SOFTWARE REQUIREMENTS SPECIFICATION

Portfolio Builder

Team members: Kevin Kiussis, Natalie Tirabassi, Evelyn Landis, Djesse Jackson

1.0 Introduction

This section provides an overview of the entire requirement document. This document describes all data, functional and behavioral requirements for software.

1.1 Goals and objectives

Overall goals and software objectives are described. The Portfolio Builder aims to provide users with a simple and effective way to create and display their personal portfolios. The software will allow users to customize their portfolios, add relevant information such as work experience, projects, and skills, and present their portfolio in a visually appealing manner. The objective is to ensure a user-friendly experience with an intuitive interface that requires no prior technical knowledge.

1.2 Statement of scope

The Portfolio Builder will allow users to create and showcase personal portfolios. The major inputs include the following:

- A sign-up and login for user accounts.
- A prompt to choose a template and add content for customization.
- Display project details with images/links for user.
- Highlight skills and work history under a Skills/Experience section.
- A personal information section with user information allowing an employer to reach out more easily.

1.3 Software context

The Portfolio Builder is designed as a front-end-only web application that does not require server-side processing or external databases. It will operate entirely within a web browser, making use of local storage for saving user data. This ensures ease of use and removes dependencies on back-end frameworks. The platform is designed for students, freelancers, and professionals who want to create an online portfolio quickly and efficiently.

1.4 Major constraints

Any business or product line constraints that will impact the manner in which the software is to be specified, designed, implemented or tested are noted here.

- No Backend Dependencies: The system will not rely on external servers or databases.
- Browser Compatibility: The application must function across all major web browsers.
- Data Persistence: User data will be stored using local storage within the browser.
- Responsive Design: The layout should be mobile-friendly and able to adapt to different screen sizes.

2.0 Usage scenario

This section provides a usage scenario for the software. It organized information collected during requirements elicitation into use-cases.

2.1 User profiles

- General users or individuals such as students, freelancers, and professionals looking to create a portfolio.

2.2 Use-cases

- Sign-up/Login: Users create an account to save and manage their portfolio.
- Portfolio Creation: Users select a template and input their details.
- Project Showcase: Users upload images and links to display their work.
- Publish Portfolio: The user finalizes and views their generated portfolio.

2.3 Special usage considerations

Special requirements associated with the use of the software are presented.

- Offline Functionality: Users should be able to edit their portfolio without an internet connection, as long as they access it from the same device.
- No Server Storage: Data is saved locally using JavaScript local storage.
- Cross-Browser Compatibility: Ensures a smooth experience across different browsers.
- The design must be responsive to accommodate different screen sizes.

3.0 Data Model and Description

This section describes information domain for the software

3.1 Data Description

Data objects that will be managed/manipulated by the software are described in this section.

3.1.1 Data objects -Evelyn

User:

- *UserName**:***

  String/name used to identify and retrieve the account.

- *Password:*

  String that will be used to access and protect the account.

- *Confirmation:*

  Boolean to signify if the email has been confirmed.

- *Portfolios*

  Data Structure that will store all of the users published and current portfolios.

- *DeletePortfolio()*

  Function that deletes a portfolio from the Portfolios data structure.

- *AddPortfolio()*

  Function that adds a Portfolio to the Portfolio data structure.

- *PublishPortfolio()*

  Function that publishes the portfolio to the domain name.

- *EditExisting()*

  Function that allows the user to edit an existing portfolio.

- *ChagePassword()*

Function that allows the user to change their password.

- *ChangeEmal()*

  Function that allows the user to change their email address.

Portfolio:

- *Name*

  String that will be used to identify the portfolio. This will be set to "New_Portfolio#" until the user changes it.

- *Domain*

  String that will be used to publish the portfolio online. This will be set to Null until the user changes it.

- *Images*

  Data structure that holds the portfolio's image files.

- *TextBoxes*

  Data structure that holds the portfolio's text boxes and information.

- *Links*

  Data structure that holds the portfolio's links and their information.

- *AddImage()*

  This is a function that will be used to add an image into the Images data structure.

- *AddLink()*

  This is a function that will be used to add a link into the Links data structure.

- *AddTextBox()*

  This is a function that will be used to add a text box into the TextBoxes data structure.

- *EditImage()*

  This is a function that will allow the user to edit any of the existing images.

- *EditLink()*

  This is a function that will allow the user to edit any of the existing links.

- *EditTextBox()*

  This is a function that will allow the user to edit any of the existing text boxes.

- *ChangeDomain()*

  This is a function that will allow the user to edit the domain name.

- *ChangeName()*

  This is a function that will allow the user to edit the portfolio name.

User Library:

- *Users*

  Data Structure that stores all new and existing users.

- *ScanLibrary()*

  This is a function that scans the User Library for a user object profile.

- *AddUser()*

  This is a function that adds a user to the User Library.

- *DeleteUser()*

  This is a function that deletes a user from the User Library.

Portfolio Template Library:

- *Portfolios*

This is a data structure that stores pre-made portfolio templates.

- *ScanLibrary()*

  This is a function that searches the Template Library for a portfolio.

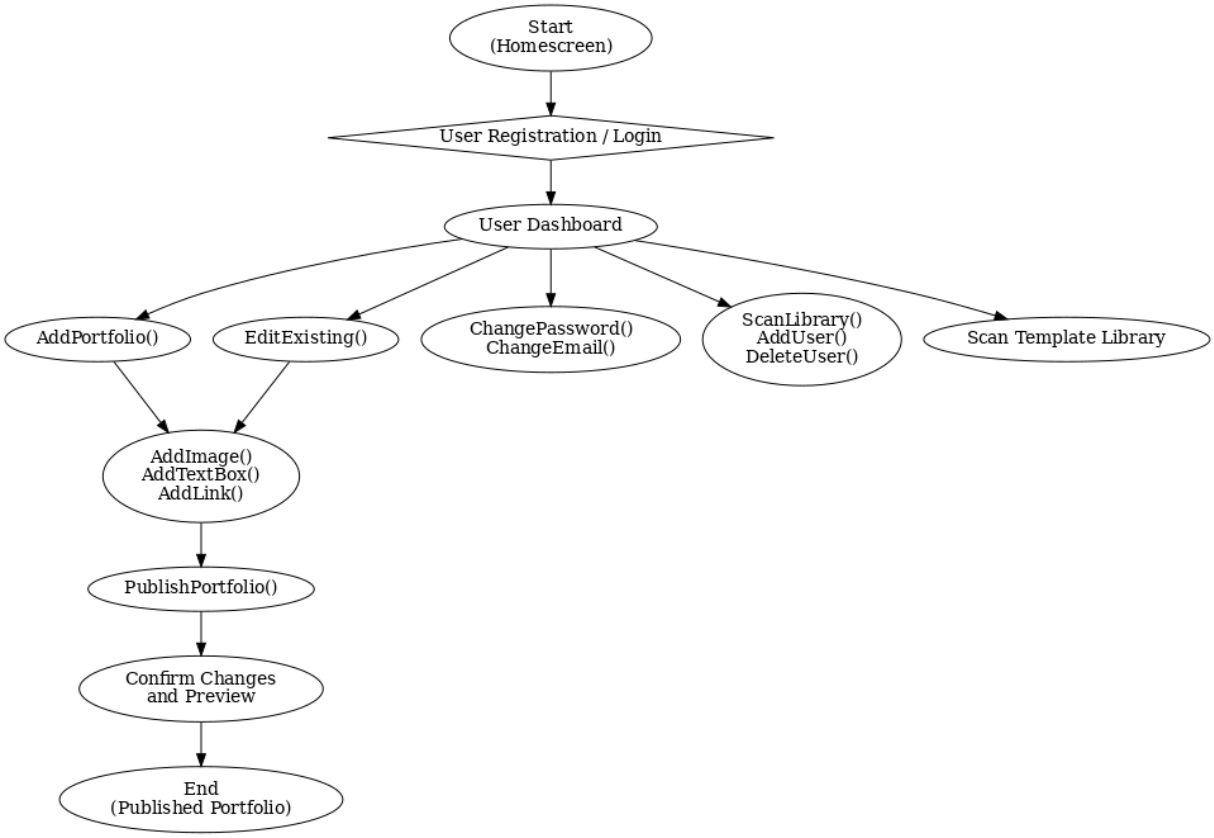
3.1.2 Relationships- Natalie

The following high-level relationships exist among the core data objects in the system:

- **User ↔ Portfolio**:
  A *User* can create, edit, and delete multiple *Portfolios*.
  A *Portfolio* belongs to exactly one *User*.
- **Portfolio ↔ Images / TextBoxes / Links**:
  Each *Portfolio* contains a collection of *Images*, *TextBoxes*, and *Links*.
  These are used to customize and structure the portfolio's content.
- **User Library ↔ User**:
  The *User Library* stores multiple *User* objects.
  Each user can be added or deleted by functions within the library.
- **Portfolio Template Library ↔ Portfolio**:
  The *Portfolio Template Library* contains a collection of pre-designed *Portfolio* templates that users can choose from when creating a new portfolio.

These relationships form the basis of our ERD in section 3.1.3.

3.1.3 Complete data model



3.1.4 Data dictionary

A reference to the data dictionary is provided. The dictionary is maintained in electronic form.

Variable Name	Variable Description	Variable Types	Variable Domain	Date Created
UserName	String is used to track a user account. Initially set to email.	String	320 char	3/23/2025
Password	String used to access an account. Password is initially set to a randomized value	String	64 char	3/23/2025
Confirmation	Boolean to prove that email is accurate. Initially set to false until email is confirmed.	Boolean	1 byte	3/23/2025
Name	String used to identify a portfolio. Initially set to	String	32 char	3/23/2025

	New_Portfolio# until set by user.			
Domain	Domain that will be used to publish the site. Initially set to NULL	String	253 char	3/23/2025
Image	Png, JPEG, GIF files uploaded into a portfolio.	image	256 MB	3/23/2025
TextBox	Text that could be modified inside a portfolio.	String	2,212 char	3/23/2025
Link	Text that links to other domains.	String	2048 char	3/23/2025

4.0 Functional Model and Description

A description of each major software function and software interface is presented.

4.1. Description of Major Functions

Each requirement is uniquely identified.

4.1.1 Requirement 1

Portfolio editing and generation functions must include basic text and image modification functions such as size and placement. It must also include link generation capabilities.

4.1.2 Requirement 2

Application must be intuitively navigable.

4.1.3 Requirement 3

Editable portfolio templets must be made available to the user and template library must be navigate able.

4.1.4 Requirement 4

The application must generate confirmation emails, save passwords, and protect users' work.

4.1.5 Requirement 5

Website publishing must be made available to the user and the user must be able to change their portfolio's domain name before publishing.

4.1.6 Requirement 6

User information must be saved to the user profile, and it must be changeable.

4.1.7 Requirement 7

Changes to user portfolios must be saved and the user must be able to easily re-access their work.

4.2 Software Interface Description

The Portfolio Builder software interacts with users and external systems through the following interfaces:

- A web-based Graphical User Interface (GUI**)**, developed using HTML and CSS for layout and styling, with interactivity powered by JavaScript and React.

- Backend API endpoints accessible via HTTP, used to retrieve and manipulate user data and portfolios.
- A secure login and registration interface provided.

4.2.1 External machine interfaces

The application may interact with external machines in the following way:

- Authentication servers for verifying user credentials and managing authentication state.
- Web browsers running on desktops, laptops, tablets, or mobile devices, used to access the Portfolio Builder website.

4.2.2 External system interfaces

- ***Web Server Interface:***

  To publish the website, we will need to interface with a webserver (through GitHub pages).

- ***Email Interface:***
  For email verification we will need to interface with an email server to send verification emails.

4.2.3 Human interface

- ***Homepage:***

  This Interface will be the first page a user will look at. It will have information about our product and a link to the login page.

- ***Register/ Login Page:***

  This Interface will allow the user to either select Register with an email address or login with the username and password.

- ***User Dashboard:***

  Once the user logins into their account, they will be taken into to the dashboard which will have access to the portfolio template library, the portfolio editor, and account information page.

- ***Portfolio Editor:***

  This Interface will have all portfolio editor tools like adding/editing images, adding text boxes, editing domain names, editing portfolio name, and adding links.

- ***Template Library Page:***

  This page will have all premade portfolios and will allow the user to select one to edit.

- ***Account Information Page:***

  This is the page will have all account information, and it will allow the user to change it if necessary.

5.0 Restrictions, Limitations, and Constraints

- Time Constraint: The project must be completed within seven weeks.
- Limited Team Resources: With only a small group, development must focus on essential features.
- Security Requirements: All user data must be securely stored and transmitted using HTTPS.

- Device Compatibility: The website must be responsive and work across desktop and mobile platforms.