Software Requirements Specification

for

Amazon Web Services Project

Version 1.1

Prepared by Jesus Garcia, Luis Lucero, Carlos Mendoza Jr, Bryan Perez, Nancy Rosa, Saul Rugama-Montenegro, Pasindu Siriwardena, Rongfeng Tan, Junli Wang

Sponsored by Commonwealth Casualty Company

05/13/2021

Table of Contents

Revision History

Name	Date	Reason For Changes	Version
Initial Creation	10/22/2020	Initial Creation of the document	1.0
Final Version	05/13/2021	Added the solid templating engine used in the system	1.1

1. Introduction

This document is the Software Requirement Specification (SRS) for the Commonwealth Casualty Company's (CCC) document templating engine with AWS. This application aims to provide an updated creation of contractual agreements within this company and ultimately have a modernized system for dealing with document generation.

1.1 Purpose

The purpose of this document is to inform readers about the Commonwealth Casualty Company's new document generation system with the use of Amazon Web Services. This new system will use more recent technology to modernize the current approach to ultimately make the entire platform run smoother and up to date. This document will show how this software application was created and the tools used to develop this software. This document is primarily intended for the developers of CCC and eventually, the final users (employees) responsible for handling customer documents.

1.2 Intended Audience and Reading Suggestions

This document is meant to be read by the software developers of Commonwealth Casualty Company to understand how the software functions and the tools used to create this software.

1.3 Product Scope

The new CCC's document generation system shall provide up to date features to CCC, which will enhance security, accuracy, and time efficiency. Authorized CCC employees shall easily generate needed documents pertaining to a client quickly and accurately in a secure manner.

This software application shall use a templating engine to create a document templating solution that enables dynamic document generation and shall have a user interface to view and modify templates. The software application shall have version controlling of templates, promotion of versions of templates into production by action or date, and comparison of documents generated from two different versions of the same document.

1.4 Definitions, Acronyms, and Abbreviations

All definitions, acronyms, and abbreviations are listed at the end of the document in Appendix A glossary.

1.5 References

Title: Spring PetClinic Sample Application
Authors: Dave Syer & Stephane Nicoll
Version/Date: Version 1.5.x / November 3, 2017
Location: https://github.com/PebbleTemplates/spring-petclinic

Title: Document Templating Engine
Authors: Mahan Hajianpour & Jeffrey Lipeles
Version/Date: Version 1.0.0 / September 18, 2020
Location: https://github.com/clucero36/senior_design

Title: Pebble Spring Boot Starter
Authors: Eric Bussieres & Hector Lopez
Version/Date: Version 3.1.4 May 28, 2020
Location: https://pebbletemplates.io/wiki/guide/spring-boot-integration/

Title: Pebble-Example-Spring
Authors: Mitchell Bösecke & Eric Bussieres
Version/Date: Version 4.0.0 / May 28, 2020
Location: https://github.com/PebbleTemplates/pebble-example-spring

2. Overall Description

This section gives background information about the specific requirements of this software application in a brief format. This section will not cover every requirement, but it will describe all the needed parts that affect the final product.

2.1 System Analysis

This project aims to build an interface that can convert data into an object and push this object into a template. This project aims to build an interface that can connect to a cloud database and easily send documents based on user information.

Some major technical hurdles associated with completing this project are time management, requirement understanding, code complexity, lack of structured features, and commercial licensing for the dependencies we are going to use.

A solution to overcome technical hurdles is communication, which can solve misunderstandings and keeps everyone on track of project deadlines. We are currently using Trello as a team management system which allows us to keep track of deadlines and assign members to tasks. A solution to the commercial licensing is to pay the licensing cost or find another dependency that is open source and allows for commercial for free.

2.2 Product Perspective

This software product is intended for CCC employees when the generation of a document is needed. This product will then communicate with the AWS servers and the templating engine to generate the current version of the necessary document.

We will use AWS for our database needs but there are many database options that we could have used. However, AWS, is a highly recommended and top-rated platform with secure access and more functionality compared to its competitors.

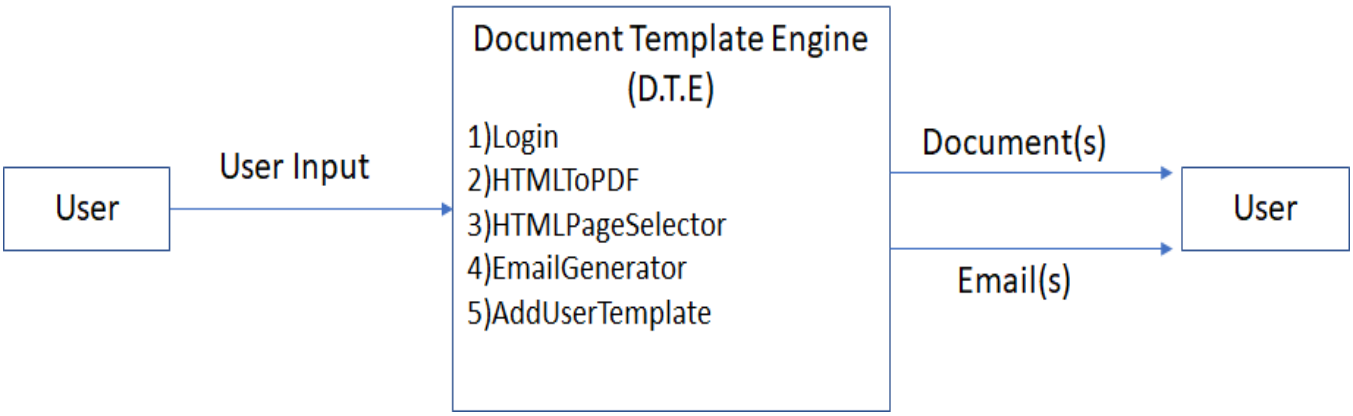
In terms of the templating engine, we are using the Pebble templating engine to generate our documents. Pebble is a modern templating engine with great features such as template

inheritance, built-in template security, functions, and more. It will provide creation of documents that will be available for any policy.

2.3 Product Functions (More Details)

Our project will implement already existing features from the current system at CCC. This includes a login module, and a collection of users and their policy information. Its current features include adding policy information to existing users and adding new customers. Some of the added features not in the old version are:

- Generating a Doc/PDF from any selected template
- Incorporating DocuSign into the Doc/PDF to allow digital signatures
- UI to view and modify templates
- Version Control including archival view and comparison of templates & generated documents
- Dashboard of current documents available for generation by product and state
- Provide a searchable/filterable list of all documents available to be generated by the system & allow filtering by state, product, dynamic/static
- Be able to send email with existing template-generated documents.



2.4 User Classes and Characteristics

The end user that is expected to use this software is the authenticated employees of CCC. The overall functionality of this software does not change, it only becomes enhanced and updated so

the end user will not be changed. No new authentication or rules will be imposed when switching to this software. This updated version will have a new layout and new functionality but will complete the same tasks as the current version CCC has implemented.

2.5 Operating Environment

The Operating Environment must include the following hardware platforms, operating systems, and software components or applications which coexist with the software.

- Windows/MacOS Operating System
    - Our software is built to run on the Windows OS as well as MacOS. Testing for other operating systems such as Linux or Ubuntu has not been conducted.
- AWS
    - Currently we are not using AWS because we want to first test out the software on smaller databases, but we plan to incorporate AWS in the future. AWS will manage all our document hosting and keep track of managing our clients' information.
- Github
    - Our project is currently located in a private repository on Github. Once complete, we may make the project public so that future groups can gain some insight into our work and improve on any downfalls we may have had.
- Terraform
    - Although not yet utilized, Terraform will be used in conjunction with AWS to manage and edit our infrastructure. It will be tested on our current SpringBoot application and then moved to a larger scale application.
- SpringBoot
    - We are currently using SpringBoot, as a backend for our software. By using SpringBoot we can test, edit, and manage our work efficiently and incorporate the many dependencies for functionality.
- Eclipse IDE
    - We have been using Eclipse as our primary development environment. It is user friendly, provides debugging tools, easy to navigate, and easily allows for the integration of our backend framework, SpringBoot.

2.6 Design and Implementation Constraints

When using software like AWS, many constraints can arise, this can influence the ability of software developers to implement the product. Some constraints include:

- Licensing
    - Since this software will be used at a large scale for an insurance company, we must be wary of the different dependencies and licenses we use. We must adjust our design to meet the terms and conditions of any components that we use.
- Safety and Security
    - When creating our software we take steps to ensure that this product can be used securely. We must make sure to encrypt data and access to the public and only show hidden and secure files to company agents.
- Hosting
    - Since we have created a software that requires hosting in the future, we must use a stable and secure hosting service. Right now it is running on personal tomcat servers but it should be transferred to a more secure location or to the existing servers that are present in the CCC company.

2.7 User Documentation

The software will not include any user documentation.

2.8 Assumptions and Dependencies

Factors that may affect the requirements if changed:

- Changing Templating engine (ex: Pebble to Thymeleaf)
    - Since we are using a modern templating engine we do not have as much access to documentation or examples to follow. If we decide to switch to another templating engine such as Thymeleaf we must change the syntax of our template files and adjust our controllers inside the software.
- Solidified the use of PebbleTemplates as it is more efficient for large scale applications (V2)

2.9 Apportioning of Requirements

The following requirements will be added if time constraints do not interfere. These requirements are not necessary for the software to work, they only add more functionality.

1. SNS Notifications
2. Server Health
3. Use of custom GraalVM runtime with Lambda

4. Email Implementation
5. PDF Conversion

3. External Interface Requirements

3.1 User Interfaces

The current system used at CCC uses XML and XSL as its markup language to generate documents. Our system will instead use a templating engine to generate document files that are easily edited and downloadable. The user interface will retain the same structure and design but the internals will be different. There will be a login page with authentication to make sure that only authorized employees of CCC are allowed to access user documents. When an authenticated user logs in, they will be able to perform various actions such as creating a new policy document, editing existing documents, and sending policy documents via email or downloading them as a PDF.

3.2 Hardware Interfaces

A user must use either a tablet, a laptop, or a desktop connected to the network of the company website to get access to the individual user page. Using any internet browser such as Chrome, Firefox, or Microsoft Edge, authorized users shall be able to enter the user page and choose the document they want to generate and send it as emails.

3.3 Software Interfaces

- Java, ver. 8+
- MySQL, ver. 5.7
- Maven, ver. 3.2
- SpringBoot, ver. 2.3.4
- PebbleTemplates, ver. 3.1.4

The front end software will be written in HTML, CSS and Bootstrap to create components and pages that are needed to view user documents. The back end software will be written in Java. The software currently uses MySQL, Maven and Spring Boot which will be modified later to fit the needs of AWS. Some stretch goals are to use lambda functions with GraalVM lambda.

3.4 Communications Interfaces

The backend software will communicate with the AWS(database) in order to pull the data to our model list. There will also be timers on certain models in order to display real time metrics on the dashboard. So far there is no limitation for the connecting tool, browser and Email type.

4. Requirements Specification

Requirements for each module are listed below:

4.1 Functional Requirements

Requirements Related to Design Element: Login Module	
Requirement No.	Requirement Description
1.1-1	The Login module shall check for username
1.1-2	The Login module shall check for password
1.1-3	The Login module shall store user login information
1.1-4	The Login module shall return error message when information is incorrect

Requirements Related to Design Element: Mainpage Module	
Requirement No.	Requirement Description
2.1-1	The Mainpage module shall display a home page.
2.1-2	The Mainpage module shall be able to access other modules.
2.1-3	The Mainpage module shall be able to receive user's input.

Requirements Related to Design Element: View/Edit Template Module	
Requirement No.	Requirement Description
3.1-1	The View/Edit Template module shall store data templates.
3.1-2	The View/Edit Template module shall be able to view stored templates.
3.1-3	The View/Edit Template module shall be able to edit stored templates.
3.1-4	The View/Edit Template module shall have version control of stored templates.
3.1-5	The View/Edit Template module shall be able to print the document.
3.1-6	The View/Edit Template module shall have a list of all documents.

Requirements Related to Design Element: Email Module	
Requirement No.	Requirement Description
4.1-1	The Email module shall be able to receive client's email from the user.
4.1-2	The Email module shall be able to be send emails to any email platform.
4.1-3	The Email module shall give confirmation of email being sent.
4.1-4	The Email module shall be able to send documents through email.

Requirements Related to Design Element: Template to PDF Module	
Requirement No.	Requirement Description
5.1-1	The Template to PDF module shall be able to convert a template into a PDF file.
5.1-2	The Template to PDF module shall combine multiple PDFs into one PDF file.
5.1-3	The Template to PDF module shall add page numbers to the PDFs.
5.1-4	The Template to PDF module shall customize page dimensions.

Requirements Related to Design Element: New Template Module	
Requirement No.	Requirement Description
6.1-1	The New Template module shall be able to take input from the user.
6.1-2	The New Template module shall be able to create a new template.
6.1-3	The New Template module shall be able to store the new template to the database.
6.1-4	The New Template module shall send confirmation of new template created

4.2 External Interface Requirements

The system shall let authenticated users login and view customers/clients that currently have a policy with CCC. These users shall be able to view each client's documents and make updates when necessary. Clients/customers may also be added to the company's database and sent various documents as PDFs. Users may also use the email module to email any document(s) to existing users. These emails are created within the software itself and sent to the user supplied email.

4.3 Logical Database Requirements

Currently, we have no specific logical database requirements. We will ultimately use AWS as our cloud database. Future versions of this document will present logical database requirements.

4.4 Design Constraints

We have not tested all the features for this project with the cloud database, so we do not have definite design constraints. After more testing has been concluded we will have revisions made. Currently we only have time constraints as there is a lot to complete before the deadline.

5. Other Nonfunctional Requirements

5.1 Performance Requirements

There are very standard requirements for document generation and any features within it. This process shall be able to generate a document in near real-time when an action tasks the generation engine to create a document or a set of merged documents.

5.2 Safety Requirements

There are no safety precautions that need to be taken when using this software.

5.3 Security Requirements

This software is meant to be used by authenticated employees of CCC. Users can then see clients/customers that are a part of the company. Since personal information is visible to these employees, precautions must be taken within the company to ensure that personal information is kept secure. No unauthorized persons shall be allowed to use this software.

5.4 Software Quality Attributes

The software as of right now is based on a Spring application that uses local Tomcat servers to run. So no matter the machine it is used on, the Spring initializer is made to run without any issues. Dependencies are currently static with no way to update unless future integrations are made to update platforms using github as version control.

5.5 Business Rules

Each user will only have one account and all users of this feature have the same rights. Our software can only be used by CCC. It cannot be used by any other company on any platform. Also, our template data will be generated by the CCC database.

6. Legal and Ethical Considerations

The team at CCC have ensured the safety of client's data and have maintained confidentiality for many years. After speaking with the team in charge of this project at CCC, we have concluded that our project does not contain any ethical or legal issues. The only specified consideration is that certain generated documents will be legally binding between the company and the insured.

Appendix A: Glossary

A
AWS - Amazon Web Services

C
CCC - Commonwealth Casualty Company
CSS - Cascading Style Sheets

H
HTML - HyperText Markup Language

I
IDE - Integrated Development Environment

O
OS - Operating System

P
PDF - Portable Document Format

S
SNS - Simple Notification Service
SQL - Structured Query Language
SRS - Software Requirement Specification

U
UI - User Interface

V
VM - Virtual Machine

X
XML - Extensible Markup Language
XSL - Extensible Stylesheet Language

Appendix B: Analysis Models

Data flow diagram: