SE 3XA3: Software Requirements Specification
FlagGenerator

Team #2, Team Jakriel
Akram Hannoufa, hannoufa
Ganghoon (James) Park, parkg10
Nathaniel Hu, hun4

February 11, 2022

Contents

List of Tables

List of Figures

Table 1: **Revision History**

Date	**Version**	**Notes**
January 25, 2022	1.0	Initial Document
February 10, 2022	1.1	Added draft of Functional Requirements Section and respective diagrams
February 11, 2022	1.2	Added draft of Non-functional Requirements
February 11, 2022	1.3	Added draft of Project Drivers and Project Issues
February 11, 2022	1.4	Made final minor review edits to SRS document

This document describes the requirements for FlagGenerator. The template for the Software Requirements Specification (SRS) is a subset of the Volere template (?). If you make further modifications to the template, you should explicity state what modifications were made.

1 Project Drivers

1.1 The Purpose of the Project

The purpose of this project is to create a user-friendly software that allows users to uniquely create a personalized random flag based on the entered input string. To enhance the interface, an extensive graphical user interface will be created. A uniquely generated flag can serve as a great inspiration to graphic designers, artists, or other media consumers and creators.

1.2 The Stakeholders

1.2.1 The Client

The clients of this project are the professor of SFWRENG 3XA3, Dr. Asghar A. Bokhari, and the teaching assistants (TAs) of this course. They will help guide the project by providing instructions, expectations, assistance, and feedback. They will also evaluate the project based on its functionality, organization, requirement, and documentation.

1.2.2 The Customers

The customers of this project will be media consumers and creators. These include users of social media websites and forums, players of nation building games, and any individuals who are want or need to create a unique flag.

1.2.3 Other Stakeholders

Members of Group 2 are our other stakeholders of this project as they will be working to successfully design, implement, test, document, and launch this project. In addition, the developers of PAGAN (Python Avatar Generator for Absolute Nerds) and other random generating software may also be stakeholders if they wish to implement our ideas, changes, and improvements.

1.3 Mandated Constraints

1. Time constraint: the entire project must be completed by April 12.
2. Application: the user must be able to operate the application.
3. Cost: the cost shall be free to users.

1.4 Naming Conventions and Terminology

Below is a table of terminologies used throughout the specification document

Table 2: Glossary of terms

Word/Phrase	Definition
Python	The programming language used in this project
Input String	The input of type string from the user
Gallery	Collection of previously generated flags
Hashing	Algorithm that converts input data to fixed-size value. Using a hash function to return an output that is usually a string or hexadecimal
UI	User interface is where interactions between machines and humans occur
GUI	Graphical user interface is a form of UI that allows users to use electronic devices to interact with graphics.
User	person who uses or operates a computer program
System/Program	collection of instructions or components that tell a computer how to operate

1.5 Relevant Facts and Assumptions

1.5.1 Relevant Facts

The PAGAN software assumes that users have no prior knowledge of programming. As such, our team will also assume that users have no prior knowledge of programming or of some of the terminology involved.

1.5.2 Assumptions

1. User has a keyboard and/or mouse input.
2. User has a basic understanding of how to use a computer.
3. User has a basic proficiency in English.

1.5.3 User Characteristics

1. User must be able to visually see.
2. User must be able to read and understand the instructions.

2 Functional Requirements

2.1 The Scope of the Work and the Product

2.1.1 The Context of the Work

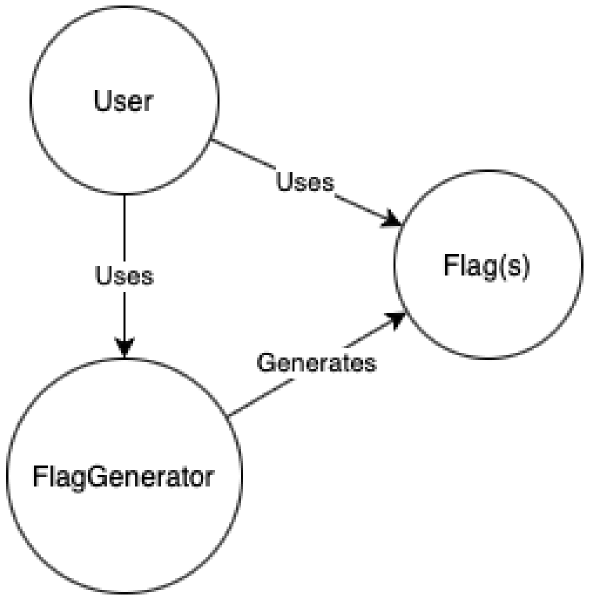

Figure 1: Context Diagram for FlagGenerator.

FlagGenerator is a standalone application that does not interact with other systems to operate. It consists of one actor, which is the user.

2.1.2 Work Partitioning

Table 3: Work Partitioning Events

Event Number	Event Name	Input	Output
1	Entering the main menu	Keyboard/Mouse	Main Menu, Input String Field
2	Entering a new input string	Keyboard/Mouse	Prompt
3	Generating a new flag	Keyboard/Mouse	Generated Flag
4	Reading the instructions	Keyboard/Mouse	Instructions
5	Opening the settings menu	Keyboard/Mouse	Different Flag Generation Settings
6	Viewing the gallery	Keyboard/Mouse	Generated Flags Gallery

Table 4: Work Partitioning Summaries

Event Number	Summary
1	The user, through the keyboard or mouse input, enters and is shown the main menu and input string field.
2	The user, through the keyboard or mouse input, enters a string and is shown a prompt to begin flag generation.
3	The user, through the keyboard or mouse input, starts the flag generation. After the flag generation has been completed, the user will be shown the generated flag.
4	The user, through keyboard or mouse input, chooses to read the instructions of using the FlagGenerator.
5	Before/after flag generation, the user can use keyboard or mouse input to open the settings menu. After the settings have been changed, the user can (re)generate a flag with the updated settings taking effect.
6	The user, though the keyboard or mouse input, views their own gallery of current (and previously) generated flags.

2.1.3 Individual Product Use Cases

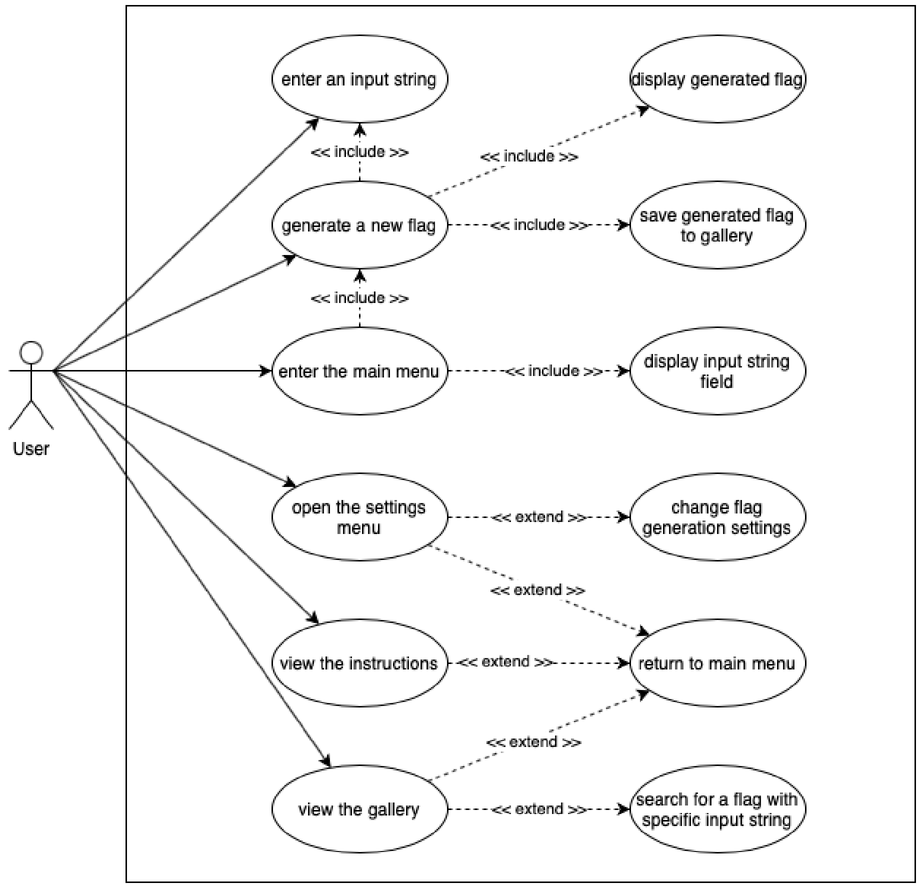

Figure 2: Use Case Diagram that displays the main functionalities of the application.

The above use case diagram shows the various ways a user can interaction with our application. Generating a flag involves entering in an input string, which is an included use case. The other uses cases such as view instructions, open settings menu and view gallery are self-explanatory.

2.2 Functional Requirements

BE1. The **user** enters the main menu.

- FR1. The **system** must present the **user** with the **main menu** and **input string field** for taking strings to generate flags from upon initializing the **GUI display**.
- FR2. The **system** must present the **buttons** to pull up the **instructions**, **flag gallery** and **settings menus**, or respond to the appropriate keyboard strokes.
- FR3. The **system** must allow the **user** to (re)enter in an input string into the **input string field** displayed on the **GUI**.
- FR4. The **system** must present the **user** with a **button** to enter in their input string and **start** the flag generation.
- FR5. The **system** must **present** the generated **flag** to the **user** in the main menu once it has finished being generated.
- FR6. The **system** must allow the **user** to **rerun** the flag generation after **settings** have been changed and **generate** a new flag that incorporates the changes to the settings.
- FR7. The **system** must **save** each generated **flag** to the user's personal **gallery**, using the **input string** as its (default) name, adding numbers as needed.

BE2. The **user** enters in a new **input string**.

- FR8. The **system** must present the **user** with a **button** to enter in their input string and **start** the flag generation.

BE3. The **user** starts the generation of a new **flag**.

- FR9. The **system** must present the generated **flag** to the **user** in the main menu once it has finished being generated.
- FR10. The **system** must allow the **user** to **rerun** the flag generation after **settings** have been changed and **generate** a new flag that incorporates the changes to the settings.
- FR11. The **system** must **save** each generated **flag** to the user's personal **gallery**, using the **input string** as its (default) name, adding numbers as needed.

BE4. The **user** reads the **instructions** for using FlagGenerator.

- FR12. The **system** must provide instructions to the **user** on how to use the **flag generator**.
- FR13. The **system** must provide a way for the **user** to return to the **main menu**.

BE5. The **user** opens the settings menu before/after generating a **flag**.

- FR14. The **system** must allow the **user** to change the **flag generator** settings.
- FR15. The **system** must specify the version of the **flag generator**.

FR16. The **system** must provide a way for the **user** to return to the **main menu**.

BE6. The **user** opens the gallery.

FR17. The **system** must present the **user** with a list of all **flags** and the **input string** used to generate them.

FR18. The **system** must allow the **user** to search for specific flags based on input string.

FR19. The **system** must provide a way for the **user** to return to the **main menu**.

3 Non-functional Requirements

3.1 Look and Feel Requirements

3.1.1 Appearance Requirements

LF1. A simple, intutitive and non-cluttered UI must be used.

3.1.2 Style Requirements

LF2. The program must generate high qulity images that are aesthetically pleasing.

LF3. Picture colours must be within certain colour ranges.

LF4. All image components must be visible in the generated image.

3.2 Usability and Humanity Requirements

3.2.1 Ease of Use Requirements

UH1. UI components must be placed in a logically flowing manner.

UH2. User must not have to jump between interfaces to accomplish a task.

UH3. Program must be easy-to-use for people aged 7 or older.

UH4. The program must use intuitive keyboard inputs to perform actions.

3.2.2 Personalization Requirements

UH5. User must be able to select output specifications (type of hashing, type of image file, etc.).

UH6. User must be able to modify their image gallery (add, delete).

3.2.3 Learning Requirements

UH7. The user should be able to use the program with no prior experience.

UH8. The user must be able to access a brief instructions blurb.

UH9. The user must be able to use the program by following the main UI instructions only.

3.2.4 Understandability Requirements

UH10. The program must use consistent language throughout.

UH11. The program must use simplified terminology wherever possible.

3.2.5 Accessibility Requirements

UH12. Program must use easy-to-read fonts and font sizes.

UH13. Program must provide a color-blind friendly UI.

3.3 Performance Requirements

3.3.1 Speed Requirements

PE1. Program must minimize the time taken to generate an image.

PE2. Program must minimize the time taken to process the downloading of an image.

PE3. Program must minimize the time taken to load in a user's gallery.

3.3.2 Safety-Critical Requirements

N/A

3.3.3 Precision or Accuracy Requirements

PE4. Different hashing systems must all deliver consistent and precise outputs.

PE5. Generated images must have accurately placed components (matching templates).

PE6. Colours in the generated image must be precise (the hexadecimal value).

3.3.4 Reliability and Availability Requirements

PE7. The program must be available to run anytime in the day.

3.3.5 Robustness or Fault-Tolerance Requirements

N/A

3.3.6 Capacity Requirements

PE8. User must be able to store up to a certain maximum number of generated images.

PE9. The program must limit the number of users to one per machine.

3.3.7 Scalability or Extensibility Requirements

PE10. Program should allow for the addition of other hashing functions.

PE11. Program should allow for the addition of other flag components.

3.3.8 Longevity Requirements

PE12. Program must be functional with exisiting software and hardware until Spring 2023.

3.4 Operational and Environmental Requirements

3.4.1 Expected Physical Environment

PE13. Program must not require an internet connection to function.

PE14. Program must run on any computer that can support the Python language.

3.5 Requirements for Interfacing with Adjacent Systems

PE15. Program must not alter files outside the working directory.

3.5.1 Productization Requirements

PE16. Program must be distributed as an executable file.

3.6 Release Requirements

RR1. The program must be released on or by April 12, 2022.

3.7 Maintainability and Support Requirements

3.7.1 Maintenance Requirements

MA1. Source code must be modularized with low cohesion.

MA2. Source code must be fully commented.

MA3. Full program documentation must be available.

MA4. Proper and consistent coding style must be followed.

3.7.2 Supportability Requirements

MA5. The program's repository must be made public to allow for issue raising.

3.7.3 Adaptability Requirements

MA6. This program must be able to be run on all major computer OS/platforms.

3.8 Security Requirements

3.8.1 Access Requirements

SR1. User must only be able to modify their own gallery.

SR2. User must only be able to generate flags for their gallery.

3.8.2 Integrity Requirements

SR3. User must be able modify their saved flag images (gallery).

SR4. In general, the input string must not be able to be deciphered from a generated image.

3.8.3 Privacy Requirements

SR5. A user's gallery must not be visible to other users.

SR6. User inputted strings must be deleted after image generation.

3.8.4 Audit Requirements

N/A

3.8.5 Immunity Requirements

SR7. Program must not have any obvious vulnerabilties.

3.9 Cultural Requirements

3.9.1 Cultural Requirements

CP1. Flag components must be respectful of countries' cultures.

CP2. Offensive flag components must not be included in possible flag components.

CP3. Input strings must have only valid hashable characters.

3.9.2 Political Requirements

CP4. No flag generated should promote any one political belief.

CP5. No flag generated should offend one's political views.

3.10 Legal Requirements

3.10.1 Compliance Requirements

LR1. Source code must follow the license under which it was obtained.

LR2. Personal information must be kept secure and local to the user.

3.10.2 Standards Requirements

N/A

3.11 Health and Safety Requirements

N/A
This section is not in the original Volere template, but health and safety are issues that should be considered for every engineering project.

4 Project Issues

4.1 Open Issues

None have been documented so far, and there are no open issues on PAGAN's repository at the time of this SRS.

4.2 Off-the-Shelf Solutions

There are similar products available, such as Scrontch's Flag Designer https://flag-designer.appspot.com/ and other flag generators; however, they do not take in an input string and they do not give users the option to personalize their flag. There are also existing products that use strings and hashing algorithms to generate random images, but not flags.

4.3 New Problems

Some problems that may exist after this software is launched include (poor) flag image quality, issues of copyright, and inappropriate use of flags. Users will be able to have high resolution images, but they may have a lower resolution flag than expected.

4.4 Tasks

Project tasks are scheduled and assigned. Please refer to project Gantt chart: ../ProjectSchedule/3XA3Ganttchart.pdf

4.5 Migration to the New Product

Since the software is simple and discrete, there is no need for migration to the new product. The gallery within the application will allow data to be stored.

4.6 Risks

This project has minimal risk because it is intended to be user-friendly and simple. There is, however, a risk of poor performance and unexpected application crashes. This is be minimized with intensive testing. There are also risks of low team productivity; however, this will be managed through our communication and problem solving skills.

4.7 Costs

There is no monetary cost for this project, as it will be developed as an open source software. The cost of time and effort are to be considered. This project is within the scope of the SFWRENG 3XA3 course. Therefore, the resource cost will be reasonable.

4.8 User Documentation and Training

4.8.1 Documentation

Software instructions will be included in a display window that is easily accessible from the main menu, for users to understand how to use the product along with its features.

4.8.2 Training

No training is required to use this application.

4.9 Waiting Room

Some features that may be added in later releases include:

1. Better resolutions and a GUI
2. Different language settings
3. Sharing options through social media

4.10 Ideas for Solutions

N/A

5 Appendix

This section has been added to the Volere template. This is where you can place additional information.

5.1 Symbolic Parameters

The definition of the requirements will likely call for SYMBOLIC_CONSTANTS. Their values are defined in this section for easy maintenance.