Software Requirements Specification

Introduction

Purpose

The purpose of this Software Requirements Specification (SRS) document is to define the requirements for the development of an **Insurance Management System**. The system aims to streamline the management of insurance policies for end users, ensuring that they have easy access to their insurance information, can receive timely notifications regarding policy expirations, and are provided with secure, reliable services.

This document outlines the **functional** and **non-functional** requirements, including user requirements, system constraints, and operational environment, to guide the design and development of the system. The goal is to create an intuitive, secure, and efficient platform for users to manage their insurance policies and associated tasks, such as policy renewal, claims processing, and product recommendations.

Key objectives of the system include:

- **Improved User Experience**: The system will display insurance information in a concise, clear, and intuitive manner to facilitate easy navigation and quick access to key details.
- **Automatic Categorization**: Insurance policies will be automatically categorized to enhance organization and user accessibility.
- **Proactive Notifications**: The system will notify users about policy expiration dates, reducing the likelihood of coverage lapses.
- **Personalized Recommendations**: Users will receive tailored insurance product recommendations based on their past history and current needs.
- **Data Security**: Strong security protocols will be implemented to protect sensitive user information, ensuring trust and privacy.

Scope

The scope of this Software Requirements Specification (SRS) document is to define the functional and non-functional requirements for the development of the **Insurance Management System**. This system is designed to assist end users in managing their insurance policies effectively and securely. The system will provide a platform for displaying insurance information, managing policy renewals, tracking claims, and ensuring data security. It will also offer personalized product recommendations and proactive notifications, all while maintaining a user-friendly interface.

Product perspective

System interfaces

The **Insurance Management System** will integrate with various external and internal systems to ensure smooth operation, data synchronization, and communication across different components. The following section outlines the system interfaces, including communication protocols, data formats, and integration points.

3.1. Internal System Interfaces

1. **Backend-Frontend Interface**
    - **Description**: The frontend web and mobile interfaces will communicate with the backend system (Java Spring Boot or Python Django/Flask) through HTTP requests.
    - **Protocol**: RESTful API over HTTPS.
    - **Data Format**: JSON for all data exchanges between the frontend and backend.
    - **Operations**: The system will use standard HTTP methods such as GET, POST, PUT, and DELETE to fetch, update, and manage user insurance information, claims, notifications, etc.
2. **Backend-Database Interface**
    - **Description**: The backend application will interact with the database (MySQL or PostgreSQL) to store, retrieve, and manage insurance data, user profiles, claims, and notifications.
    - **Protocol**: SQL queries through a Database Management System (DBMS) protocol.
    - **Data Format**: Structured data in SQL format.
    - **Operations**: CRUD operations (Create, Read, Update, Delete) for managing insurance policies, claims, user data, and notifications.
3. **Backend-Security Interface**
    - **Description**: The backend system will communicate with security services (OAuth 2.0, JWT, and encryption modules) to ensure secure user authentication and authorization, as well as data protection.
    - **Protocol**: HTTPS with SSL/TLS encryption for secure communication.
    - Operations

      :

        - User authentication via OAuth 2.0 or JWT tokens.
        - Encryption of sensitive data using AES-256 for storage and transmission.

3.2. External System Interfaces

1. **Third-Party Notification Service Interface**
    - **Description**: The system will integrate with a third-party notification service (e.g., email, SMS, push notifications) to send reminders for policy expiration and other notifications to users.
    - **Protocol**: HTTP/HTTPS for API communication.
    - **Data Format**: JSON or XML for data exchange, depending on the third-party service requirements.
    - Operations

      :

        - Send policy expiration reminders, product recommendations, and other user notifications.
        - Provide feedback on successful or failed notifications.
2. **Payment Gateway Interface (Optional, for future integration)**

   - **Description**: The system will integrate with external payment gateways for processing payments related to insurance renewals, claims, or other services in future releases.
   - **Protocol**: HTTPS with secure payment protocols (e.g., PCI-DSS for card payments).
   - **Data Format**: JSON for data exchanges such as transaction details and payment statuses.
   - Operations

     :

     - Initiate and process payments.
     - Retrieve payment confirmations and error messages.

3. **External Data Providers Interface (Optional, for future integration)**
   - **Description**: The system may integrate with external insurance data providers (e.g., government or industry data sources) to retrieve external policy information, fraud detection data, or other relevant information for user policy management.
   - **Protocol**: HTTP/HTTPS with RESTful APIs.
   - **Data Format**: JSON or XML for external data exchanges.
   - Operations

     :

     - Fetch external policy data and verification information.
     - Update internal records with external data as necessary.

4. **Web Interface**

- **Description**: The web interface will allow end users to view and manage their insurance policies, track claims, and receive notifications.
- **Technology**: HTML5, CSS3, JavaScript (React.js or Angular).
- **Data Format**: JSON (from backend to frontend).
- Operations

  :

  - View policy details, expiration dates, and status updates.
  - Interact with forms for claims, renewals, and product recommendations.
  - Receive notifications via in-app messages or email.

5. **Mobile App Interface**

- **Description**: The mobile app interface will allow users to interact with the insurance management system on their smartphones or tablets.
- **Technology**: React Native (cross-platform) or Swift (iOS) and Kotlin (Android).
- **Data Format**: JSON (from backend to frontend).
- Operations

  :

  - Display policy information, renewal reminders, claims status, and notifications.

   - Provide functionalities similar to the web interface, with mobile-specific optimizations for small screens and touch input.

6. **Internet Connectivity**

- **Description**: The system requires stable internet connectivity for real-time synchronization of data between the backend, frontend, and client devices.
- **Protocol**: TCP/IP with standard internet communication protocols.
- Operations

  :

  - Ensure seamless access to the system's functionalities from client devices.
  - Allow real-time notifications and updates from backend to client devices (via push notifications or emails).

7. **Firewall and VPN Interfaces**

- **Description**: A firewall will be used to protect internal servers from unauthorized access, and VPN access will be required for secure administrative functions.
- **Protocol**: IPsec or SSL VPN for secure access, along with firewall rules to block unauthorized traffic.
- Operations

  :

  - Prevent unauthorized external access to the backend systems.
  - Provide secure communication for administrative functions.

8. **API Gateway Interface**

- **Description**: The system will use an API Gateway to route requests between services, enabling centralized management of all API calls.
- **Protocol**: HTTP/HTTPS for RESTful API calls.
- **Data Format**: JSON for all API exchanges.

- **External System API Integration**
  - **Description**: The system may integrate with external services such as email/SMS gateways, payment processors, and data providers via API calls.
  - **Protocol**: HTTP/HTTPS for secure data exchange.
  - **Data Format**: JSON or XML, depending on the external service's requirements.

User interfaces

The **User Interface (UI)** of the Insurance Management System is a critical component designed to ensure users can easily interact with the system, view insurance information, manage their policies, and access key functionalities. The following section outlines the user interface requirements for the system, including functional requirements, design principles, and specific details for both web and mobile interfaces.

1. **Navigation and Layout**:
   - Main Menu

    : A top or side navigation bar that allows users to quickly access key features such as:

        - Insurance Information
        - Claims Management
        - Policy Renewal
        - Notifications
        - Settings and Profile

    - **Dashboard**: The homepage/dashboard must display a summary of the user's insurance policies, their expiration dates, upcoming renewals, and active claims. Users should be able to quickly access detailed views of each section from the dashboard.
    - **Search Functionality**: A search bar should be available at the top of the interface to allow users to quickly locate specific insurance policies, claims, or other related information by keyword, policy number, or other criteria.

2. **Insurance Information Display**:
    - Policy List View

      : Insurance policies should be listed in a clear and concise format, displaying the following key details:

        - Policy Type
        - Policy Number
        - Expiration Date
        - Coverage Details
        - Premium Amount

    - **Policy Detail View**: Users should be able to click on a policy to view additional details, including terms, conditions, and payment history.
3. **Claims Management**:
    - **Claims Overview**: A section where users can view the status of their claims (e.g., "Under Review," "Paid," etc.).
    - **Claims Detail View**: Users should be able to click on individual claims for more detailed information about the claim status, associated documents, and progress.
4. **Notifications and Reminders**:
    - **Policy Expiration Notifications**: Expiration date notifications should be prominently displayed on the dashboard or via pop-up reminders when a user accesses the system. These notifications should include a link to renew the policy.
    - **Personalized Recommendations**: The system should display product recommendations for policy renewals based on the user's insurance history, shown as suggestions in the "Recommendations" section.
5. **Security and Login**:
    - **Login Page**: A secure login page should allow users to access their accounts using their credentials, with support for OAuth 2.0 or JWT-based authentication. Password recovery options should be provided.

   - **Session Management**: The UI should clearly indicate when a session is about to expire and allow the user to extend their session without logging out.

6. **Help and Support**:
   - **Help Section**: A dedicated section containing FAQs, user guides, and support contact information.
   - **Tooltip and Help Tips**: Context-sensitive help tips and tooltips should appear as users hover over or focus on certain UI elements to provide additional information or guidance.

Hardware interfaces

This section outlines the hardware requirements and specifications for the infrastructure supporting the Insurance Management System (IMS). These requirements ensure the system's performance, scalability, and reliability.

1. **Processor**:
   The servers must be equipped with multi-core processors to ensure smooth processing of real-time requests and handling of large data volumes.
   - Minimum: Intel Xeon or equivalent
   - Recommended: High-performance processors with at least 4 cores (preferably 8 or more for better parallel processing)
2. **Memory (RAM)**:
   The system must have sufficient RAM to handle multiple concurrent users and large datasets.
   - Minimum: 16 GB RAM
   - Recommended: 32 GB or more, depending on traffic and data handling needs
3. **Storage**:
   Reliable and redundant storage is essential for ensuring data availability and safety. The servers must have:
   - Redundant storage solutions such as RAID for data redundancy and reliability
   - Minimum: 1 TB SSD storage for fast data access and processing
   - Recommended: 2 TB or more SSD or NVMe storage for high-speed data access and scalability
4. **Network Interface**:
   The servers must support high-speed networking for handling real-time transactions, notifications, and user data synchronization.
   - Minimum: 1 Gbps Ethernet connection
   - Recommended: 10 Gbps or higher for future-proofing and handling higher traffic volumes
5. **Backup and Disaster Recovery**:
   The system must support daily backups to ensure data safety in case of hardware failure. The servers should have:
   - Offsite backup solutions (cloud or remote storage) to ensure data is preserved in the event of a server crash
   - Minimum: Daily full-system backups
   - Recommended: Hourly incremental backups for mission-critical data

6. **Power Supply**:
   High-availability servers should be equipped with an uninterruptible power supply (UPS) to prevent data loss during power outages.
7. **Desktop Devices**:
   Users may access the system from desktops or laptops. These devices must meet the following requirements:

- Operating System: Windows (7 or above), macOS (10.10 or above), or Linux (Ubuntu 16.04 LTS or above)
- Processor: Minimum of Intel Core i3 or equivalent
- RAM: Minimum of 4 GB
- Storage: Minimum of 100 GB free space for the installation of web browsers and caching
- Network Interface: Wi-Fi or Ethernet connection with a minimum of 10 Mbps download speed

8. **Mobile Devices**:
   The system will have mobile applications available for both Android and iOS platforms. Client devices should meet the following requirements:

- Android

  :

  - Minimum: Android 8.0 (Oreo) or above
  - Processor: ARM-based architecture (Snapdragon 600 series or higher)
  - RAM: Minimum of 2 GB
  - Network Interface: Mobile data (4G or 5G) or Wi-Fi connection

- iOS

  :

  - Minimum: iOS 12.0 or above
  - Processor: A9 chip or higher (iPhone 6s and above)
  - RAM: Minimum of 2 GB
  - Network Interface: Mobile data (4G or 5G) or Wi-Fi connection

9. **Peripheral Devices**:
   Although not mandatory, peripheral devices such as printers or barcode scanners may be used for certain functionalities like printing insurance policies or scanning documents. These devices must be compatible with the operating system and software interface.

Requirements

Functions

The **Requirements Function** section defines the functional aspects of the Insurance Management System, providing a detailed overview of the specific features and behaviors that the system must support to meet user needs and business objectives. These requirements will guide the development and implementation of the system and are categorized based on user needs, system operations, and interactions within the operating environment.

1. Insurance Information Display

- **Description**: The system must display insurance information in a concise, clear, and intuitive manner.
- Functional Requirements

  :

  - The system will provide a user-friendly interface that presents insurance details such as policy types, expiration dates, coverage levels, and other key information in an easily accessible format.
  - Information will be grouped and displayed in logical categories, allowing users to find relevant details quickly.
  - The system must minimize the number of steps required to access insurance data, ensuring that users can view critical information without unnecessary navigation through multiple pages.

2. Improved Categorization of Insurance Types

- **Description**: The system must automatically categorize different types of insurance policies to improve organization and facilitate easy access.
- Functional Requirements

  :

  - The system will automatically categorize policies into predefined types (e.g., life insurance, health insurance, property insurance).
  - Users will be able to filter and sort policies based on category, expiration date, and other relevant criteria.
  - The categorization should improve system usability by enabling quicker and more intuitive access to specific types of insurance information.

3. Insurance Expiration Date Notifications

- **Description**: The system should proactively notify users one week before their insurance policy expiration date.
- Functional Requirements

  :

  - The system will monitor expiration dates for all active insurance policies.
  - A notification will be sent to users via email, SMS, or within the system at least seven days prior to the expiration of their policies.
  - Notifications will contain relevant details, such as the insurance policy name, expiration date, and renewal instructions.

    - The system will track whether the notification has been acknowledged by the user and remind them periodically until action is taken.

4. Product Recommendation Based on History

- **Description**: The system should recommend insurance products for renewal based on the user's past insurance history and needs.
- Functional Requirements

  :

    - The system will analyze user behavior and historical data to generate personalized recommendations for policy renewals or new products.
    - Recommendations will be displayed on the user dashboard, with information on why the suggested product is relevant based on their history.
    - The system will allow users to accept, decline, or request more information regarding the recommendations.

5. Data Security and Privacy Protection

- **Description**: The system must have strong security measures in place, particularly regarding the protection of personal information.
- Functional Requirements

  :

    - All user data, including personal details and insurance records, must be encrypted both during transmission (via SSL/TLS) and at rest (using AES-256 encryption).
    - The system will implement role-based access control (RBAC) to restrict data access based on user roles (e.g., end users, administrators).
    - User authentication must be handled securely using OAuth 2.0 or JWT for authorization, ensuring that only authorized individuals can access sensitive information.
    - The system will log all access attempts and data changes to detect and respond to any potential security breaches.

6. Claims Processing Progress Tracking

- **Description**: The system should provide real-time tracking of claims processing status.
- Functional Requirements

  :

    - The system will allow users to view the status of their claims, such as "under review," "approved," "paid," or "rejected."
    - Users will receive notifications regarding updates to the status of their claims, ensuring that they are kept informed throughout the process.
    - The claims tracking feature will be accessible from both the web interface and mobile apps.
    - A status dashboard will display the progress of all active claims, with the ability to drill down for more detailed information.

7. Operational Guidance and Help Tips

- **Description**: The system should provide operational guidance and help tips to assist users, particularly with new features.
- Functional Requirements

  :

  - Contextual help tips will be displayed on the user interface, explaining new features or guiding users on how to complete tasks (e.g., filing a claim, renewing a policy).
  - The system will include a help center accessible from the main menu, providing detailed documentation on system usage, FAQs, and troubleshooting steps.
  - The help system will be integrated with a search function to allow users to quickly find relevant assistance based on their queries.

Usability requirements

This section outlines the usability requirements for the Insurance Management System, which are aimed at ensuring that the system is user-friendly, intuitive, and accessible for all users, particularly end users who will interact with the system on a daily basis.

1. User Interface (UI) Design

- **Consistency**: The user interface must maintain consistent design elements across all pages and modules. This includes consistent placement of navigation controls, colors, fonts, and iconography. This ensures that users can easily recognize features and understand the interface without the need for extensive retraining.
- **Clarity and Simplicity**: The interface should be designed to minimize complexity. The information presented on the screen should be clear and concise. Visual clutter should be avoided, and the use of simple language should be prioritized to cater to users of all technical backgrounds.
- **Responsiveness**: The system must support responsive design, ensuring that it works effectively across various screen sizes and devices, including desktops, tablets, and smartphones. This will ensure that users have a seamless experience regardless of the device they are using.

2. Navigation

- Intuitive Navigation

  : The system must have an easy-to-understand navigation structure. The primary categories (e.g., Insurance Information, Claims Processing, Policy Renewal) should be easily accessible via a well-organized menu.

  - **Search Functionality**: A robust search functionality should be available to help users quickly locate information or policies. The search bar should support keywords and be accessible from all pages within the system.
- **Quick Access to Key Information**: Key information such as policy details, expiration dates, and claim status should be readily accessible within one or two clicks. Users should not need to search through multiple pages to find critical information.

3. Error Prevention and Handling

- **Error-Free Input**: The system must minimize user input errors by using features such as input validation, drop-down menus, and auto-complete suggestions wherever applicable.

- **Error Feedback**: If an error occurs (e.g., invalid input), clear and actionable error messages must be displayed. These messages should be easy to understand and provide users with guidance on how to resolve the issue.
- **Confirmation of Actions**: The system should provide confirmation dialogues for critical actions, such as submitting claims or modifying personal information, to prevent accidental changes.

4. Help and Support

- **Help Tips**: The system should provide on-demand help tips and instructions for users. These tips should be available on the user interface and provide guidance on how to use the features, particularly for complex tasks like policy renewal or claims processing.
- **Help Documentation**: A comprehensive online help guide must be available to users. This should cover the system's functionalities, troubleshooting steps, and frequently asked questions (FAQs).
- **Customer Support Access**: There should be easy access to customer support via multiple channels, including email, chat, or phone. The support section should be accessible from every page within the system.

5. Accessibility

- **Visual Accessibility**: The system should ensure that users with visual impairments can use it effectively. This includes compatibility with screen readers and providing options for users to adjust font sizes, contrast, and color schemes to meet their specific needs.
- **Keyboard Navigation**: The system must be fully navigable via the keyboard, allowing users to perform all tasks without the need for a mouse.
- **Language and Locale**: The system should support multiple languages and locales to cater to a wide range of users. The ability to switch between languages should be available within the user settings.

6. Personalization

- **User Preferences**: The system should allow users to customize their experience based on personal preferences. This includes customizing dashboard views, saving frequently accessed policies or actions, and configuring notification settings.
- **Personalized Recommendations**: The system should offer personalized insurance product recommendations based on the user's past history and preferences, providing a more tailored and relevant experience for each user (as detailed in **Req ID: 004**).

Performance requirements

This section outlines the performance requirements for the Insurance Management System. The system must be able to handle a high volume of data, provide fast response times, and operate reliably under various conditions. These performance metrics will ensure that the system can efficiently meet user needs while maintaining high standards of availability and responsiveness.

1. Response Time

- **System Response Time**: The system must respond to user requests within a reasonable time frame. The maximum response time for the following critical operations should be:
    - **Login/Authentication**: 2 seconds

    - **Policy Information Retrieval**: 3 seconds
    - **Claims Status Update**: 3 seconds
    - **Search Results Display**: 2 seconds
    - **Notification Alerts**: 5 seconds
- **Dashboard Load Time**: The home page/dashboard must load fully within 5 seconds for a typical user session. It should display relevant insurance policy details and notifications promptly.
- **Real-time Notifications**: The system must push notifications (e.g., for insurance expiration or claims status updates) to the user within 10 seconds of triggering events.

2. Throughput

- **Concurrent Users**: The system must support at least **5,000 concurrent users** accessing the system simultaneously without a significant degradation in performance.
- **Data Processing Throughput**: The system should be capable of processing at least **1,000 insurance policy updates** per minute without delays. The backend should handle large-scale updates, particularly during high traffic periods such as policy renewal deadlines.
- **Claims Processing**: The system must be able to process **500 claims per hour** with real-time updates on their status.

Design constraints

Modular Architecture: The system must be designed using a modular architecture, which will allow for easier maintenance, scalability, and future enhancements. Each component, such as user authentication, policy management, and claims tracking, must be loosely coupled and interact through defined interfaces or APIs.

Platform Independence: The system should be platform-independent, ensuring that it runs smoothly across a variety of operating systems, including Windows, macOS, Linux, and mobile platforms (iOS and Android). The web interface should be optimized for use across different browsers (Google Chrome, Mozilla Firefox, Safari).

Cross-Device Compatibility: The system must support cross-device functionality. This includes desktop and mobile devices, which should provide an identical user experience, with adaptive design principles for varying screen sizes (e.g., smartphones, tablets, desktops).