# *SOFTWARE PROJECT FINAL REPORT*

Team members: Kevin Kiussis, Natalie Tirabassi, Evelyn Landis

Date: 4/17/2025

Table of Contents

List of Figures

List of Tables

## 1. Introduction

### 1.1. Purpose and Scope - Kevin

The purpose of this software application is to allow users to create and showcase their personal portfolios with an easy-to-use web application and utilize a user-friendly interface. Giving users the ability to customize themes/colors of their portfolio, insert projects and links to those projects, and tell more about themselves is the overall scope.

### 1.2. Product Overview (including capabilities, scenarios for using the product, etc.) Kevin

This product will have the following capabilities:

- Theme/Color Customization
- Custom Text Box Areas for user information such as:
  - Socials Section
  - Work Experience Section
  - Education Section
  - Projects Section
  - Contact Section
- Project Link Redirection for anyone viewing the user’s portfolio
- Realtime Portfolio Generation
- PDF Download for Portfolio for personal use

### 1.3. Structure of the Document – Kevin

The structure of the document is as follows with no additional changes to formatting.

### 1.4. Terms, Acronyms, and Abbreviations - Kevin

UI – User Interface, JS – JavaScript,

## 2. Project Management Plan - Natalie

### 2.1. Project Organization

The team worked in coordination, with designated roles:

- Kevin: Documentation, HTML architecture, test case management
- Natalie: JavaScript functions, requirements, and use-case descriptions
- Evelyn: CSS Styling and Formatting

## 2.2. Lifecycle Model Used

The project followed an Iterative Development Lifecycle Model, which allowed the team to build the application incrementally over multiple cycles, ensuring flexibility and continuous improvement. The lifecycle included the following phases:

- Phase 1: Planning and Requirements Gathering (Week 1)
  - Defined goals, gathered requirements, and planned team responsibilities.
- Phase 2: Design and Prototyping (Week 2)
  - Developed the software requirements specification (SRS), wireframes, and initial UI components.
- Phase 3: Core Development (Weeks 3-4)
  - Built major UI components including theme toggling, editable fields, and the portfolio editor.
- Phase 4: Feature Completion and Integration (Weeks 5-6)
  - Integrated PDF export, local storage functionality, and refined portfolio generation logic.
- Phase 5: Testing and Finalization (Week 7)
  - Conducted cross-browser testing, resolved defects, documented results, and finalized deliverables.

Each iteration provided opportunities for review, user testing, and refinement based on feedback and observations.

## 2.3 Risk Analysis

- Data loss due to local storage only (impact: medium, probability: low)
- Browser compatibility issues (impact: high, probability: medium)
- UI inconsistencies across devices (impact: medium, probability: medium)

## 2.4 Hardware/Software Resources

- Hardware: Standard web-enabled devices
- Software: HTML, CSS, JavaScript (React), GitHub Pages

## 2.5. Deliverables and schedule - Kevin

- **Week 1-2:** Planning and Design
- **Week 3-6:** Development and Implementation
- **Week 7:** Fine Tuning and Finalized Adjustments

## 3. Requirement Specifications-Natalie

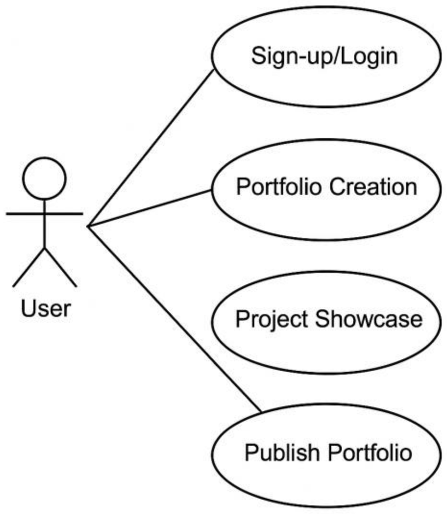
3.1. Stakeholders for the system

End users (students, freelancers, professionals), instructors

3.2. Use cases

- Sign-up/Login
- Portfolio Creation
- Project Showcase
- Publish Portfolio

3.2.1. Graphic use case model - Natalie



3.2.2. Textual Description for each use case - Kevin

- 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.

3.3. Rationale for your use case model

The use case model was designed to reflect the user-centered nature of the Portfolio Builder application. Each use case corresponds to a primary action a user would need to perform when using the software. The simplicity and clarity of the model help ensure a seamless user experience. By organizing the application into four distinct use cases—Sign-up/Login, Portfolio Creation, Project Showcase, and Publish Portfolio—we encapsulated all essential user interactions. This approach not only supports straightforward navigation and development but also aligns closely with the project's goal of providing an intuitive, front-end only solution for portfolio creation.

### 3.4. Non-functional requirements

- Cross-browser compatibility
- Offline functionality via local storage
- Responsive UI

## 4. Architecture - Natalie

### 4.1. Architectural style used

Client-side web app using a component-based architecture

### 4.2. Architectural model (includes components and their interactions)

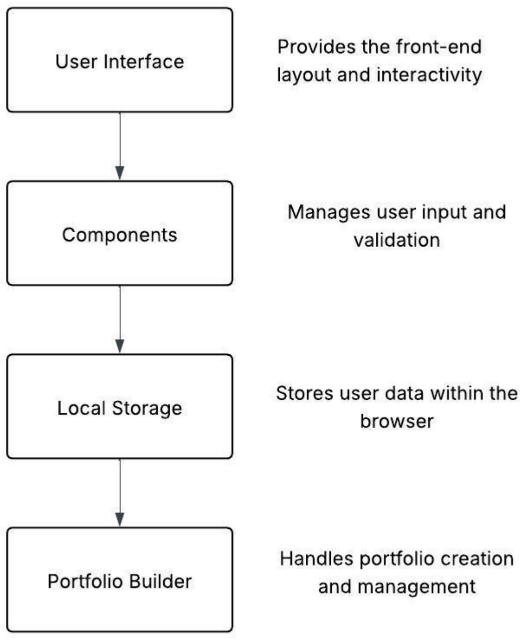
Architectural Model of the Portfolio Builder (see image below)

**Architectural Model:**

The architectural model includes the following layers:

1. **User Interface** – Provides the front-end layout and interactivity.
2. **Components** – Manages user input and validation.
3. **Local Storage** – Stores user data within the browser.
4. **Portfolio Builder** – Handles portfolio creation and management.

Each layer performs a dedicated function within the client-side application, and together they form the complete front-end architecture. The architecture of the Portfolio Builder is structured as a front-end only application, relying entirely on the client-side browser environment.


User Interface
Provides the front-end layout and interactivity
Components
Manages user input and validation
Local Storage
Stores user data within the browser
Portfolio Builder
Handles portfolio creation and management

### 4.3. Technology, software, and hardware used

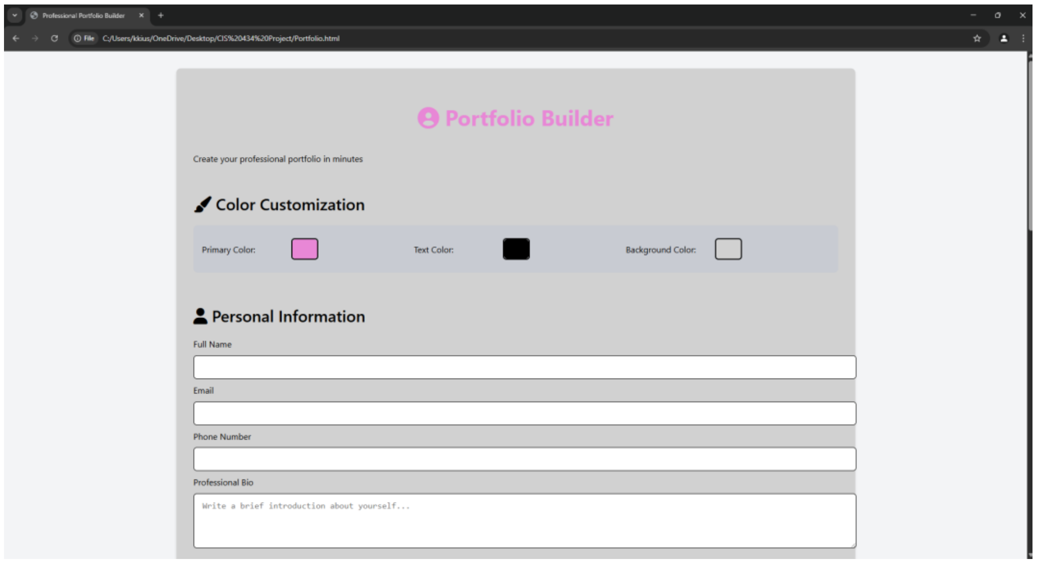
HTML/CSS, Javascript

### 4.4. Rationale for your architectural style and model

Chosen for ease of deployment, zero backend overhead, and accessibility

## 5. Design

### 5.1. User Interface design


Professional Portfolio Builder
File C:/Users/kkius/OneDrive/Desktop/CIS%20434%20Project/Portfolio.html

Portfolio Builder

Create your professional portfolio in minutes

Color Customization

Primary Color: 🟪 Text Color: ⬛ Background Color: ⬜

Personal Information

Full Name

Email

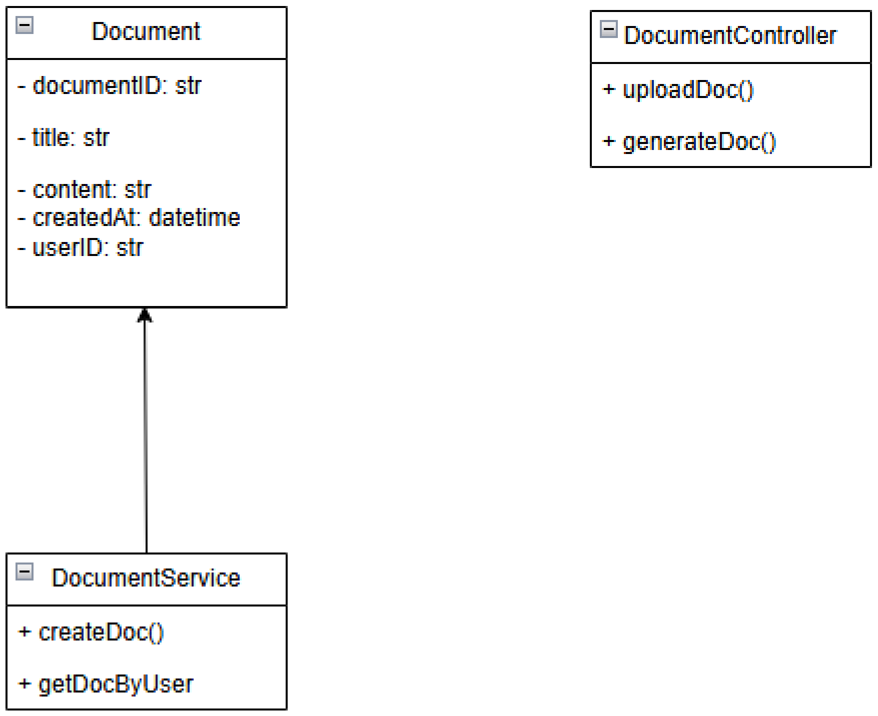
Phone Number

Professional Bio

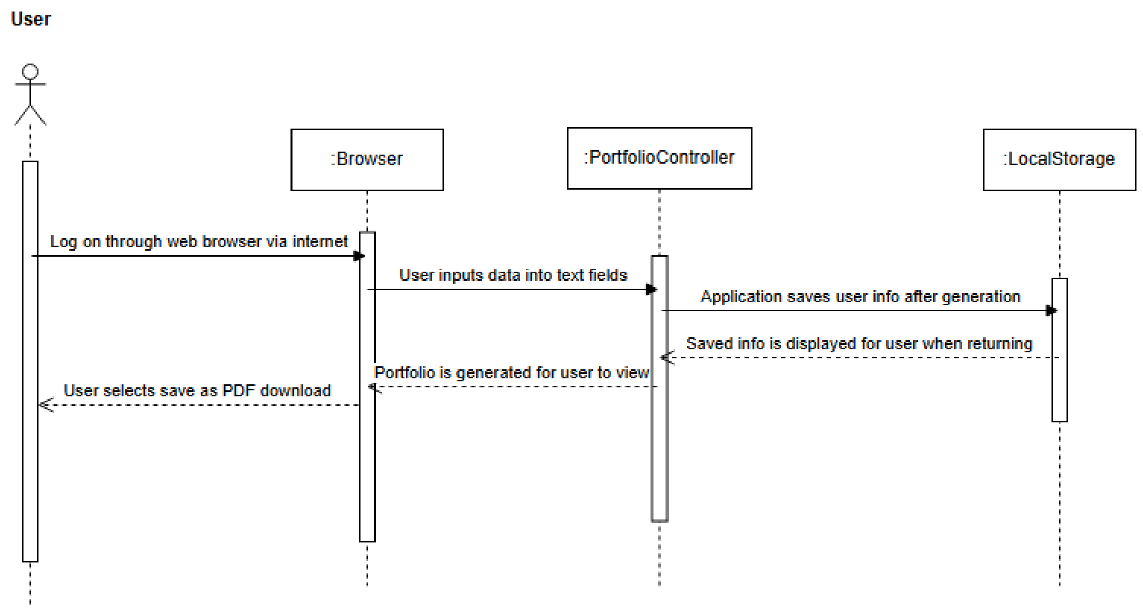
Write a brief introduction about yourself...

5.2. Components design (static and dynamic models of each component)

Below is a class diagram for some components (static model):



Below is a sequence diagram for the system as a whole (dynamic model):


User
:Browser
:PortfolioController
:LocalStorage
Log on through web browser via internet
User inputs data into text fields
Application saves user info after generation
Saved info is displayed for user when returning
Portfolio is generated for user to view
User selects save as PDF download

5.3. Database design - Kevin

This project consists of local storage using JavaScript, so no database was needed.

5.4. Rationale for your detailed design models

The design models were selected to ensure clarity, scalability, and adherence to best practices. UML diagrams allow clear system decomposition and modularity. MVC architecture promotes separation of concerns, making the system easier to test and maintain.

5.5. Traceability from requirements to detailed design models

Traceability was maintained by linking each functional requirement to corresponding use cases, components, and database entities. An example of this is below:

- **Requirement:** “Users can generate and store documents”
  - **Use Case:** "Document Creation"
  - **Component:** DocumentController, Document model
  - **Database:** Documents table

This traceability ensured that all requirements were addressed in the design and supported consistent validation and testing processes.

## 6. Test Management - Natalie

### 6.1. A complete list of system test cases

- Account creation/login
- Portfolio template selection
- Text/Image/Link edits
- Publish and PDF download

### 6.2. Traceability of test cases to use cases

Each test case directly corresponds to a primary use case:

- Account creation/login → Sign-up/Login use case
- Portfolio template selection → Portfolio Creation use case
- Editing content (text, images, links) → Project Showcase use case
- Portfolio finalization and PDF export → Publish Portfolio use case

### 6.3. Techniques used for test case generation

Test cases were generated using a combination of:

- Use case analysis
- Boundary value analysis for user input fields
- Exploratory testing to identify edge-case failures

### 6.4. Test results and assessments (how good are your test cases? How good is your software?)

All test cases passed under standard and cross-browser conditions. Manual testing across Chrome, Firefox, and Edge revealed no critical defects, except for one issue with background color scope. The software meets its intended use with a smooth, bug-free experience for users. The test suite was effective in uncovering issues early, and overall software quality is rated high for reliability and usability.

### 6.5. Defects reports - Kevin

There is one main defect that deals with the customization feature of the background color. The color is changing for the entire page instead of just the portfolio being generated.

## 7. Conclusions - Natalie

### 7.1. Outcomes of the project

All core functionality was implemented. The product meets its design goals of user-friendliness and efficient portfolio generation.

### 7.2. Lessons learned

- Creating modular UI elements allowed flexible layout adjustments and clear separation of responsibilities.
- Establishing clear goals and timelines from the beginning helped the team stay organized and meet deadlines.
- Testing on multiple browsers early helped catch display and functionality issues that would have otherwise gone unnoticed.
- Working without a backend simplified development but introduced challenges for data persistence and multi-device support.
- Frequent peer feedback and iteration led to improved usability and user interface consistency.

### 7.3. Future development

- The current implementation is fully functional as a front-end-only portfolio builder, but several improvements could be made in future iterations:
- User Authentication: Introduce account-based login with secure user management.
- Cloud Storage: Enable portfolio data to be saved to the cloud for cross-device access.
- Mobile App Version: Develop a mobile-native app for wider platform accessibility.
- Advanced PDF Export Options: Improve export quality with page formatting, branding, and templates.
- Custom Domain Support: Allow users to publish their portfolios with a personalized domain.
- Accessibility Features: Enhance usability for screen readers and keyboard navigation.

## References

No external libraries or sources were used beyond standard web development tools. All work was custom developed by the team.