Augmented and Automated Insurance Underwriting Using Machine Learning.

(BSE21-)

|  |  |  |  |
| --- | --- | --- | --- |
| No. | Names | Registration Number | Signature |
| 1 | KYAKA HERMAN CEASER | 15/U/6851/PS | ceaserbanks09@gmail.com |
| 2 | DDOMBO NASSER | 15/U/4898/EVE | dombonasser52@gmail.com |
| 3 | OLARA SAMUEL OBADIA | 17/U/9634/EVE | obadz.99@gmail.com |
| 4 | WANDA ERIC | 15/U/ |  |

**Software Requirements Specification**

**Document**

**Version: (1.0)** **Date: (08/20/2021)**

**Table of Contents**

1. Introduction

1.1 Purpose

1.2 Scope

1.3 Definitions, Acronyms, and Abbreviations

1.4 References

1.5 Overview

2. The Overall Description

2.1 Product Perspective

2.1.1 System Interfaces

2.1.2 Interfaces

2.1.3 Hardware Interfaces

2.1.4 Software Interfaces

2.1.5 Communications Interfaces

2.1.6 Memory Constraints

2.1.7 Operations

2.1.8 Site Adaptation Requirements

2.2 Product Functions

2.3 User Characteristics

2.4 Constraints

2.5 Assumptions and Dependencies

2.6 Apportioning of Requirements

3. Specific Requirements

3.1 External interfaces

3.2 Functions

3.3 Performance Requirements

3.4 Logical Database Requirements

3.5 Design Constraints

3.5.1 Standards Compliance

3.6 Software System Attributes

3.6.1 Reliability

3.6.2 Availability

3.6.3 Security

3.6.4 Maintainability

3.6.5 Portability

3.7 Organizing the Specific Requirements

3.7.1 System Mode

3.7.2 User Class

3.7.3 Objects

3.7.4 Feature

3.7.5 Stimulus

3.7.6 Response

3.7.7 Functional Hierarchy

3.8 Additional Comments

4. Change Management Process

5. Document Approvals

6. Supporting Information

# 1. Introduction

*To many consumers, buying insurance can be painful. Despite insurance companies’ substantial investments over the past several years in digitizing customer onboarding and policy binding, progress has been slow and incremental and for that, many companies have failed to meaningfully scale their efforts to modernize underwriting.*

## 1.1 Purpose

*The purpose of this document is to define the Software Requirement Specifications (SRS) that explains the informational, functional, behavioural and operational aspects of an Automated Insurance (AIS) system.*

*The SRS describes in detail the design of an Automated Insurance system that is augmented through use Machine learning models to automate the insurance process and reduce the amount of time taken for one to buy an insurance policy. It also* serves as a basis and helps to create system design, system verification and validation procedures.

*Timeliness of the underwriting process can be signiﬁcantly improved, instances of human error can be reduced and misunderstandings or knowledge gaps in the underwriters can be ﬁlled. The current underwriting completion time frame of weeks can be reduced signiﬁcantly with the assistance of automated decision making tools*

## 1.2 Scope

*The SRS scope includes a requirements model that defines the informational, functional behavioral and operational aspects of AAIU system under consideration. It will result in the implementation of an Automated Insurance System that will allow clients to buy insurance, get insurance quotes in a paperless way.*

*The Automated Insurance System will allow users to get insurance quotes after they enter a few details like their age, region, medical history and family background.*

*The system will use machine algorithms like SVM and logistic regression to assess the risk profile of the insurance applicant and attach the corresponding weights.*

*The system will then use the attached weights to validate whether a person is eligible for a certain insurance policy or not and if yes, the system would return the expected quote computed based on the weight.*

*There are two beneficiaries of this kind of system*

1. *The Insurance underwriter: whose work of assessing client’s risk profiles would have been automated.*
2. *The Insured/Client: whose insurance application process will be reduced from weeks to minutes/hours?*

*The SRS shall be subjected to periodic review and evaluation as per the requirements management process for verification of its coverage and completeness.*

*Revisions shall be performed in an iterative manner based on a rapid incremental delivery (agile process) model to elicit the emergent requirements of the system under consideration as AI systems continue to evolve over time to attain progressive maturity levels*

## 1.3 Definitions, Acronyms, and Abbreviations.

*AIS – Automated Insurance System*

*AAIU – Augmented and Automated Insurance Underwriting*

*SRS – Software Requirements Specification*

## 1.4 References

1. *Boodhun, N., & Jayabalan, M. (2018). Risk prediction in life insurance industry using supervised learning algorithms. Complex & Intelligent Systems, 4(2), 145–154.* [*https://doi.org/10.1007/s40747-018-0072-1*](https://doi.org/10.1007/s40747-018-0072-1)
2. *Destination AI - Augmented automated underwriting and the evolution of the life insurance market. (n.d.). Retrieved from Munich Re Automation Solutions Ltd: https://www.munichre.com/automation-solutions/en/company/newsroom/2019/destination\_ai\_aau.html*
3. *Quantiphi. (n.d.). Breaking through the challenges of underwriting with AI. Retrieved from Quantinphi Inc: https://quantiphi.com/breaking-through-the-challenges-of-underwriting-with-ai/*

## 1.5 Overview

*AIS will be a web based application baked with Machine learning models that will automate the insurance underwriting process that clients will use to get insurance quotes and buy their favorite policies whereas underwriters will use the system for recommendation and automation where by the system will help them assess risk profiles of different clients and tell them whether they are eligible for a particular policy or not.*

*System requirements specifications are broadly organized in terms of (a) Functional Requirements, (b) External Interface Requirements, and (c) Non-Functional Requirements*

*Requirement specifications are defined in terms of different formats including use cases, graphical methods, mathematical models, documentation, etc. or combination of these format*

# 2. The Overall Description

## 2.1 Product Perspective

*The AIS is a self-contained web application that will invoke a machine learning model if a client requests a quote or invokes the insurance buying process. The model will process data given to it and store the results into a database and then sent to the user who invoked the process.*

### 2.1.1 System Interfaces

*The AIS has three actors which include the agent, the customer and the underwriter, and one cooperating system.*

### 2.1.2 Hardware Interfaces

*A 64bit dual core processor machine with a minimum of 1 GB RAM and 10 GB of free space running Ubuntu or Windows 10 should suffice. The Computer should be able to connect to the internet.*

### 2.1.3 Software Interfaces

*MySQL Server 5.6 DB. The system should use MySQL Server 5.6 as its database component. Communication with the DB is through ODBC Connections. The system must provide SQL Data table definitions.*

*Python 3.7. The system is should be placed on a server running a scripting server side language like python 3.7 and above.*

*Cloud Storage: A Cloud storage system where the Machine learning model will be stored will also be required. Choices are Google cloud storage.*

*Serverless Functions: The AIS should have a serverless function that will be invoked when client’s data needs to be processed. We will use Google Cloud Run or Amazon Lambda Functions.*

*Web Browser: The system should have a web browser ie Either Google Chrome or Mozilla Firefox*

### 2.1.5 Communications Interfaces

*The AIS will communicate over standard HTTP protocol on port 80.*

### 2.1.6 Memory Constraints

*For better results, the memory footprint should be 1GB RAM and above.*

### 2.1.7 Operations

*Specify the normal and special operations required by the user such as:*

1. *The various modes of operations in the user organization*
2. *Periods of interactive operations and periods of unattended operations*
3. *Data processing support functions*
4. *Backup and recovery operations*

*(Note: This is sometimes specified as part of the User Interfaces section.) If you separate this from the UI stuff earlier, then cover business process type stuff that would impact the design. For instance, if the company brings all their systems down at midnight for data backup that might impact the design. These are all the work tasks that impact the design of an application, but which might not be located in software.*

## 2.2 Product Functions

|  |  |  |
| --- | --- | --- |
| Req. ID | Requirement Specification | Description |
| R-PD1 | System shall have specification for insurance quote use case for which AIS software is used | A use case of a client asking for an insurance quote. They enter their details into the form and click generate quote. |
| R-PD2 | System shall have specification for Risk Assesment(Underwriting) | e.g Risk Assesment Task  - classification  - Regression/prediction  - Support vector Machine  - Matching  - Labelling  - Automated Weight attachment according to data provided by the client  - Assess based on data provided whether client is eligible or not for an insurance policy. |
| R-PD3 | System shall have specification for buying insurance | Describes how the software onboard a client and guides them to buy a policy. |
| R-PD4 | The system shall have a specification for recommending policies | Based on the client’s data, the system should recommend to them suitable policies. |

## 2.3 User Characteristics

*The project The intended users should be computer literates and know the basics of browsing the web and upload documents.*

Agent needs the knowledge of computer as well as insurance policy domain Insurance

company employees need knowledge of Insurance domain as well as computer.

*Agent needs the knowledge of computer as well as insurance policy domain.*

*Agent needs the knowledge of computer as well as insurance policy domain Insurance*

*company employees need knowledge of Insurance domain as well as computer.*

## 2.4 Constraints

*The data that is to be used to train our model is limited to only one insurance provider and so it exposes our model to only a few insurance policies that are provided by that provider.*

## 2.5 Assumptions and Dependencies

*We assume that the data used will represent all insurance providers’ data and the rules applied in the underwriting process are universal.*

# 3. Specific Requirements

*This section contains all the software requirements at a level of detail sufficient to enable designers to design a system to satisfy those requirements, and testers to test that the system satisfies those requirements. Throughout this section, every stated requirement should be externally perceivable by users, operators, or other external systems. These requirements should include at a minimum a description of every input (stimulus) into the system, every output (response) from the system and all functions performed by the system in response to an input or in support of an output. The following principles apply:*

1. *Specific requirements should be stated with all the characteristics of a good SRS*
2. *correct*
3. *unambiguous*
4. *complete*
5. *consistent*
6. *ranked for importance and/or stability*
7. *verifiable*
8. *modifiable*
9. *traceable*
10. *Specific requirements should be cross-referenced to earlier documents that relate*
11. *All requirements should be uniquely identifiable (usually via numbering like 3.1.2.3)*
12. *Careful attention should be given to organizing the requirements to maximize readability (Several alternative organizations are given at end of document)*

*Before examining specific ways of organizing the requirements it is helpful to understand the various items that comprise requirements as described in the following subclasses. This section reiterates section 2, but is for developers not the customer. The customer buys in with section 2, the designers use section 3 to design and build the actual application.*

*Remember this is not design. Do not require specific software packages, etc. unless the customer specifically requires them. Avoid over-constraining your design. Use proper terminology:*

*The system shall… A required, must have feature*

*The system should… A desired feature, but may be deferred till later*

*The system may… An optional, nice-to-have feature that may never make it to implementation.*

*Each requirement should be uniquely identified for traceability. Usually, they are numbered 3.1, 3.1.1, 3.1.2.1 etc. Each requirement should also be testable. Avoid imprecise statements like, “The system shall be easy to use” Well no kidding, what does that mean? Avoid “motherhood and apple pie” type statements, “The system shall be developed using good software engineering practice”*

*Avoid examples, this is a specification, a designer should be able to read this spec and build the system without bothering the customer again. Don’t say things like, “The system shall accept configuration information such as name and address.” The designer doesn’t know if that is the only two data elements or if there are 200. List every piece of information that is required so the designers can build the right UI and data tables.*

## 

## 3.1 External Interfaces

*This contains a detailed description of all inputs into and outputs from the software system. It complements the interface descriptions in section 2 but does not repeat information there. Remember section 2 presents information oriented to the customer/user while section 3 is oriented to the developer.*

*It contains both content and format as follows:*

1. *Name of item*
2. *Description of purpose*
3. *Source of input or destination of output*
4. *Valid range, accuracy and/or tolerance*
5. *Units of measure*
6. *Timing*
7. *Relationships to other inputs/outputs*
8. *Screen formats/organization*
9. *Window formats/organization*
10. *Data formats*
11. *Command formats*
12. *End messages*

## 3.2 Functions

*Functional requirements define the fundamental actions that must take place in the software in accepting and processing the inputs and in processing and generating the outputs. These are generally listed as “shall” statements starting with "The system shall…*

*These include:*

1. *Validity checks on the inputs*
2. *Exact sequence of operations*
3. *Responses to abnormal situation, including*
4. *Overflow*
5. *Communication facilities*
6. *Error handling and recovery*
7. *Effect of parameters*
8. *Relationship of outputs to inputs, including*
9. *Input/Output sequences*
10. *Formulas for input to output conversion*

*It may be appropriate to partition the functional requirements into sub-functions or sub-processes. This does not imply that the software design will also be partitioned that way.*

## 3.3 Performance Requirements

*This subsection specifies both the static and the dynamic numerical requirements placed on the software or on human interaction with the software, as a whole. Static numerical requirements may include:*

*(a) The number of terminals to be supported*

*(b) The number of simultaneous users to be supported*

*(c) Amount and type of information to be handled*

*Static numerical requirements are sometimes identified under a separate section entitled capacity.*

*Dynamic numerical requirements may include, for example, the numbers of transactions and tasks and the amount of data to be processed within certain time periods for both normal and peak workload conditions.*

*All of these requirements should be stated in measurable terms.*

*For example,*

*95% of the transactions shall be processed in less than 1 second*

*rather than,*

*An operator shall not have to wait for the transaction to complete.*

*(Note: Numerical limits applied to one specific function are normally specified as part of the processing subparagraph description of that function.)*

## 3.4 Logical Database Requirements

*This section specifies the logical requirements for any information that is to be placed into a database. This may include:*

1. *Types of information used by various functions*
2. *Frequency of use*
3. *Accessing capabilities*
4. *Data entities and their relationships*
5. *Integrity constraints*
6. *Data retention requirements*

*If the customer provided you with data models, those can be presented here. ER diagrams (or static class diagrams) can be useful here to show complex data relationships. Remember a diagram is worth a thousand words of confusing text.*

## 3.5 Design Constraints

*Specify design constraints that can be imposed by other standards, hardware limitations, etc.*

### 3.5.1 Standards Compliance

*Specify the requirements derived from existing standards or regulations. They might include:*

*(1) Report format*

*(2) Data naming*

*(3) Accounting procedures*

*(4) Audit Tracing*

*For example, this could specify the requirement for software to trace processing activity. Such traces are needed for some applications to meet minimum regulatory or financial standards. An audit trace requirement may, for example, state that all changes to a payroll database must be recorded in a trace file with before and after values.*

## 3.6 Software System Attributes

*There are a number of attributes of software that can serve as requirements. It is important that required attributes by specified so that their achievement can be objectively verified. The following items provide a partial list of examples. These are also known as non-functional requirements or quality attributes.*

*These are characteristics the system must possess, but that pervade (or cross-cut) the design. These requirements have to be testable just like the functional requirements. Its easy to start philosophizing here, but keep it specific.*

### 3.6.1 Reliability

*Specify the factors required to establish the required reliability of the software system at time of delivery. If you have MTBF requirements, express them here. This doesn’t refer to just having a program that does not crash. This has a specific engineering meaning.*

### 3.6.2 Availability

*Specify the factors required to guarantee a defined availability level for the entire system such as checkpoint, recovery, and restart. This is somewhat related to reliability. Some systems run only infrequently on-demand (like MS Word). Some systems have to run 24/7 (like an e-commerce web site). The required availability will greatly impact the design. What are the requirements for system recovery from a failure? “The system shall allow users to restart the application after failure with the loss of at most 12 characters of input”.*

### 3.6.3 Security

*Specify the factors that would protect the software from accidental or malicious access, use, modification, destruction, or disclosure. Specific requirements in this area could include the need to:*

1. *Utilize certain cryptographic techniques*
2. *Keep specific log or history data sets*
3. *Assign certain functions to different modules*
4. *Restrict communications between some areas of the program*
5. *Check data integrity for critical variables*

### 3.6.4 Maintainability

*Specify attributes of software that relate to the ease of maintenance of the software itself. There may be some requirement for certain modularity, interfaces, complexity, etc. Requirements should not be placed here just because they are thought to be good design practices. If someone else will maintain the system*

### 3.6.5 Portability

*Specify attributes of software that relate to the ease of porting the software to other host machines and/or operating systems. This may include:*

1. *Percentage of components with host-dependent code*
2. *Percentage of code that is host dependent*
3. *Use of a proven portable language*
4. *Use of a particular compiler or language subset*
5. *Use of a particular operating system*

*Once the relevant characteristics are selected, a subsection should be written for each, explaining the rationale for including this characteristic and how it will be tested and measured. A chart like this might be used to identify the key characteristics (rating them High or Medium), then identifying which are preferred when trading off design or implementation decisions (with the ID of the preferred one indicated in the chart to the right). The chart below is optional (it can be confusing) and is for demonstrating tradeoff analysis between different non-functional requirements. H/M/L is the relative priority of that non-functional requirement.*

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **ID** | **Characteristic** | **H/M/L** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** | **11** | **12** |
| 1 | Correctness |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 2 | Efficiency |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 3 | Flexibility |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 4 | Integrity/Security |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 5 | Interoperability |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 6 | Maintainability |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 7 | Portability |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 8 | Reliability |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 9 | Reusability |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 10 | Testability |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 11 | Usability |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 12 | Availability |  |  |  |  |  |  |  |  |  |  |  |  |  |

*Definitions of the quality characteristics not defined in the paragraphs above follow.*

*• Correctness - extent to which program satisfies specifications, fulfills user’s mission objectives*

*• Efficiency - amount of computing resources and code required to perform function*

*• Flexibility - effort needed to modify operational program*

*• Interoperability - effort needed to couple one system with another*

*• Reliability - extent to which program performs with required precision*

*• Reusability - extent to which it can be reused in another application*

*• Testability - effort needed to test to ensure performs as intended*

*• Usability - effort required to learn, operate, prepare input, and interpret output*

*THE FOLLOWING (3.7) is not really a section, it is talking about how to organize requirements you write in section 3.2. At the end of this template there are a bunch of alternative organizations for section 3.2. Choose the ONE best for the system you are writing the requirements for.*

## 3.7 Organizing the Specific Requirements

*For anything but trivial systems the detailed requirements tend to be extensive. For this reason, it is recommended that careful consideration be given to organizing these in a manner optimal for understanding. There is no one optimal organization for all systems. Different classes of systems lend themselves to different organizations of requirements in section 3. Some of these organizations are described in the following subclasses.*

### 3.7.1 System Mode

*Some systems behave quite differently depending on the mode of operation. When organizing by mode there are two possible outlines. The choice depends on whether interfaces and performance are dependent on mode.*

### 3.7.2 User Class

*Some systems provide different sets of functions to different classes of users.*

### 3.7.3 Objects

*Objects are real-world entities that have a counterpart within the system. Associated with each object is a set of attributes and functions. These functions are also called services, methods, or processes. Note that sets of objects may share attributes and services. These are grouped together as classes.*

### 3.7.4 Feature

*A feature is an externally desired service by the system that may require a sequence of inputs to effect the desired result. Each feature is generally described in as sequence eof stimulus-response pairs.*

### 3.7.5 Stimulus

*Some systems can be best organized by describing their functions in terms of stimuli.*

### 3. 7.6 Response

*Some systems can be best organized by describing their functions in support of the generation of a response.*

### 3.7.7 Functional Hierarchy

*When none of the above organizational schemes prove helpful, the overall functionality can be organized into a hierarchy of functions organized by either common inputs, common outputs, or common internal data access. Data flow diagrams and data dictionaries can be use dot show the relationships between and among the functions and data.*

## 3.8 Additional Comments

*Whenever a new SRS is contemplated, more than one of the organizational techniques given in 3.7 may be appropriate. In such cases, organize the specific requirements for multiple hierarchies tailored to the specific needs of the system under specification.*

*Three are many notations, methods, and automated support tools available to aid in the documentation of requirements. For the most part, their usefulness is a function of organization. For example, when organizing by mode, finite state machines or state charts may prove helpful; when organizing by object, object-oriented analysis may prove helpful; when organizing by feature, stimulus-response sequences may prove helpful; when organizing by functional hierarchy, data flow diagrams and data dictionaries may prove helpful.*

*In any of the outlines below, those sections called “Functional Requirement i” may be described in native language, in pseudocode, in a system definition language, or in four subsections titled: Introduction, Inputs, Processing, Outputs.*

# Change Management Process

*Identify the change management process to be used to identify, log, evaluate, and update the SRS to reflect changes in project scope and requirements. How are you going to control changes to the requirements? Can the customer just call up and ask for something new? Does your team have to reach consensus? How do changes to requirements get submitted to the team? Formally in writing, email or phone call?*

# Document Approvals

*Identify the approvers of the SRS document. Approver name, signature, and date should be used.*

# Supporting Information

*The supporting information makes the SRS easier to use. It includes:*

1. *Table of Contents*
2. *Index*
3. *Appendices*

*The Appendices are not always considered part of the actual requirements specification and are not always necessary. They may include:*

*(a) Sample I/O formats, descriptions of cost analysis studies, results of user surveys*

*(b) Supporting or background information that can help the readers of the SRS*

*(c) A description of the problems to be solved by the software*

*(d) Special packaging instructions for the code and the media to meet security, export, initial loading, or other requirements*

*When Appendices are included, the SRS should explicitly state whether or not the Appendices are to be considered part of the requirements.*

Tables on the following pages provide alternate ways to structure section 3 on the specific requirements. You should pick the best one of these to organize section 3 requirements.