body:

```
{
    "fullName": "John Doe",
    "email": "john@example.com",
    "password": "SecurePass123!"
}
```

```
POST /api/auth/login
```

Description: Authenticate user and receive JWT token

Request Body:

```
{
    "email": "john@example.com",
    "password": "SecurePass123!"
}
```

Response:

```
{
    "token": "eyJhbGciOiJIUzI1NiIs...",
    "email": "john@example.com",
    "roles": ["Client"]
}
```

7.3 Policy Endpoints

Method	Endpoint	Description	Auth Required

GET	/api/policies	Get all policies for authenticated user	Yes
GET	/api/policies/{id}	Get specific policy by ID	Yes
POST	/api/policies	Create a new policy	Yes
PUT	/api/policies/{id}	Update an existing policy	Yes

7.4 Claims Endpoints

Method	Endpoint	Description	Auth Required
GET	/api/claims	Get all claims for authenticated user	Yes
GET	/api/claims/{id}	Get specific claim by ID	Yes
POST	/api/claims	Submit a new claim	Yes
PUT	/api/claims/{id}	Update claim details	Yes

DELETE	/api/claims/{id}	Delete/Withdraw a claim	Yes

7.5 Admin Endpoints

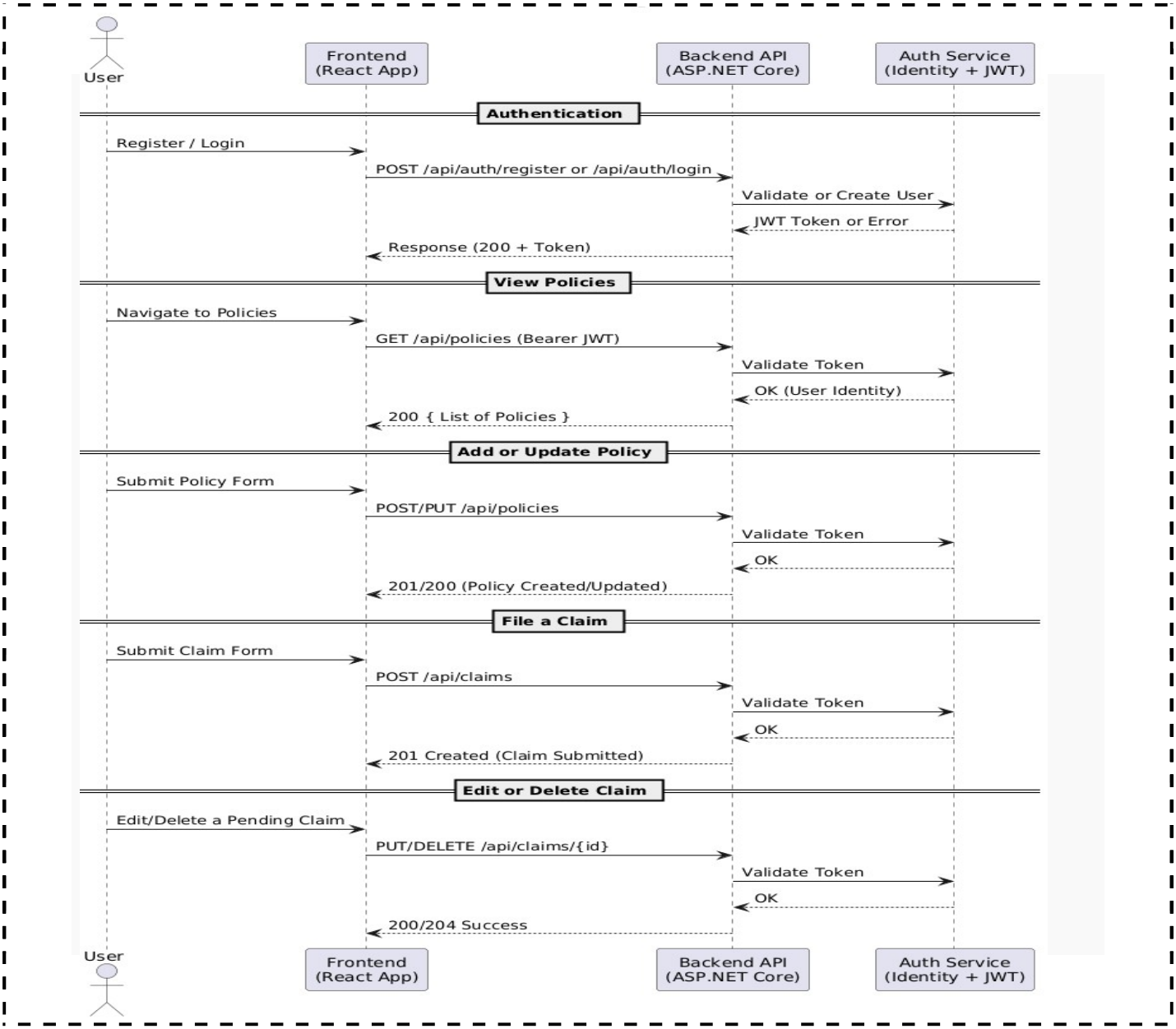
Note: All admin endpoints require Admin role authorization.

Method	Endpoint	Description
GET	/api/admin/users	Get all registered users
GET	/api/admin/users/{userId}/policies	Get all policies for a specific user
GET	/api/admin/users/{userId}/claims	Get all claims for a specific user
PUT	/api/admin/policies/{id}	Update any policy
PUT	/api/admin/claims/{id}	Update claim status
DELETE	/api/admin/policies/{id}	Delete any policy

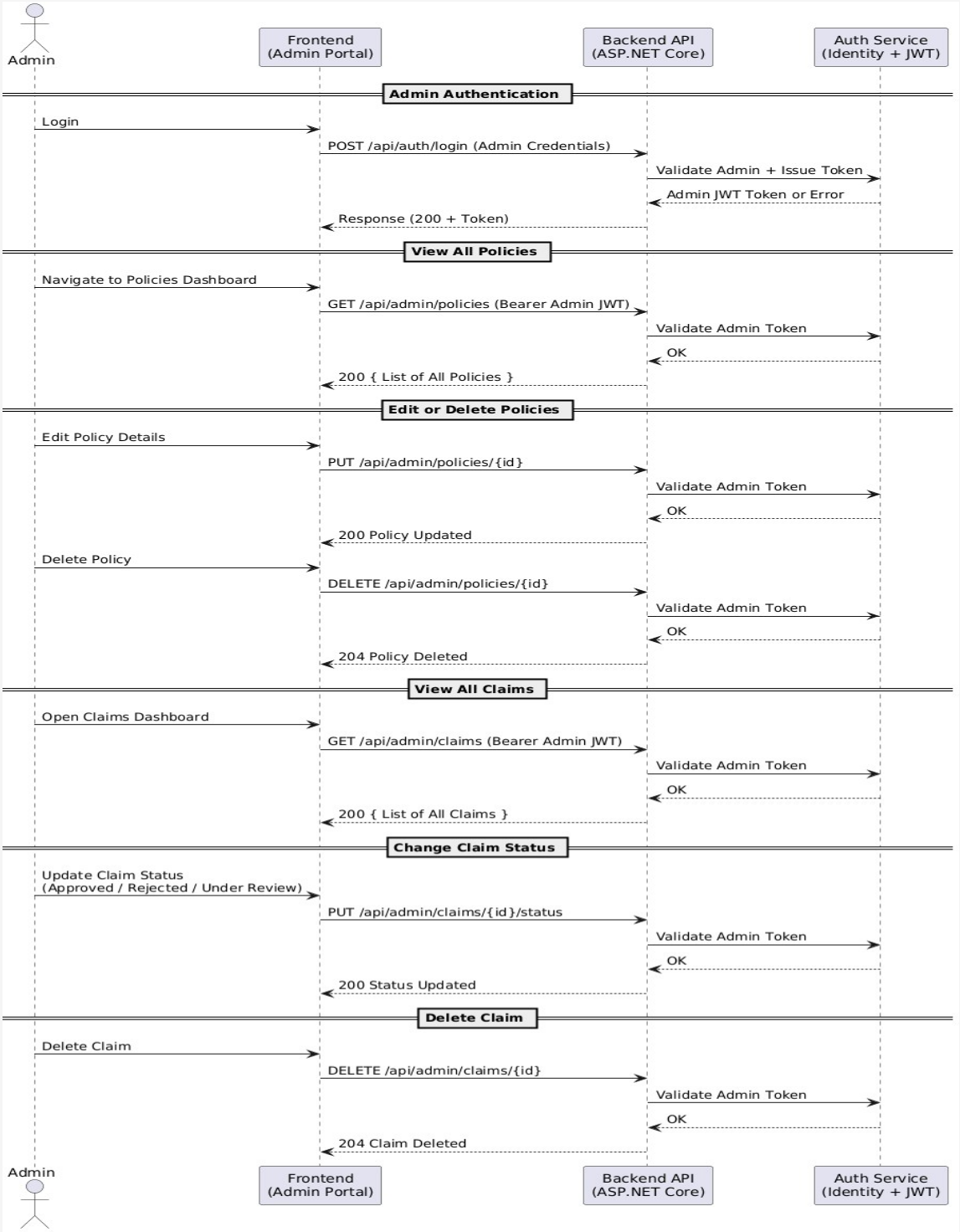
DELETE	/api/admin/claims/{id}	Delete any claim

8. Sequence Diagram

8.1 User



8.2 Admin


Admin
Frontend (Admin Portal)
Backend API (ASP.NET Core)
Auth Service (Identity + JWT)
Admin Authentication
Login
POST /api/auth/login (Admin Credentials)
Validate Admin + Issue Token
Admin JWT Token or Error
Response (200 + Token)
View All Policies
Navigate to Policies Dashboard
GET /api/admin/policies (Bearer Admin JWT)
Validate Admin Token
OK
200 { List of All Policies }
Edit or Delete Policies
Edit Policy Details
PUT /api/admin/policies/{id}
Validate Admin Token
OK
200 Policy Updated
Delete Policy
DELETE /api/admin/policies/{id}
Validate Admin Token
OK
204 Policy Deleted
View All Claims
Open Claims Dashboard
GET /api/admin/claims (Bearer Admin JWT)
Validate Admin Token
OK
200 { List of All Claims }
Change Claim Status
Update Claim Status
(Approved / Rejected / Under Review)
PUT /api/admin/claims/{id}/status
Validate Admin Token
OK
200 Status Updated
Delete Claim
Delete Claim
DELETE /api/admin/claims/{id}
Validate Admin Token
OK
204 Claim Deleted
Admin
Frontend (Admin Portal)
Backend API (ASP.NET Core)
Auth Service (Identity + JWT)

9. Features & Modules

9.1 Authentication Module

Features:

- **User Registration:** New users can create accounts with email and password
- **Secure Login:** JWT-based authentication with role-based access
- **Password Security:** Minimum 6 characters, requires digits and lowercase
- **Role Assignment:** Automatic Client role on registration, Admin role for system admin
- **Token Management:** Secure token storage with automatic expiry handling

9.2 Dashboard Module

Client Dashboard:

- Overview statistics: Total Policies, Active Policies, Total Claims, Pending Claims
- Quick action buttons for adding policies and filing claims
- Support contact information

Admin Dashboard:

- List of all registered users (excluding system admin)
- Search functionality by username
- Navigation to user-specific policies and claims

9.3 Policy Management Module

Features:

- **Add Policy:** Create new insurance policies with insurer, type, premium, and dates
- **View Policies:** Grid view with status indicators and expiry warnings
- **Renew Policy:** Extend policy end date (available when expiring within 30 days)
- **Cancel Policy:** Cancel active policies with confirmation modal
- **Policy Info:** Detailed policy information view
- **Auto-Lapse Detection:** Automatic status change when policy expires
- **Search & Filter:** Search by insurer/type, filter by status

9.4 Claims Management Module

Features:

- **File Claim:** Submit claims against active policies
- **Edit Claim:** Modify pending claims (not approved/rejected)
- **Withdraw Claim:** Cancel submitted claims with confirmation
- **Claims History:** View all claims including withdrawn ones
- **Status Tracking:** Real-time status updates (Submitted, Under Review, Approved, Rejected)
- **Auto-Withdraw:** Claims automatically withdrawn when policy is cancelled
- **Search & Filter:** Search by policy name, filter by status

9.5 Admin Management Module

Features:

- **User Management:** View all registered users with details
- **Policy Oversight:** Access any user's policies with full CRUD capabilities

- **Claims Processing:** View and update claim statuses for approval/rejection
- **User Search:** Search users by name

10. Security Implementation

10.1 Authentication Security

JWT Token Structure

```
{
  "sub": "user-guid",
  "email": "user@example.com",
  "id": "user-guid",
  "full_name": "User Name",
  "http://schemas.microsoft.com/ws/2008/06/identity/claims/role": "Client",
  "exp": 1234567890,
  "iss": "MyAPI",
  "aud": "MyAPIUser"
}
```

Password Requirements

Requirement	Validation
Minimum Length	6 characters
Require Digit	At least one number

Require Lowercase	At least one lowercase letter

10.2 Authorization

Role-Based Access Control

Role	Access Level	Permissions
Client	Own Resources	View/Manage own policies and claims
Admin	All Resources	View/Manage all users, policies, and claims

Protected Routes (Frontend)

- /dashboard - Requires authentication
- /policies - Requires authentication
- /claims - Requires authentication
- /admin/* - Requires authentication + Admin role

10.3 API Security

- **HTTPS:** All API communications encrypted
- **CORS Policy:** Configured to allow only frontend origin
- **JWT Validation:** Token validation on every request
- **Token Expiry:** 3 days token lifetime

- **Error Handling:** Custom middleware for consistent error responses

11. Testing Strategy

11.1 Unit Testing

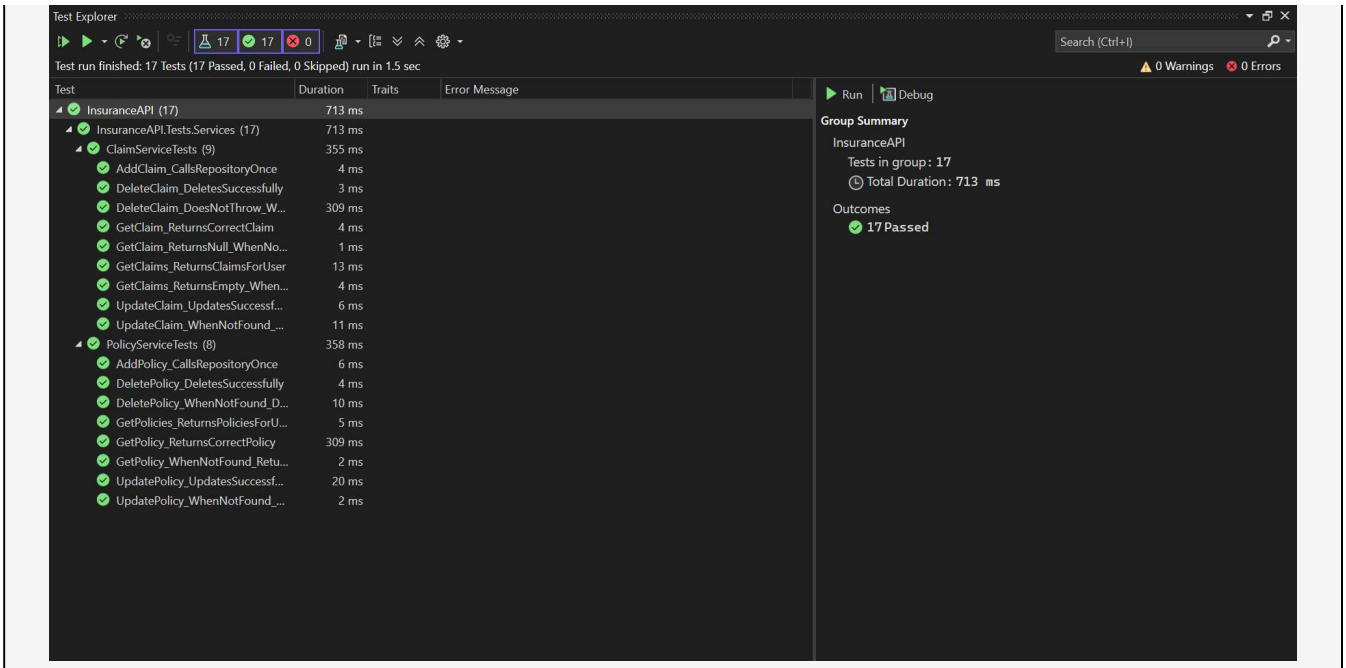
The project uses **xUnit** as the testing framework with **Moq** for mocking dependencies.

Test Coverage

Test Class	Tests	Description
PolicyServiceTests	8	Business logic tests for policies
ClaimServiceTests	9	Business logic tests for claims

11.2 Running Tests

```
cd .NET-backend
dotnet test
```



11.3 API Testing

Swagger UI is available at `http://localhost:8080/swagger` for interactive API testing during development.

12. Deployment & Docker Setup

12.1 Docker Architecture

- **Backend Container:** Insurance-API with environment variables for database connection
- **Frontend Container:** Insurance-frontend serving built React app via Nginx
- **Docker Compose:** Orchestrates multi-container setup for local testing
- **Networking:** Containers communicate through Docker bridge network

12.3 Deployment Commands

Start All Services

```
docker-compose up --build
```

Stop Services

```
docker-compose down
```

View Logs

```
docker-compose logs -f [service-name]
```

12.4 Manual Deployment (Without Docker)

Frontend

```
cd React-frontend
npm install
npm run dev
```

Backend

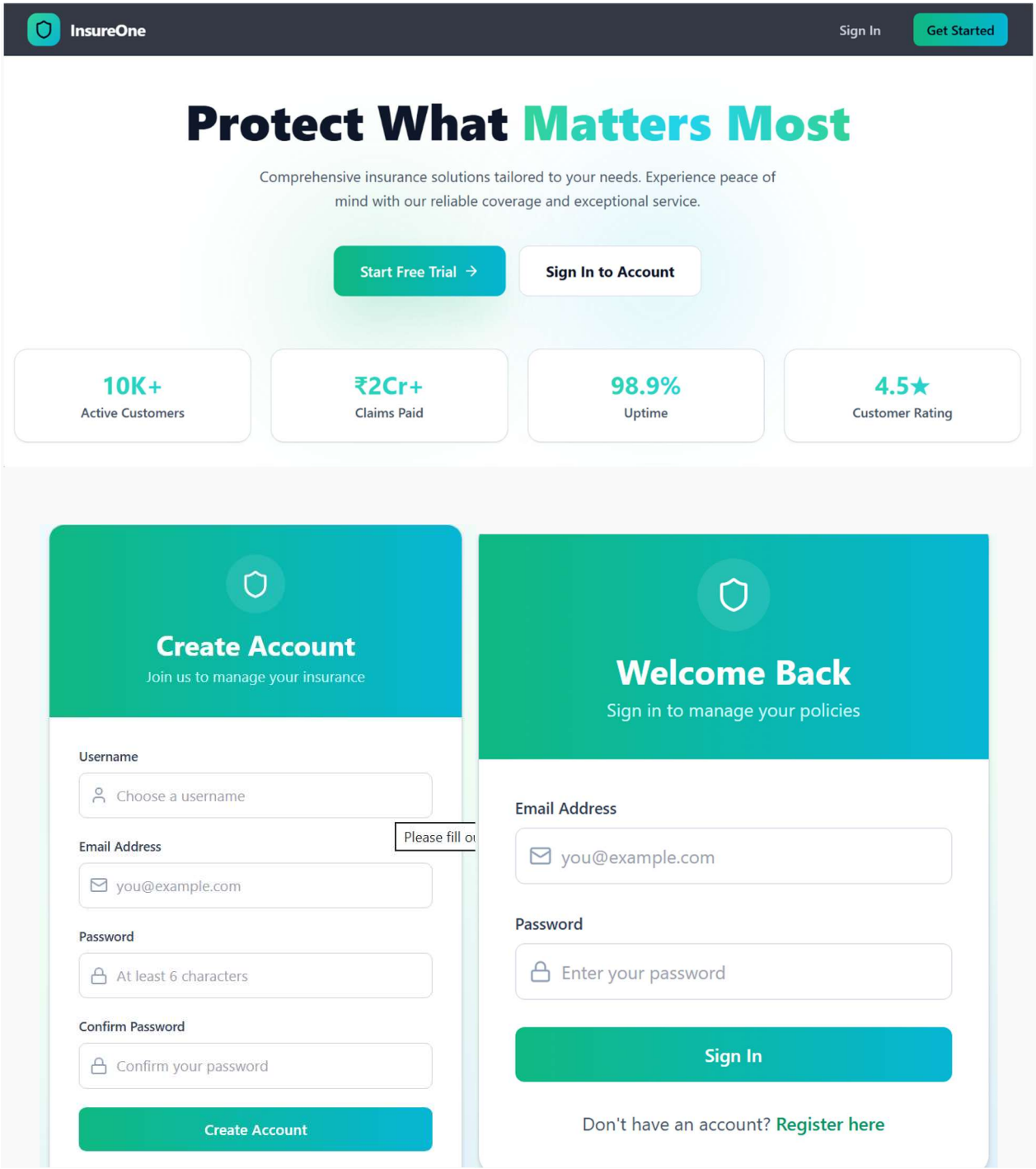
```
cd .NET-backend
dotnet restore
dotnet build
dotnet run
```

12.5 Port Configuration

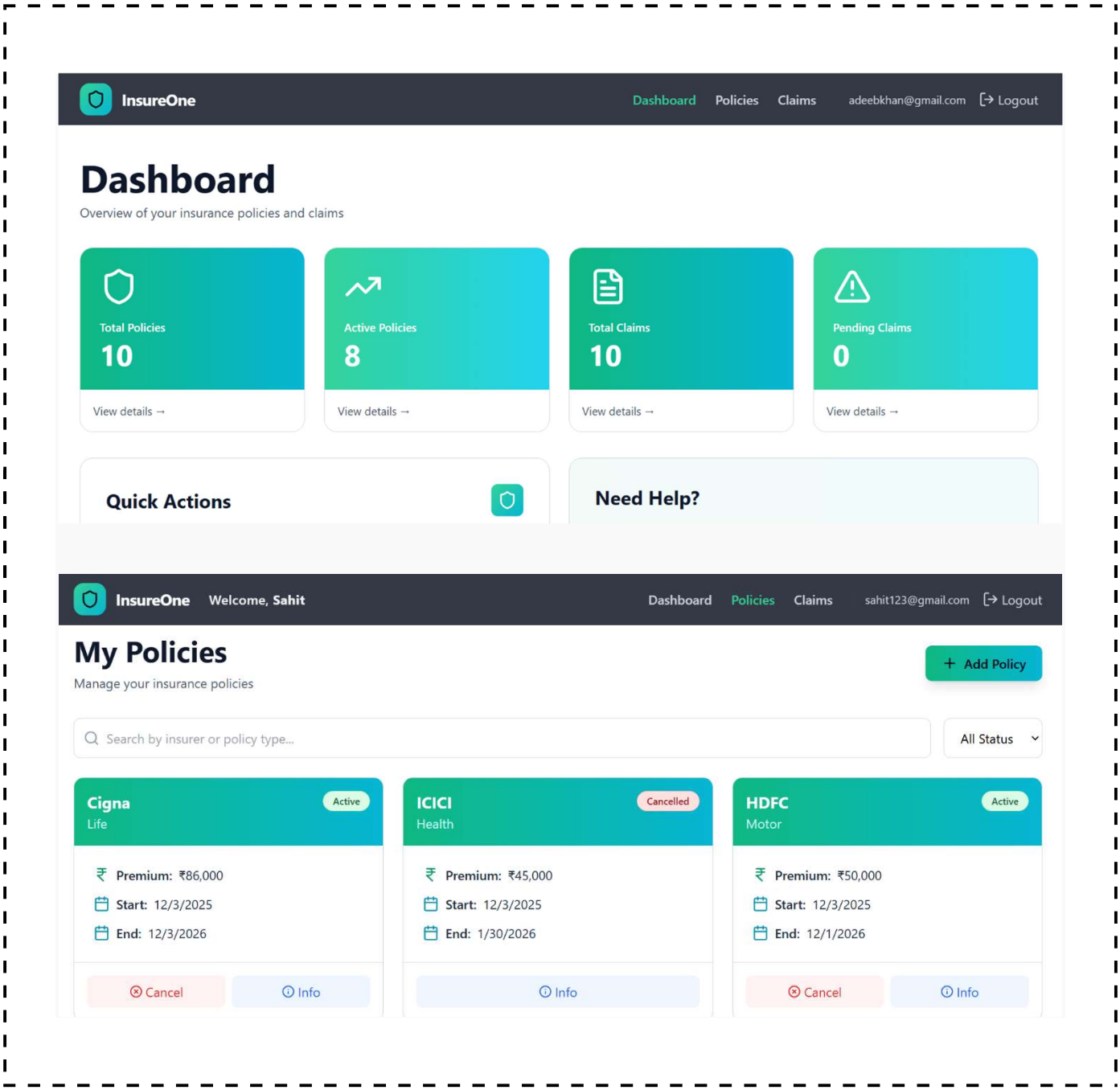
Service	Port	URL
Frontend (Docker)	3000	http://localhost:3000
Frontend (Dev)	5173	http://localhost:5173
Backend API	8080	http://localhost:8080
Oracle Database	1521	localhost:1521
Swagger UI	8080	http://localhost:8080/swagger

13. Application Screenshots

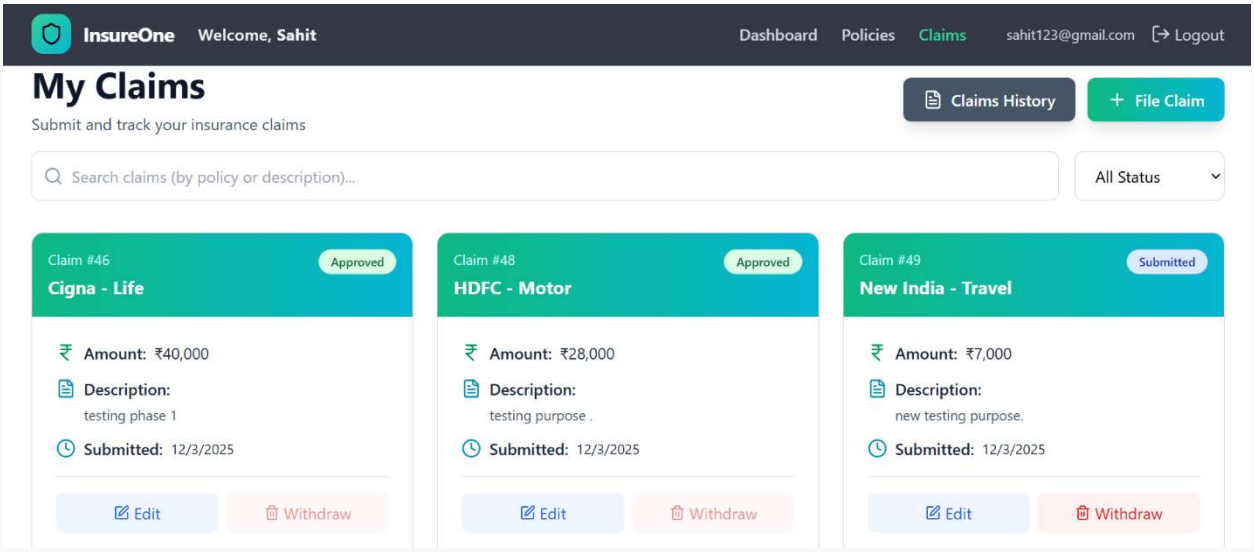
13.1 Landing & Authentication Pages



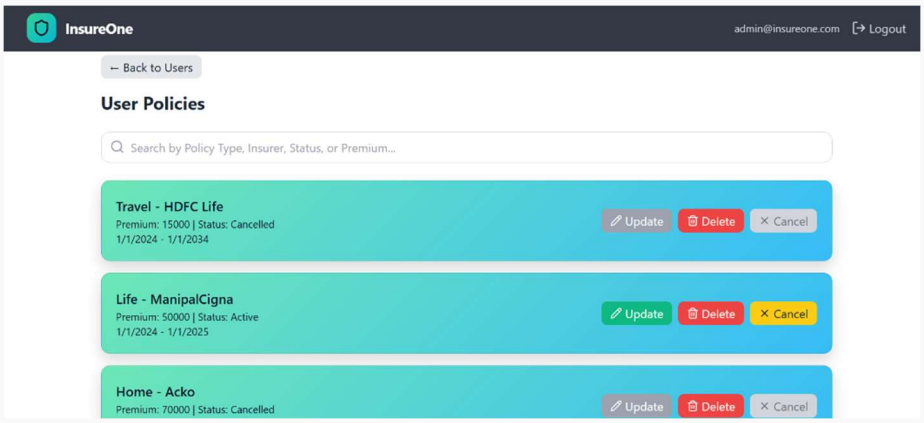
13.2 Client Dashboard & Policies

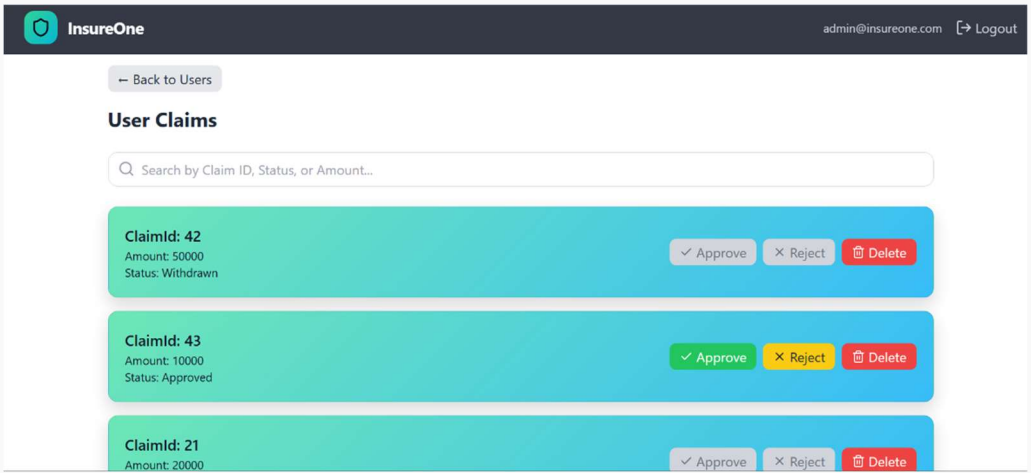


13.3 Claims & Admin Interface


InsureOne
Welcome, Sahit
Dashboard
Policies
Claims
sahit123@gmail.com
Logout
My Claims
Submit and track your insurance claims
Claims History
File Claim
Search claims (by policy or description)...
All Status
Claim #46
Approved
Cigna - Life
Amount: ₹40,000
Description:
testing phase 1
Submitted: 12/3/2025
Edit
Withdraw
Claim #48
Approved
HDFC - Motor
Amount: ₹28,000
Description:
testing purpose .
Submitted: 12/3/2025
Edit
Withdraw
Claim #49
Submitted
New India - Travel
Amount: ₹7,000
Description:
new testing purpose.
Submitted: 12/3/2025
Edit
Withdraw

Admin:


InsureOne
admin@insureone.com
Logout
← Back to Users
User Policies
Search by Policy Type, Insurer, Status, or Premium...
Travel - HDFC Life
Premium: 15000 | Status: Cancelled
1/1/2024 - 1/1/2034
Update
Delete
Cancel
Life - ManipalCigna
Premium: 50000 | Status: Active
1/1/2024 - 1/1/2025
Update
Delete
Cancel
Home - Acko
Premium: 70000 | Status: Cancelled
Update
Delete
Cancel


InsureOne
admin@insureone.com
Logout
← Back to Users
User Claims
Search by Claim ID, Status, or Amount...
ClaimId: 42
Amount: 50000
Status: Withdrawn
Approve
Reject
Delete
ClaimId: 43
Amount: 10000
Status: Approved
Approve
Reject
Delete
ClaimId: 21
Amount: 20000
Approve
Reject
Delete

14. Team Contributions

Member	Features	Backend	Frontend	Extras
Adeeb Ahmed Khan	User => Policies – [GetAll/ Add] Auth => [Register/ Login] Admin => [Delete Policy]	Services/ Repo/ Tests/ Controllers/ JWT authentication	Login & Register page/ Policies page/ Admin policies page	Docker files
Krishi Y Antad	User => Policies – [GetById/ Update/ Delete] Admin => [GetPolicies/ UpdatePolicy]	Services/ Repo/ Tests/ Controllers/ Integration	Landing Page/ Policies page/ Admin Policies page	Documentation
Lekhana K	User Claims – [GetAll/Update] Admin => [GetUsers/ GetClaimsForUser/ UpdateClaim]	Services/ Repo/ Tests/ Controllers/ Integration	Admin Dashboard/ Claims page/ Admin Claims page	Documentation
M Ranjitha	User => Claims – [GetById/ Add] Admin => [DeleteClaim]	Services/ Repo/ Tests/ Controllers/ JWT authentication	User Dashboard/ User Claims page/ Admin Claims page	Presentation

15. Future Enhancements

Feature	Description
Email Notifications	Send email alerts for policy expiry and claim status updates
Document Upload	Allow users to upload supporting documents for claims
Password Reset	Forgot password functionality with email verification
Chat Support	Real-time AI-assisted customer support chat

16. Conclusion

16.1 Project Summary

InsureOne successfully delivers a comprehensive insurance client servicing platform that addresses the key challenges faced by traditional insurance service delivery. The platform provides:

- A modern, responsive user interface for seamless policy and claims management
- Secure authentication and role-based access control
- Real-time claim status tracking and policy management
- Administrative capabilities for efficient backend operations
- Containerized deployment for easy scalability and maintenance

16.2 Technical Achievements

- Successfully integrated React frontend with ASP.NET Core backend
- Implemented JWT-based authentication with role-based authorization
- Designed and implemented a normalized database schema with Oracle
- Achieved containerization with Docker for consistent deployment
- Developed comprehensive API endpoints following RESTful principles
- Implemented unit testing with xUnit and Moq

16.3 Lessons Learned

- **Architecture Planning:** Proper upfront design of the system architecture saved significant refactoring time
- **Testing Early:** Writing tests alongside development helped catch bugs early
- **Docker Benefits:** Containerization simplified deployment and environment consistency

- **Component Reusability:** Designing reusable React components improved development speed

16.4 Acknowledgments

We would like to express our gratitude to our project mentors and instructors for their guidance throughout the development of this project. Their insights and feedback were invaluable in shaping the final product.