***TEST SPECIFICATION***

***Portfolio Builder***

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

**1.0 Introduction**

This section provides an overview of the entire test document. This document describes both the test plan and the test procedure.

**1.1 Goals and objectives**

* Ensure that all form inputs accept and validate user data correctly.
* Verify that the color customization updates the interface dynamically.
* Test that the generated portfolio displays the correct structure and formatting.
* Confirm that data is correctly saved to and retrieved from localStorage.
* Verify the "Download PDF" functionality outputs an accurate representation of the portfolio.

**1.2 Statement of scope**

Included in Testing:

* Form field validation (name, email, phone, LinkedIn, GitHub)
* Dynamic UI color changes
* Skill/project/experience/education entry rendering
* Portfolio preview generation
* PDF export functionality
* Responsive behavior on desktop and mobile
* Cross-browser performance testing (Chrome and Edge)

Excluded from Testing:

* Backend integration (none exists; app is frontend-only)

**2.0 Test Plan**

This section describes the overall testing strategy and the project management issues that are required to properly execute effective tests.

**2.1 Software to be tested**

Software:

* Name: Portfolio Builder
* Version: 1.0
* Platform: HTML5, CSS3, JavaScript (Vanilla)
* Libraries: jsPDF, FontAwesome
* Hosting Platform: Netlify
* Deployment URL: [https://portfoliogeneratorg15.netlify.app](https://portfoliogeneratorg15.netlify.app/)/
* Browser Target: Google Chrome (latest)

Exclusions:

* Server-side components (none used)
* User authentication or account system

**2.3 Testing tools and environment**

* Browser: Google Chrome
* Dev Tools: Chrome Developer Console
* Simulator/Devices: Mobile view in Chrome’s responsive mode
* Libraries: jsPDF (for PDF export), FontAwesome (icons)
* Development Environment: Visual Studio Code
* Storage: localStorage for saving the form data
* OS: Windows 10/11 (test machine)
* Deployment Environment:
  + Platform: Netlify
  + Live Test URL: <https://portfoliogeneratorg15.netlify.app/>

**2.4 Test schedule**

A detailed schedule for testing is described.

**Launch Tests:**

* Desktop Test:
* Chrome Test

This test would launch the URL through a chrome browser on a desktop to ensure that it is possible to connect.

* Microsoft Edge Test

This test would launch the URL through a Microsoft Edge browser on a desktop to ensure that it is possible to connect.

* Mobile Test:
* Chrome Test

This test would launch the URL through a chrome browser on a mobile device to ensure that it is possible to connect.

* Microsoft Ege Test

This test would launch the URL through a Microsoft Edge browser on a mobile device to ensure that it is possible to connect.

**Input Tests:**

* Valid Test Cases:

These tests will follow the expected and anticipated behavior of the form. Testers will input valid entries into all required fields, such as a properly formatted email address (e.g., user@example.com), a full name, a correctly formatted phone number, and valid URLs for LinkedIn and GitHub profiles. Skills, projects, experiences, and education entries will be added as intended. The form should accept all inputs without displaying any validation errors, and the data should be saved to localStorage correctly.

* Invalid Test Cases:

These tests will intentionally use incorrect or missing inputs to trigger validation errors. Examples include leaving required fields blank, entering an invalid email (e.g., user@com), providing improperly formatted URLs, inputting nonsensical characters into the name or phone fields, or exceeding reasonable character limits. The form should block submission and display appropriate error messages, and invalid data should not be saved to localStorage.

Output Tests:

These tests focus on the dynamic generation and display of the portfolio content, customization features, and PDF export functionality. Testers will verify that the on-screen portfolio preview accurately reflects the user’s inputs (skills, experience, education, colors, etc.). Color customization changes should immediately update the visible theme without page reloads. When downloading the portfolio as a PDF, the file should match the preview content closely, maintaining correct formatting, layout, and user-specified styles. Any discrepancies or missing content in the PDF will be logged as defects.

**3.0 Test Cases**

This section enumerates a complete list of test cases for the software. A template for test cases is as follows.

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ID | TC-001 | TC-002 | TC-003 | TC-004 | TC-005 | TC-006 | TC-007 | TC-008 | TC-009 | TC-010 |
| Test Input | Enter valid data in all form fields | Leave required fields blank and submit. | Enter invalid email format (e.g., "user@com") | Change primary color setting using color picker. | Add multiple skills/projects/  education entries. | Save form inputs and refresh the page. | Generate and download portfolio as PDF. | Open site on mobile device (responsive view). | Load site on Microsoft Edge | Input extremely long strings into form fields. |
| Expected Output | Form accepts data, no validation errors shown. | Error messages displayed; submission blocked. | Error message displayed for invalid email. | Interface updates colors across site. | All entries appear correctly in the portfolio preview section. | Data persists after page reload via localStorage. | PDF matches on-screen portfolio content and formatting. | Layout adjusts properly for smaller screens; no elements are broken. | All functionality works identically to Chrome version. | Application handles input gracefully (text wraps, no overflow errors). |
| Description | Verify form field input and validation for valid entries. | Verify required field validation triggers properly. | Validate email input format checking. | Test dynamic UI color customization. | Test dynamic rendering of added user content. | Test localStorage functionality for form persistence. | Test PDF export accuracy using jsPDF. | Test responsive behavior for mobile displays. | Cross-browser compatibility check. | Test input handling for edge cases and long content. |