Software Requirements Specification

for

Insurance Workflow Automation Software

Version 1.0 approved

Prepared by

Aayush Nagar

Aayush Senapati

Abhiraj

Achintya Krishna

16-09-2023

Table of Contents

Revision History

Name	Date	Reason For Changes	Version

1. Introduction

1.1 Purpose

This SRS describes the software functional and nonfunctional requirements for release 1.0 of the Insurance Workflow Automation Software (IWAS). This document is intended to be used by the members of the project team that will implement and verify the correct functioning of the system. Unless otherwise noted, all requirements specified here are high priority and committed for release 1.0.

1.2 Document Conventions

<Describe any standards or typographical conventions that were followed when writing this SRS, such as fonts or highlighting that have special significance. For example, state whether priorities for higher-level requirements are assumed to be inherited by detailed requirements, or whether every requirement statement is to have its own priority.>
No document conventions were used

1.3 Intended Audience and Reading Suggestions

<Describe the different types of reader that the document is intended for, such as developers, project managers, marketing staff, users, testers, and documentation writers. Describe what the rest of this SRS contains and how it is organized. Suggest a sequence for reading the document, beginning with the overview sections and proceeding through the sections that are most pertinent to each reader type.>
This document serves as a reference to the software developers who will be involved in the implementation of the product, and it will be used for testing and quality assurance. Section 2.2 should be referred to by the software developer team to implement functionality.
Section 2.3-2.4 and 2.6can be used by the end user for information nd their implementation, and serves as a guide for simplicity in usage.

1.4 Product Scope

The Insurance Workflow Automation Software will make it convenient for a user to do any activity related to Insurance policy, through the use of a virtual assistant. The features implemented are elaborated on in section 2.2 - Product features

1.5 References

<List any other documents or Web addresses to which this SRS refers. These may include user interface style guides, contracts, standards, system requirements specifications, use case documents, or a vision and scope document. Provide enough information so that the reader could

access a copy of each reference, including title, author, version number, date, and source or location.>

In addition to this document, we recommend referring to the following resources for a comprehensive understanding of the insurance landscape:

https://www.icicilombard.com/ - Offers a virtual assistant to make it easier to do anything related to insurance.
https://www.hdfcergo.com/ - Offers a user-friendly website that is easy to navigate.

These references can provide valuable insights and inspiration for the development of IWAS.

2. Overall Description

2.1 Product Perspective

The Insurance Workflow Automation Software (IWAS) represents a dynamic and competitive solution within the realm of online insurance systems. As depicted in Figure 1, our context diagram illuminates the intricate web of external entities and system interfaces that define Release 1.0. It's crucial to underscore that IWAS is not merely a static product but a platform poised for continual evolution through a series of releases. Ultimately, our vision is to transform it into an all-encompassing, unified platform capable of seamlessly handling a diverse spectrum of insurance types, offering a comprehensive and integrated solution to our users.

2.2 Product Functions

<Summarize the major functions the product must perform or must let the user perform. Details will be provided in Section 3, so only a high level summary (such as a bullet list) is needed here. Organize the functions to make them understandable to any reader of the SRS. A picture of the major groups of related requirements and how they relate, such as a top level data flow diagram or object class diagram, is often effective.>

1. *User Management:*

- *Manage user accounts, roles, and permissions.*
- *Allow users to register, log in, and reset their passwords securely.*

2. *Insurance Application Processing:*

- *Enable users to submit insurance applications electronically.*
- *Validate and process insurance applications, including underwriting and risk assessment.*

3. *Document Management:*

- *Store and manage digital copies of insurance policies, claims, and related documents.*
- *Facilitate document retrieval and version control.*

4. *Claims Processing:*

- *Support the submission and tracking of insurance claims.x*
- *Automate claims processing, assessment, and approval workflows.*

5. *Policy Management:*

- *Administer and manage insurance policies, including policy creation, modification, and renewal.*
- *Handle policy endorsements and cancellations.*

6. *Premium Calculations:*

- *Calculate insurance premiums based on policy details, risk factors, and other variables.*
- *Handle premium payments and invoicing.*

7. *Reporting and Analytics:*

- *Generate reports and analytics on policy performance, claims, and financial data.*
- *Provide dashboards for data visualization and insights.*

8. *Customer Support and Communication:*

- *Enable communication between users, agents, and customer support teams.*
- *Provide chat, messaging, or ticketing systems for inquiries and support.*

9. *Workflow Automation:*

- *Automate insurance-related workflows, including policy issuance and claims processing.*
- *Reduce manual tasks and streamline business processes.*

10. *Mobile Access:*

- *Provide mobile-friendly interfaces or dedicated mobile apps for users on the go.*

2.3 User Classes and Characteristics

1. Insurance Agents and Brokers:

   Characteristics:
   - Professionals specializing in insurance sales and client services.
   - Require access to policy issuance, underwriting, and client management features.
   - Need tools for generating quotes, managing policies, and processing claims efficiently.

2. Claims Adjusters:

Characteristics:

- Responsible for evaluating and settling insurance claims.
- Require access to claims processing modules to review and assess claims data.
- Need tools for documentation, investigations, and communicating with policyholders.

3. Policyholders:

   Characteristics:

   - Customers who purchase insurance policies.
   - May interact with IWAS for policy inquiries, premium payments, and claim submissions.
   - Require a user-friendly interface for accessing policy details and submitting claims.

4. Managers and Executives:

   Characteristics:

   - Senior executives and management personnel within insurance companies.
   - Require high-level dashboards and reports for monitoring business performance.
   - May need customizable analytics and key performance indicators (KPIs) to make strategic decisions.

2.4 Operating Environment

OE-1: IWAS shall operate on the following browsers: Google Chrome and Firefox
OE-2: IWAS shall be hosted on Render. Render is a unified cloud to build and run all your apps and websites with free TLS certificates

OE-3: IWAS shall permit user access, if he/she is authorized from an Internet connection at the user's home.

2.5 Design and Implementation Constraints

CO-1: Security and Regulatory Compliance: This constraint requires IWAS to implement specific security measures and comply with industry and government regulations to protect sensitive data and ensure legal compliance.

2.6 User Documentation

The first time a new user accesses the system and on user demand thereafter, the system shall provide a README markdown file explaining all functions available to a particular user class.

2.7 Assumptions and Dependencies

AS-1: Users' systems should be connected to the Internet to use the software.
AS-2: Database should be running on the server at all times.

DE-1: IWAS requires third-party payment gateway integration to accept payments.
DE-2: IWAS depends on Render to host the software on the web

3. External Interface Requirements

3.1 User Interfaces

<Describe the logical characteristics of each interface between the software product and the users. This may include sample screen images, any GUI standards or product family style guides that are to be followed, screen layout constraints, standard buttons and functions (e.g., help) that will appear on every screen, keyboard shortcuts, error message display standards, and so on. Define the software components for which a user interface is needed. Details of the user interface design should be documented in a separate user interface specification.>

- *The user interface should be graphical and user-friendly, with clear and concise labels and instructions.*
- *The user interface should be consistent with other insurance software products, so that users can easily learn how to use it.*
- *The user interface should be customizable, so that users can change the layout and appearance to suit their preferences.*
- *The user interface should be accessible to users with disabilities.????*

3.2 Hardware Interfaces

<Describe the logical and physical characteristics of each interface between the software product and the hardware components of the system. This may include the supported device types, the nature of the data and control interactions between the software and the hardware, and communication protocols to be used.>

- *The software should be compatible with a variety of hardware platforms, including Windows, Mac, and Linux.*
- *The software should be able to run on a variety of hardware devices, including desktop computers, laptops, and tablets.*
- *The software should be able to connect to a variety of peripheral devices, such as printers, scanners, and barcode readers.?????*

3.3 Software Interfaces

<Describe the connections between this product and other specific software components (name and version), including databases, operating systems, tools, libraries, and integrated commercial components. Identify the data items or messages coming into the system and going out and describe the purpose of each. Describe the services needed and the nature of communications. Refer to documents that describe detailed application programming interface protocols. Identify data that will be shared across software components. If the data sharing mechanism must be implemented in a specific way (for example, use of a global data area in a multitasking operating system), specify this as an implementation constraint.>

- *The software should be able to interface with a variety of insurance software products, including claims management systems, billing systems, and customer relationship management systems.*
- *The software should be able to interface with a variety of databases, including relational databases and NoSQL databases.??*
- *The software should be able to interface with a variety of operating systems, including Windows, Mac, and Linux.*

3.4 Communications Interfaces

<Describe the requirements associated with any communications functions required by this product, including e-mail, web browser, network server communications protocols, electronic forms, and so on. Define any pertinent message formatting. Identify any communication standards that will be used, such as FTP or HTTP. Specify any communication security or encryption issues, data transfer rates, and synchronization mechanisms.>

- *The software should be able to send and receive email messages.*
- *The software should be able to access and process web pages.*
- *The software should be able to connect to a variety of network servers.*
- *The software should be able to exchange electronic forms.*

4. System Features

<This template illustrates organizing the functional requirements for the product by system features, the major services provided by the product. You may prefer to organize this section by use case, mode of operation, user class, object class, functional hierarchy, or combinations of these, whatever makes the most logical sense for your product.>

4.1 User Authentication and Authorization

<Don't really say "System Feature 1." State the feature name in just a few words.>

4.1.1 Description and Priority

<Provide a short description of the feature and indicate whether it is of High, Medium, or Low priority. You could also include specific priority component ratings, such as benefit, penalty, cost, and risk (each rated on a relative scale from a low of 1 to a high of 9).>

This is achieved through the login page, and is of a high priority, as it allows for users to have various different permissions based on their authority, and unrestricted access must be prevented.

4.1.2 Stimulus/Response Sequences

<List the sequences of user actions and system responses that stimulate the behavior defined for this feature. These will correspond to the dialog elements associated with use cases.>

Stimulus	Response
User accesses the system	System renders a login page
User enters login credentials	Verifies credentials and grants access if valid
User clicks on registration page	System renders a registration page

4.1.3 Functional Requirements

REQ-1: System should securely store user credentials (encryption or hashing)
REQ-2: Implement lockout mechanisms after failed attempts
REQ-3: Display error message on entering invalid credentials.

4.2 Policy Creation and Management

4.2.1 Description and Priority

This is a feature that allows users to create a policy. It also allows them to view a list of all their policies and manage them as well. This is a high priority feature as it involves the crux of the project.

4.2.2 Stimulus / Response sequences

Stimulus	Response
User selects ‘create new policy’	System presents a form to select details

User edits an existing policy	System updates the policy to reflect changes
User requests policy details	System displays required information

4.3.3 Functional Requirements

REQ-1: Policies should have mandatory entry fields
REQ-2: System should provide validation checks on policy data
REQ-3: Error handling for incomplete forms

4.3 Claims Processing

4.3.1 Description and Priority

This feature enables the user to submit tickets for insurance claims and track its progress. The priority is medium, under the assumption that insurance claims for a user is not as important as the other functionalities, albeit necessary to include in the system.

4.3.2 Stimulus / Response sequences

Stimulus	Response
User selects ‘file a claim’	System presents claim submission form
User checks on the current status of claim	System displays the required information

4.3.3 Functional Requirements

REQ-1: System must allow users to submit supporting documents for claims
REQ-2: Claims must go through validation process
REQ-3: Error handling must be implemented to ensure no information is missing.

4.4 Customer support Chatbot

4.4.1 Description and Priority

This feature offers an option to interact with a virtual chatbot that can assist with any needs of the user. It is an easy alternative to navigating around the page. The priority of this feature is medium.

4.4.2 Stimulus / Response sequences

Stimulus	Response
User clicks ‘chat with chatbot’	Chatbot starts a conversation and helps user with queries

4.4.3 Functional Requirements

REQ-1: The chatbot should use a language model to understand natural language.
REQ-2: It should provide a default response when the user prompt goes beyond its scope.

5. Other Nonfunctional Requirements

5.1 Performance Requirements

- Response Time: During initial claim submission the response time should be near instant (typically a few seconds). The response with the status of the claim can be within minutes to a few hours. The final approval/denial status should be provided typically within 24-48 hrs. Policy generation should take a few seconds to minutes depending on the complexity. The policy underwriting process can take from hours to even days based on the amount of automation being used in the process.
- Scaling: Implement load balancing mechanisms to distribute incoming requests evenly across multiple instances or servers. The system should add new virtual machines (VMs) when CPU utilization exceeds a certain threshold or when the number of concurrent users reaches a predefined limit.

5.2 Safety Requirements

- Implement database replication to maintain multiple synchronized copies of data on different servers. In case of a hardware failure, the system can switch to a standby server, ensuring data availability.
- Configure the system in a high availability cluster where multiple servers work together. If one server fails, another takes over seamlessly.

- Enable transaction logging in the database management system (DBMS) to record all changes to the database. Transaction logs can be used to recover data in case of a software failure.
- Conduct regular testing of failover and recovery scenarios to ensure that the system can respond effectively to hardware or software failures without data loss.

5.3 Security Requirements

- We'll be utilizing AES-256 encryption in order to use it with data at rest, this will include personally identifiable information (PII), financial records, policy documents, and any other sensitive data.
- Using TLS protocols during web based interactions will allow us to secure the data currently in transit.
- Collect comprehensive logs of network traffic, particularly for systems and components involved in encrypted data transmission. This includes web servers, application servers, load balancers, and VPN gateways. Continuously monitor SSL/TLS handshakes for anomalies or failures. Set up alerts for failed handshakes, expired certificates, or protocol version issues.
- Periodically perform penetration testing and red teaming exercises to assess the effectiveness of monitoring and incident response processes. Implement a SIEM system to centralize log collection, analysis, and correlation.

5.4 Software Quality Attributes

- Designing software in a modular fashion is a best practice that simplifies maintenance and upgrades, enhances code reusability, and promotes overall software quality. Identify different functional components and separate them logically. For example, you can have modules for user management, policy processing, claims processing, reporting, and so on.
- Use abstraction to create clear and simple APIs (Application Programming Interfaces) for interacting with each module. Document and define interfaces between modules to specify how they interact with each other.
- Provide comprehensive documentation for each module, including usage instructions, APIs, and any dependencies. Keep documentation up to date to aid maintainers during updates.
- Set up continuous integration and continuous deployment (CI/CD) pipelines to automate the build, testing, and deployment of individual modules.
- Write unit tests for each module to verify its functionality in isolation. Modular code is easier to test and maintain.

5.5 Business Rules

<List any operating principles about the product, such as which individuals or roles can perform which functions under specific circumstances. These are not functional requirements in themselves, but they may imply certain functional requirements to enforce the rules.>

User Authentication and Authorization: Only authenticated users with appropriate roles and permissions should access the system's functionalities. Authentication should be based on secure methods such as username/password or multi-factor authentication.

Role-Based Access Control (RBAC): Implement RBAC to define which individuals or roles can perform specific functions within the application. For example, administrators may have access to all features, while regular users have limited access.

Data Access Control: Ensure that data is accessed and modified by authorized users only. Implement access controls to restrict data access based on user roles and the principle of least privilege

6. Other Requirements

<Define any other requirements not covered elsewhere in the SRS. This might include database requirements, internationalization requirements, legal requirements, reuse objectives for the project, and so on. Add any new sections that are pertinent to the project.>

Appendix A: Glossary

<Define all the terms necessary to properly interpret the SRS, including acronyms and abbreviations. You may wish to build a separate glossary that spans multiple projects or the entire organization, and just include terms specific to a single project in each SRS.>

Appendix B: Analysis Models

<Optionally, include any pertinent analysis models, such as data flow diagrams, class diagrams, state-transition diagrams, or entity-relationship diagrams.>

Appendix C: To Be Determined List

<Collect a numbered list of the TBD (to be determined) references that remain in the SRS so they can be tracked to closure.>