Software Requirements Specification

# Table of Contents

1. Introduction   
 1.1 Purpose   
 1.2 Scope   
 1.3 Definitions, Acronyms, and Abbreviations   
 1.4 References   
  
  
2. Overall Description   
 2.1 Product Perspective   
 2.2 Product Features   
 2.3 User Classes and Characteristics   
 2.4 Operating Environment   
 2.5 Design and Implementation Constraints   
 2.6 Assumptions and Dependencies   
  
3. System Features   
 3.1 Feature 1   
 3.2 Feature 2   
 3.3 Feature 3   
  
4. External Interface Requirements   
 4.1 User Interfaces   
 4.2 Hardware Interfaces   
 4.3 Software Interfaces   
 4.4 Communication Interfaces   
  
5. Functional Requirements

6. Non-functional Requirements

# 1. Introduction

## 1.1 Purpose

The purpose of InsurAI is to automate and intelligently manage corporate insurance policies and claims. The system will streamline the policy lifecycle, reduce manual intervention, enhance compliance checks, and provide data-driven insights to administrators and corporate clients.

This SRS document describes the **functional and non-functional requirements**, system architecture, and data specifications for the development of InsurAI.

## 1.2 Scope

InsurAI will provide:

* Automated policy creation, updates, and renewals
* AI-powered compliance verification and fraud detection
* Role-based dashboards for administrators, underwriters, and corporate clients
* Intelligent chat assistance for queries
* Real-time claim status tracking
* Notifications via Email/SMS

The system will reduce processing time, minimize errors, improve customer satisfaction, and provide a secure platform for corporate insurance management.

## 1.3 Definitions, Acronyms, and Abbreviations

| **Term** | **Definition** |
| --- | --- |
| AI | Artificial Intelligence |
| JWT | JSON Web Token, used for secure authentication |
| ORM | Object-Relational Mapping |
| REST API | Representational State Transfer Application Programming Interface |

## 1.4 References

* Spring Boot Documentation: <https://spring.io/projects/spring-boot>
* React Documentation: <https://reactjs.org/docs/getting-started.html>
* MySQL Documentation: <https://dev.mysql.com/doc/>

# 2. Overall Description

## 2.1 Product Perspective

InsurAI is a **standalone full-stack system** integrated with AI components. It will interact with databases, REST APIs, and AI/ML modules. The product is web-based and cloud-deployed.

## 2.2 Product Functions

* User registration, login, and authentication
* Policy lifecycle management (creation, renewal, update)
* Claim submission and AI-based claim processing
* Fraud detection and anomaly alerts
* Real-time claim tracking and notifications
* Admin dashboards with analytics and reporting

## 2.3 User Characteristics

* **Corporate Clients:** Non-technical users submitting policies and claims
* **Administrators:** Insurance staff managing policies and claims
* **Underwriters:** Staff verifying compliance and risk assessment

Users should have basic computer skills and internet access.

## 2.4 Constraints

* System must comply with data privacy and security regulations (e.g., GDPR/ISO standards)
* Must handle concurrent users and claims efficiently
* Cloud deployment may require internet connectivity

## 2.5 Assumptions and Dependencies

* Users have access to modern browsers (Chrome, Firefox, Edge)
* AI modules are available via APIs and trained on relevant datasets
* Notifications via email/SMS assume third-party service integration

# 3. Specific Requirements

## 3.1 Functional Requirements

| **ID** | **Requirement** | **Description** |
| --- | --- | --- |
| FR-1 | User Registration & Login | Secure authentication for corporate clients and admins with password encryption and JWT-based sessions. |
| FR-2 | Policy Management | Users can view, create, update, and renew policies. Admins can approve or reject changes. |
| FR-3 | Claim Submission | Users upload claim details and supporting documents (PDF, images). |
| FR-4 | Claim Processing | AI-powered document analysis for validation, anomaly detection, and fraud alerts. |
| FR-5 | Claim Status Tracking | Users receive real-time updates on claim status. |
| FR-6 | Admin Dashboard | Monitor claims, approve/reject, view fraud alerts, generate reports. |
| FR-7 | Notification System | Email and SMS alerts for policy renewals, claim approvals, or rejections. |

## 3.2 Non-Functional Requirements

| **Requirement** | **Description** |
| --- | --- |
| Scalability | Must handle large volumes of policies and claims efficiently. |
| Security | Data encryption, secure JWT authentication, secure storage of documents. |
| Performance | Quick response time for all user interactions and claim processing. |
| Reliability | High availability, fault tolerance, and data backup mechanisms. |
| User Experience | Simple, intuitive, and responsive UI. |

## 3.3 Data Requirements

| **Data Type** | **Description** |
| --- | --- |
| User Data | Name, email, phone number, login credentials, policy details |
| Policy Data | Policy ID, type, coverage details, premium details, validity |
| Claim Data | Claim ID, amount, status, supporting documents (PDF/images) |
| Document Data | Uploaded medical bills, accident reports, ID proofs |
| Fraud Check Data | Claim history, duplicate submissions, anomaly detection results |

## 3.4 Interface Requirements

* **Frontend:** React, HTML5, CSS3, JavaScript
* **Backend:** Java with Spring Boot REST APIs
* **Database:** MySQL relational storage
* **AI & Automation:** Python ML models integrated via APIs
* **Notification Services:** Email/SMS API integration

## 3.5 System Architecture

* **Client-Server Architecture:**
  + Frontend (React) communicates with backend (Spring Boot REST APIs)
  + Backend interacts with MySQL database and Python AI modules via APIs
  + Role-based dashboards with secure authentication
* **Deployment:** Cloud-based using Heroku/Render/Railway

## 3.6 Other Requirements

* API documentation via **Swagger**
* Version control with **Git & GitHub**
* Testing with **Postman** and unit tests in Java

# 4. Technology Stack

| **Layer** | **Technology** |
| --- | --- |
| Frontend | React, HTML5, CSS3, JavaScript |
| Backend | Java, Spring Boot, Hibernate/JPA |
| Database | MySQL |
| AI/Automation | Python, Rule Engines (Drools) |
| DevOps | Git, GitHub, Maven, Cloud Deployment |
| Tools | VS Code, Swagger, Postman |

# 5.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.

# 6.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.