SOFTWARE DESIGN SPECIFICATION

1.0 Introduction

This Software Design Specification (SDS) outlines the data structures, architectural design, component-level implementation, and interface considerations for the Portfolio Builder Website. The document ensures a systematic development process and alignment with the Software Requirements Specification (SRS).

1.1 Goals and objectives

The primary goal of this project is to provide users with a modern, intuitive web interface to create and publish personalized portfolios. The software aims to:

- Allow users to create accounts and manage secure logins
- Enter personal information, education, work experience, and projects
- Preview, style, and download their portfolio in PDF format
- Offer light/dark theme options
- Be fully responsive and accessible on desktop and mobile devices

1.2 Statement of scope

The Portfolio Builder software includes both frontend and backend components that:

- Allow users to register and log in securely
- Accept user input: name, bio, contact, skills, projects, work experience, education, and theme selection
- Dynamically render a live preview of the portfolio based on user input
- Enable exporting the portfolio as a downloadable PDF
- Support visual theme switching (light/dark mode)

Inputs: User form data (text, links, skills, dates) and login credentials **Processing:** Form validation, live rendering, session authentication, and PDF generation **Outputs:** A formatted, downloadable digital portfolio ready for sharing

1.3 Software context

This software serves students, job-seekers, and professionals looking to quickly generate a portfolio without coding. It includes a backend to support user authentication

and persistent storage of user-created portfolios. The frontend is built using HTML, CSS, and JavaScript. The backend is built using Node.js with Express.js.

1.4 Major constraints

- Must be completed within a 7-week timeline
- Frontend: HTML, CSS, JavaScript
- Backend: Node.js with Express.js
- Authentication: Custom login system
- Interface must be responsive and accessible
- PDF generation must accurately reflect user content

2.0 Data design

A description of all data structures and databases.

2.1 Data structures

- **User Object**:
  - `username`: Unique login name (String)
  - `email`: Contact Email (String)
  - `password`: Encrypted password (String)
  - `portfolios`: Array of Portfolio references
- **Portfolio Object**:
  - `name`: Portfolio title (String)
  - `bio`, `linkedinUrl`, `githubUrl`, `phone`: Profile data (Strings)
  - `skills`: Array of technical skills (Strings)
  - `projects`: Array of project objects (title, description, link)
  - `experience`: Job experience (role, company, date range, description)
  - `education`: Degree, institution, graduation year
  - `contactMessage`: Optional summary (String)
  - `theme`: Enum ("Light", "Dark")

2.2 Database description - Natalie

- The databse is used to store:
  - Users collection (with authentication data and references to portfolios)

    - Portfolios collection (all custom portfolio data per user)
- Mongoose models define schema constraints and relationships
- Passwords are encrypted before storage
- Sessions or tokens are used for user login management

3.0 Architectural and component-level design

A description of the software architecture is presented.

3.1 Architecture diagrams

Various views (logical, process, physical, development) of architecture are presented with descriptions.

Logical

This diagram shows a simplistic view of the two objects that are manipulated within the system, the user object and the portfolio object. This overview also shows how the interfaces are layered and which interface is responsible for the manipulation of the objects.

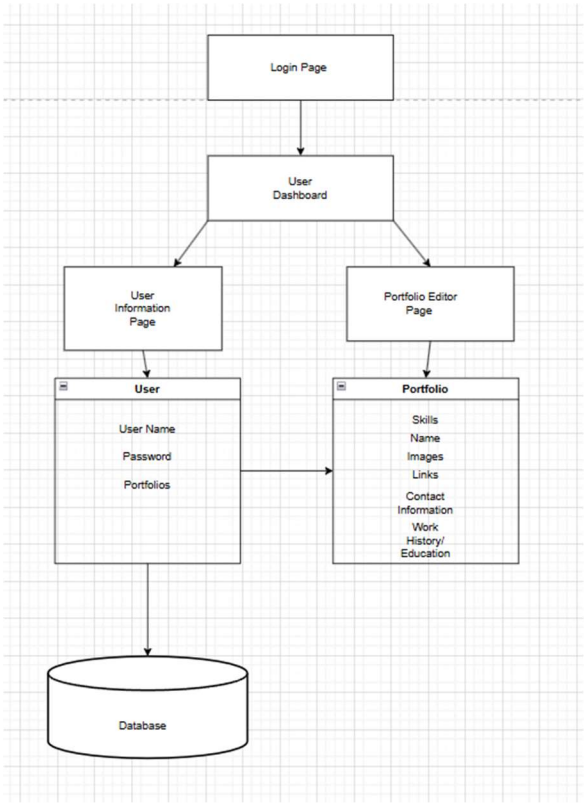


Figure 1: Functional Diagram shows the basic objects and interfaces that makeup the user requirements.

Physical:

This diagram shows how the software will be distributed mainly between the client and application devices. Our software also may include an extra component, a printer, depending on the user's needs. These devices are connected to each other through the internet and Wi-Fi connections.

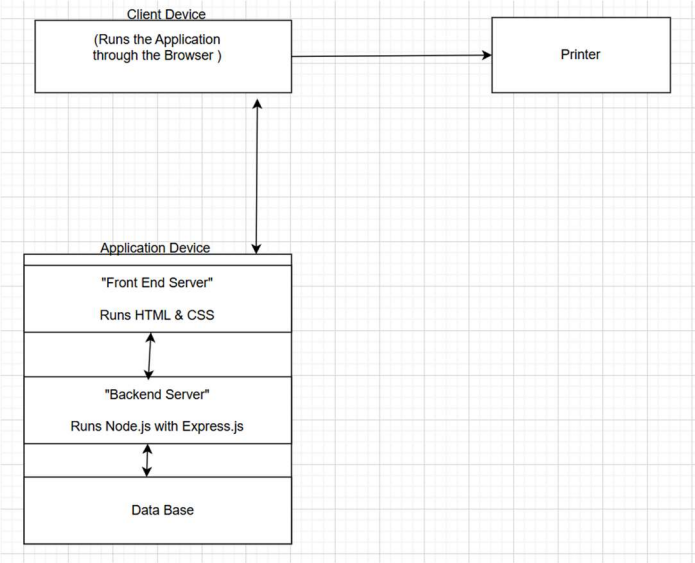
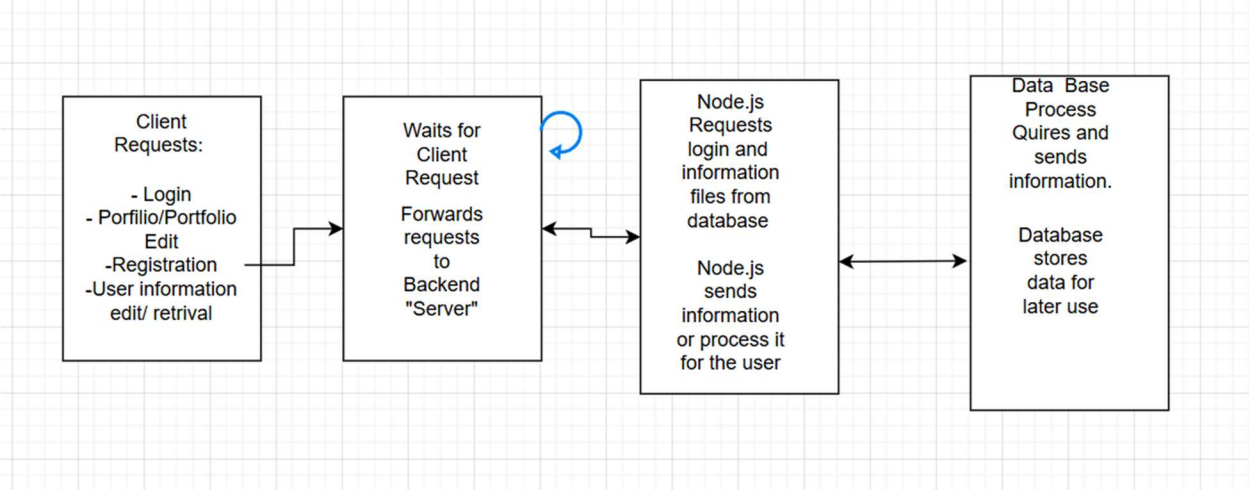


Figure 2 The Physical Diagram shows that our design only requires 2-3 hardware components; the client device, the application device, and potentially the client's printer.

Process

The process model shows the basic breakdown of the processes of each layer and their connection to one another. The 3 layers shown here is the client or "Front End" process, the Application layer (box 2 &3), and the Data Base layer.



Development

The development diagram is divided into three categories of functionality: the login page, portfolio development page, and the user dashboard page. These categories then define the types of files and functionalities that need to be developed to reach the final software specifications.

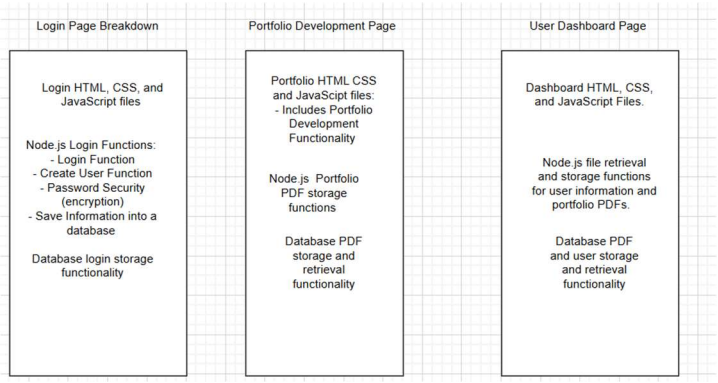


Figure 3: Development Diagram; this diagram breaks down the software development into the three main categories of functionality.

3.2 Description for Components - Kevin

A description of major software components contained within the architecture is presented. Section 3.2.1 is repeated for each of n components.

- Input text fields prompt user to enter personal information for portfolio generation.

- Color Picker prompt that allows user control over portfolio theme/design.
- Generate PDF button that converts the generated portfolio displayed on the website to a PDF file for personal use.
- Generate Portfolio Button which uses information entered by the user in the input text fields and formats this information into a professional looking portfolio.

3.2.1 Component n description

3.2.1.1 Interface description

Input, output, exceptions, etc.

Input:

Text Fields store user information for each given section into a html class and divs to group each major section.

Output:

Display generated portfolio of the content entered in the text input fields so that user may see how the portfolio looks after all changes are made in the information entering process.

Exceptions:

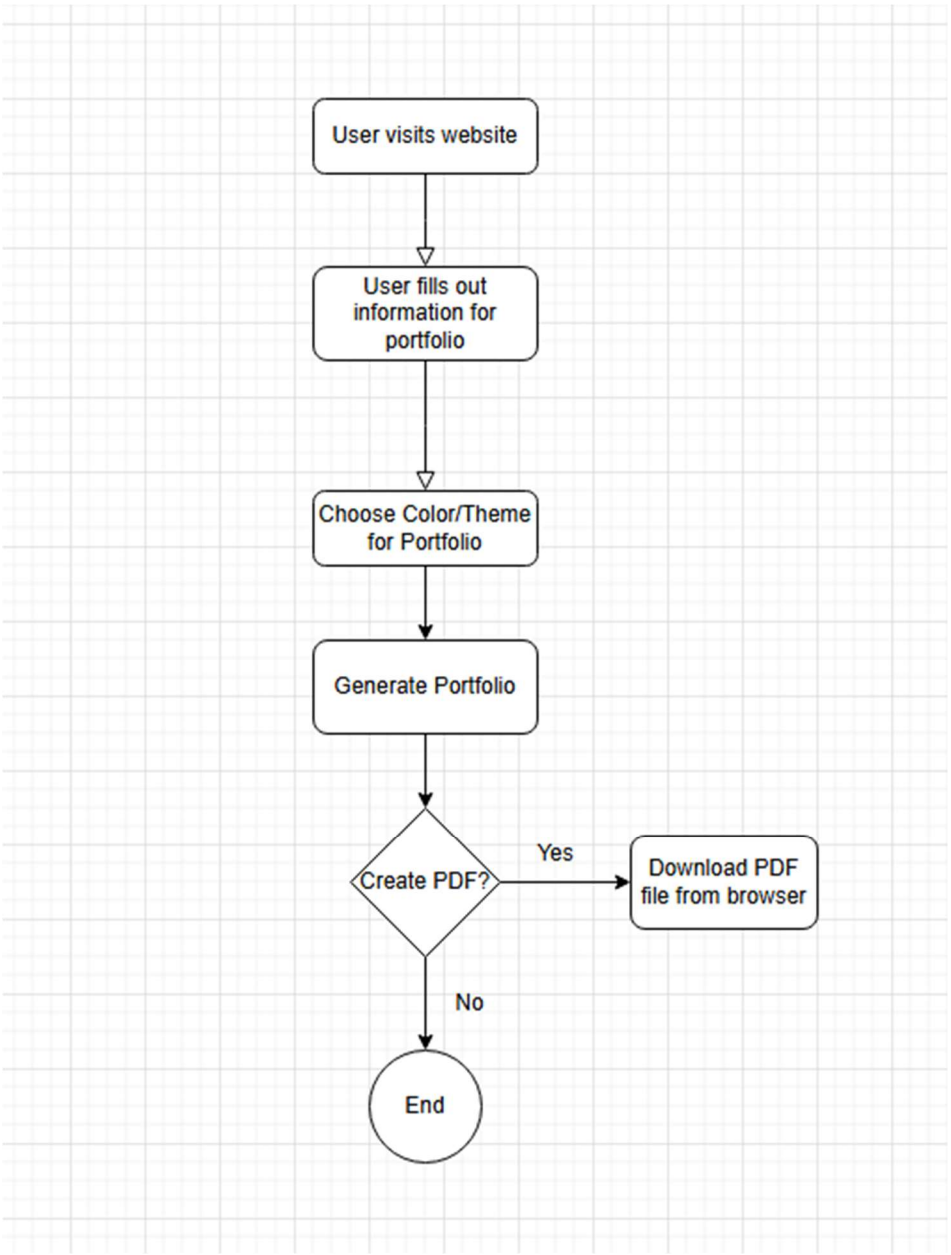
When using the color changer, it changes the entire HTML page along with the portfolio so that users do not have to scroll down all the way to see how the portfolio would look with their desired/selected colors.

3.2.3.2 Static models

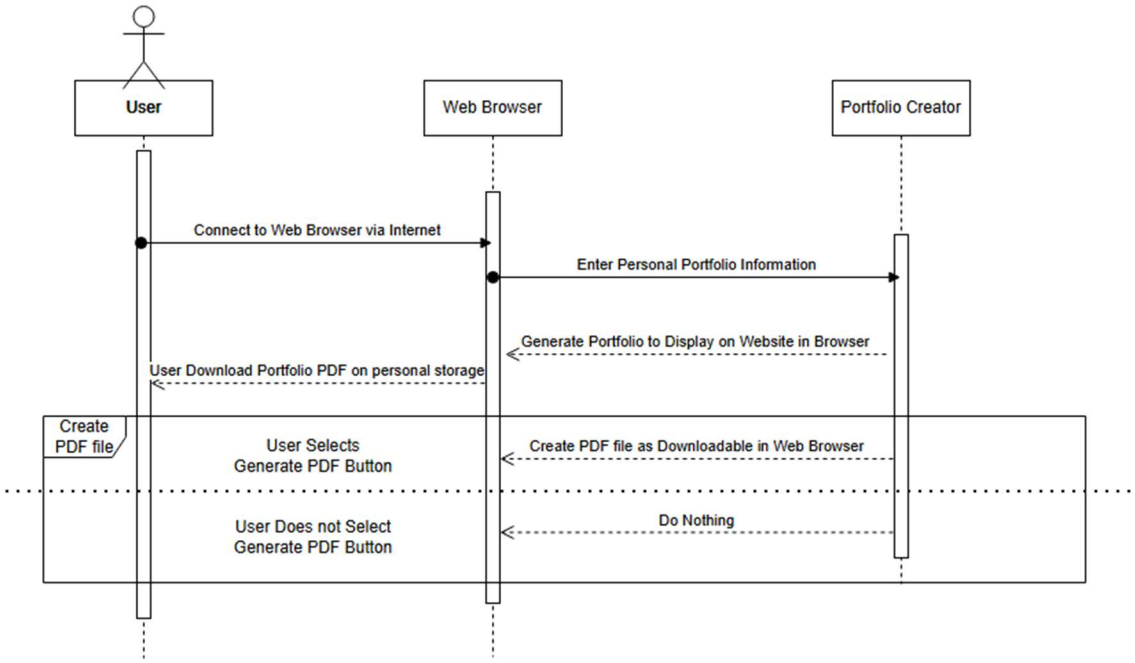
Class diagrams, composite structure diagram, etc.

Dynamic models

Activity diagram:



Sequential Diagram:



3.3 External Interface Description

- **Local Storage**: Stores all user and portfolio data using JavaScript functions/arrays.
- **PDF Generator Library**: Converts in-browser HTML/CSS content to PDF
- **Theme APIs**: CSS variables managed with JavaScript toggle themes

4.0 User interface design

A description of the user interface design of the software is presented.

4.1 Description of the user interface

There will be three main user interfaces:

- The User Dashboard:

  The user Dashboard allows the user to either edit an existing portfolio, create a new portfolio, or change user information.

- The Portfolio Builder:

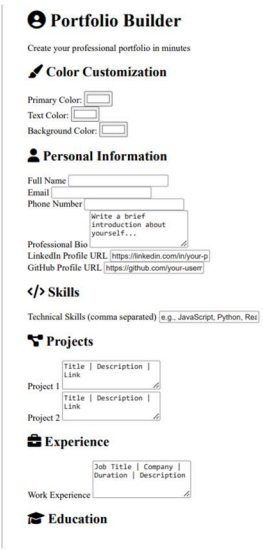


Figure 4 Portfolio builder interface prototype.

This interface allows the user to create, edit, and a portfolio object. This interface (as seen above) prompts the user to add their personal information, skills, experience, education, and project links. It also has a option to change text and background color.

- The Login Page:

  The login page is a simple standard login page that prompts the user to choose to register or login. It then prompts the user to enter a password and username or an email address if the user is registering.

4.2 Interface design rules

For our user interfaces we followed the standards set by the Interaction Design Foundation, the main standards being:

- Language and concepts found in the interface design must match those found in the real world.
- Users must be able to re-do/ un-do actions where possible.
- Terminology and graphical elements must be consistent throughout the design.
- Designs must keep potential user errors to a minimum.

5.0 Restrictions, limitations, and constraints

- Software must be implemented on one device and cannot rely on external servers or databases.

6.0 Appendices

Presents information that supplements the design specification.

6.1 Requirements traceability matrix

A matrix that traces stated components and data structures to software requirements is developed.

6.2 Implementation issues