Project Documentation

# Project Title

InsurAI: Corporate Policy Automation and Intelligence System

# Problem Statement

In the insurance industry, managing corporate policies involves handling large volumes of documents, compliance checks, renewals, and customer queries. These processes are often manual, time-consuming, and prone to human errors. Corporate clients face delays in policy issuance, renewals, and claim processing, resulting in reduced efficiency and customer satisfaction. There is a need for an automated, intelligent system that streamlines policy management, ensures compliance, reduces manual intervention, and improves decision-making.

# Solution Approach

I propose building 'InsurAI,' a Corporate Policy Automation and Intelligence System. This system will leverage automation, Artificial Intelligence (AI), and a full-stack application architecture to streamline policy lifecycle management. Key features include:  
- Automated policy creation, updates, and renewals  
- AI-powered compliance verification  
- Intelligent chat assistant for corporate clients  
- Secure role-based dashboards for clients, underwriters, and administrators  
- Data-driven insights and reporting dashboards  
  
The system will reduce manual work, speed up policy processing, minimize errors, and enhance customer experience.

# Tech Stack

Since the project will be developed with a focus on Full Stack Java, the following technologies will be used:  
  
1. Frontend:  
 - React for UI  
 - HTML5, CSS3, JavaScript  
  
2. Backend:  
 - Java (Spring Boot for REST APIs)  
 - Hibernate/JPA for ORM  
  
3. Database:  
 - MySQL for relational data storage  
  
4. AI & Automation Components:  
 - Python (for AI/ML models, integrated via APIs)  
 - Rule Engines (Drools/Java-based)  
  
5. DevOps & Deployment:  
 - Git & GitHub for version control  
 - Deployment on cloud (Heroku, Render, or Railway)

6. Build & Dependency Management:  
 - Maven

7. Other Tools:  
 - Postman (API testing)  
 - Swagger (API documentation)  
 - VS Code (Development)

# Functional Requirements

1. **User Registration & Login** – Secure authentication for customers and admins.
2. **Policy Management** – View, update, and manage policy details.
3. **Claim Submission** – Upload claim details and documents.
4. **Claim Processing** – AI-powered document analysis and fraud detection.
5. **Claim Status Tracking** – Real-time updates for customers.
6. **Admin Dashboard** – Approve/reject claims, monitor fraud alerts.
7. **Notification System** – Email/SMS updates for claim progress.

# Non-Functional Requirements

* **Scalability:** Handle large volumes of claims and users.
* **Security:** Data encryption, JWT authentication, secure storage of documents.
* **Performance:** Quick response time for claim verification.
* **Reliability:** High availability and fault tolerance.
* **User Experience:** Simple and intuitive UI for customers and insurers.

# Data Collected

* **User Data:** Name, email, phone number, policy details, claim details.
* **Policy Data:** Policy ID, policy type, coverage details, premium details.
* **Claim Data:** Claim ID, claim amount, supporting documents (PDF, images).
* **Document Data:** Uploaded medical bills, accident reports, ID proofs.
* **Fraud Check Data:** Claim history, duplicate submissions, anomaly detection results.

Daily Progress Log

Day 1 (20-Aug-2025)

- Set up project repository in GitHub

- Configured Maven and environment variables

- Created initial project structure

Day 2 (21-Aug-2025)

Database Setup (MySQL 8):

* Installed MySQL 8 and MySQL Workbench.
* Successfully configured MySQL service and set the root password.
* Verified installation via Workbench.
* Created insurai database (noted that it already exists, so confirmed database setup).

Backend Setup (Spring Boot):

* Chose Maven as build tool.
* Used Spring Initializr to generate project skeleton with the following setup:
  + Group: com.insurai
  + Artifact: insurai
  + Dependencies: Spring Web, Spring Data JPA, MySQL Driver.
* Downloaded, extracted, and opened project in IntelliJ IDEA Community Edition.
* Updated application.properties with database connection details for MySQL.

Environment Prepared:

* IDE: IntelliJ IDEA Community configured for Java + Spring Boot.
* Database: MySQL running with insurai schema ready.
* API Testing tools identified (Postman/cURL).

Day 3-4 (22/23-Aug-2025)

Completed Insurance module

Day 4-5(23/25-Aug-2025)

Completed Employee module