Project Title: Health Insurance Claim Management System

1. Overview

The **Health Insurance Claim Management System** is designed to manage the end-to-end flow of health insurance claims, from policy creation to claim approval. This system provides a seamless user experience for policyholders and administrators while ensuring a smooth workflow for insurance agents and claim adjusters. The system follows the **MVC architecture** and is compatible with both **Java (Spring MVC)** and **.NET (ASP.NET Core MVC)** frameworks.

The core modules of the application include:

1. **Policy Management** – Manages the creation and tracking of insurance policies.
2. **Claim Management** – Handles claims submission, approval, and processing.
3. **Customer Support** – Provides assistance to policyholders and claimants.
4. **Document Management** – Manages the storage and retrieval of claim-related documents.
5. **User Authentication & Authorization** – Handles user login, registration, and access controls.

2. Assumptions

1. The system will be deployed locally during development using a relational database (MySQL or SQL Server).
2. Role-based authentication will be used to ensure only authorized users can access sensitive information.
3. The application will support both Java and .NET environments, with ORM tools like Hibernate (Java) and Entity Framework (ASP.NET Core) for data management.
4. Document storage will be implemented via file storage systems or a dedicated document database for the local environment.
5. Claims will be processed by a mock approval mechanism for local development.

3. Module-Level Design

3.1 Policy Management Module

Purpose: Manages the creation, update, and tracking of health insurance policies.

- **Controller**:
  - PolicyController

        - createPolicy(policyData)
        - updatePolicy(policyId, policyData)
        - getPolicyDetails(policyId)
        - getAllPolicies()
        - deletePolicy(policyId)
- **Service**:
    - PolicyService
        - Handles the business logic for creating, updating, and retrieving policies.
- **Model**:
    - **Entity**: Policy
        - Attributes:
            - policyId (PK)
            - policyNumber (VARCHAR)
            - policyholderId (FK)
            - coverageAmount (DECIMAL)
            - policyStatus (ENUM: ACTIVE, INACTIVE, CANCELLED)
            - createdDate (DATE)

3.2 Claim Management Module

Purpose: Handles the submission, processing, and approval of insurance claims.

- **Controller**:
    - ClaimController
        - submitClaim(claimData)
        - getClaimDetails(claimId)
        - updateClaimStatus(claimId, status)
        - getAllClaimsByPolicy(policyId)
- **Service**:
    - ClaimService
        - Manages the processing and approval of claims.
- **Model**:

    - **Entity**: Claim
        - Attributes:
            - claimId (PK)
            - policyId (FK)
            - claimAmount (DECIMAL)
            - claimDate (DATE)
            - claimStatus (ENUM: PENDING, APPROVED, REJECTED)
            - adjusterId (FK)

3.3 Customer Support Module

Purpose: Provides support for policyholders and claimants, allowing them to track claims, ask questions, and resolve issues.

- **Controller**:
    - SupportController
        - createTicket(ticketData)
        - getTicketDetails(ticketId)
        - resolveTicket(ticketId)
        - getAllTicketsByUser(userId)
- **Service**:
    - SupportService
        - Handles the creation and resolution of customer support tickets.
- **Model**:
    - **Entity**: SupportTicket
        - Attributes:
            - ticketId (PK)
            - userId (FK)
            - issueDescription (TEXT)
            - ticketStatus (ENUM: OPEN, RESOLVED)
            - createdDate (DATE)

3.4 Document Management Module

Purpose: Manages documents related to claims, such as medical reports, policy documents, and other supporting materials.

- **Controller**:
  - DocumentController
    - uploadDocument(documentData)
    - getDocumentDetails(documentId)
    - deleteDocument(documentId)
- **Service**:
  - DocumentService
    - Handles the uploading, retrieval, and deletion of documents.
- **Model**:
  - **Entity**: Document
    - Attributes:
      - documentId (PK)
      - claimId (FK)
      - documentName (VARCHAR)
      - documentPath (VARCHAR)
      - documentType (ENUM: PDF, JPG, PNG, DOC)

3.5 User Authentication & Authorization Module

Purpose: Manages user login, registration, and access permissions.

- **Controller**:
  - AuthController
    - registerUser(userData)
    - loginUser(username, password)
    - logoutUser()
    - getUserProfile(userId)
- **Service**:
  - AuthService
    - Handles user authentication and role management.

- **Model**:
    - **Entity**: User
        - Attributes:
            - userId (PK)
            - username (VARCHAR)
            - password (VARCHAR, Encrypted)
            - role (ENUM: ADMIN, AGENT, CLAIM_ADJUSTER, POLICYHOLDER)
            - email (VARCHAR)

4. Database Schema

4.1 Table Definitions

1. **Policy Table**

```
CREATE TABLE Policy (
  policyId INT PRIMARY KEY AUTO_INCREMENT,
  policyNumber VARCHAR(50),
  policyholderId INT,
  coverageAmount DECIMAL(10, 2),
  policyStatus ENUM('ACTIVE', 'INACTIVE', 'CANCELLED'),
  createdDate DATE,
  FOREIGN KEY (policyholderId) REFERENCES User(userId)
);
```

2. **Claim Table**

```
CREATE TABLE Claim (
  claimId INT PRIMARY KEY AUTO_INCREMENT,
  policyId INT,
  claimAmount DECIMAL(10, 2),
  claimDate DATE,
  claimStatus ENUM('PENDING', 'APPROVED', 'REJECTED'),
  adjusterId INT,
  FOREIGN KEY (policyId) REFERENCES Policy(policyId),
  FOREIGN KEY (adjusterId) REFERENCES User(userId)
);
```

3. **SupportTicket Table**

```
CREATE TABLE SupportTicket (
  ticketId INT PRIMARY KEY AUTO_INCREMENT,
  userId INT,
  issueDescription TEXT,
  ticketStatus ENUM('OPEN', 'RESOLVED'),
  createdDate DATE,
  FOREIGN KEY (userId) REFERENCES User(userId)
);
```

4. **Document Table**

```
CREATE TABLE Document (
    documentId INT PRIMARY KEY AUTO_INCREMENT,
    claimId INT,
    documentName VARCHAR(100),
    documentPath VARCHAR(255),
    documentType ENUM('PDF', 'JPG', 'PNG', 'DOC'),
    FOREIGN KEY (claimId) REFERENCES Claim(claimId)
);
```

5. **User Table**

```
CREATE TABLE User (
    userId INT PRIMARY KEY AUTO_INCREMENT,
    username VARCHAR(50) UNIQUE,
    password VARCHAR(255),
    role ENUM('ADMIN', 'AGENT', 'CLAIM_ADJUSTER', 'POLICYHOLDER'),
    email VARCHAR(100)
);
```

5. Local Deployment Details

1. **Environment Setup:**
    - Install JDK 17 or .NET SDK 7.0.
    - Install MySQL or SQL Server locally.
    - Use Tomcat (Java) or Kestrel (ASP.NET Core) for local server hosting.
2. **Deployment Steps:**
    - Clone the repository to the local machine.
    - Configure the database connection settings in application.properties (Java) or appsettings.json (.NET).
    - Execute SQL scripts to set up the database schema.
    - Build and run the application locally.

6. Conclusion

This document provides the low-level design for the **Health Insurance Claim Management System**, detailing the key modules, database schema, and deployment strategy. The design ensures scalability, modularity, and security in managing insurance policies, claims, and user data, and it is fully compatible with both **Spring MVC** and **ASP.NET Core MVC** frameworks.