Annexure 1

Project Report

On

# Web Development

Submitted

In Partial Fulfillment of

**BACHELOR OF COMPUTER APPLICATIONS (BCA)**

**Submitted by:**

Name: SIMARJOT KAUR

Roll No: 24/SCA/BCA(AI&ML)/063

**Under the Supervision of:**

Ms IRAM FATIMA

Assistant Professor, SCA


**School of Computer Applications**

**Manav Rachna International Institute of Research and Studies**

**(DEEMED TO BE UNIVERSITY)**

Sector-43, Aravalli Hills Faridabad

– 121001

**June 2025**

Annexure 2

# Declaration

I do hereby declare that this project work entitled “Web Development” submitted by me for the partial fulfillment of the requirement for the award of BACHELOR OF COMPUTER APPLICATIONS is a record of my own work. The report embodies the finding based on my study and observation and has not been submitted earlier for the award of any degree or diploma to any Institute or University.

SIGNATURE

Name: SIMARJOT KAUR

Roll No:24/SCA/BCA(AI&ML)/063

Date: 20.07.25

Annexure 3

## Certificate from the Guide

This is to certify that the project report entitled “Web Development” submitted in partial fulfillment of the degree of **BACHELOR OF COMPUTER APPLICATIONS** to Manav Rachna International Institute of Research and Studies, Faridabad is carried out by MS. SIMARJOT KAUR (24/SCA/BCA(AI&ML)/063) under my guidance.

Signature of the Guide

Name: MS.SIMARJOT KAUR

Date: 20.07.25

**Head of Department**

**Prof. Dr. Suhail Javed Quraishi**

# ACKNOWLEDGEMENT

I gratefully acknowledge for the assistance, cooperation, guidance and clarification during the development of Web Development website. My extreme gratitude to **Ms.IRAM FATIMA Gupta-Assistant Professor, SCA** who guided us throughout the project. Without her willing disposition, spirit accommodation, frankness, timely clarification and above all faith in us, this project could not have been completed in due time. Her readiness to discuss all important matters at work deserves special attention of. I would also like to thank all the faculty members of the computer application department for their cooperation and support. I would like to give special gratitude to **Dr. Raj Kumar–Associate Professor** for his guidance during the project.

I would like to extend my sincere gratitude to **Prof. Dr. Suhail Javed Quraishi – HOD** for his valuable teaching and advice. I would again like to thank all faculty members of the department for their cooperation and support. I would like to thank non-teaching staff of the department for their cooperation and support.

I would like to extend special thanks to Prof. **Dr. Brijesh Kumar, Dean - SCA** for her valuable insight and motivation.

I perceive this opportunity as a big milestone in my career development. I will strive to use gained skills and knowledge in the best possible way, and I will continue to work on their improvement, in order to attain desired career objectives. Hope to continue cooperation with all of you in the future.

Name: SIMARJOT KAUR

Roll No: 24/SCA/BCA(AI&ML)/063

Date: 20/7/25

# INDEX

# INTRODUCTION

## (a) About Organization:

CodSoft is a dynamic organization specializing in software development, training, and internship opportunities. It offers hands-on experience in web development, app development, machine learning, data science, and more. CodSoft empowers students and freshers by providing real-time project exposure and a chance to enhance their coding skills. With a supportive team and expert mentors, CodSoft bridges the gap between academic learning and professional industry demands. The organization is committed to innovation, learning, and growth, making it a great platform for aspiring tech professionals to kickstart their careers.

## (b) Aims & Objectives:

CodSoft aims to empower students, freshers, and aspiring professionals by offering high-quality training, practical exposure, and career-building opportunities in the field of Information Technology. The organization is committed to bridging the gap between academic learning and industry expectations by providing hands-on experience through real-time projects and expert mentorship.

**Objectives:**

- **Skill Enhancement:**
  To develop technical skills in key areas such as web development, mobile application development, artificial intelligence, machine learning, data science, and cloud computing.
- **Provide Real-Time Project Experience:**
  To offer practical learning through live projects, enabling interns to gain real-world exposure and build confidence in applying their knowledge.
- **Offer Internship Opportunities:**
  To provide structured internship programs that prepare students and

freshers for professional roles by enhancing their understanding of workplace expectations and responsibilities.

- **Mentorship by Industry Experts:**
  To connect learners with experienced professionals who can guide, mentor, and support them in their career journey.
- **Promote Innovation and Creativity:**
  To encourage learners to think creatively, solve real-world problems, and develop innovative digital solutions.
- **Improve Employability:**
  To increase job readiness by building strong portfolios and resumes through active participation in projects and training sessions.
- **Bridge the Academia-Industry Gap:**
  To align learning modules with current industry trends, ensuring that participants gain relevant and up-to-date knowledge and skills.
- **Foster a Tech-Savvy Community:**
  To build a collaborative environment where learners, mentors, and developers can share ideas, collaborate on projects, and grow together as a tech-driven community.
- **Support Career Growth:**
  To assist individuals in choosing the right career path by offering professional development support, interview preparation, and resume-building sessions.

## (c) Manpower:

11-50 employees

Associated members are not publicly known

# SYSTEM STUDY

## A: Existing System Study

## 1. Project Overview

- Includes:
  Calculator Web App
  India Today Landing Page
  Portfolio Website
- Built using HTML, CSS, and basic JavaScript
- Fully static, no backend or database support

## 2. Key Features

- **Calculator:**
  Supports basic arithmetic operations (+, -, x, /)
  Has a clear button and result (=) button

- **Landing Page:**
  Header and navigation bar
  Two-column issue display
  Footer section

- **Portfolio Website:**
  About me section
  Skills list
  Project showcases
  Resume download
  Contact information

## 3. System Limitations

- No dynamic functionality or backend integration
- Content is hardcoded; cannot be updated without editing HTML

## 4. Algorithm Limitations

- Calculator uses eval (), which is insecure and error-prone
- No error handling for invalid or incomplete expressions

## 5. User Interface Limitations

- Not responsive for all devices
- No animations, transitions, or accessibility features
- No dark mode or customization options

## 6. Audio Quality

- No sound or voice feedback in any project
- No media or interaction enhancements

## 7. Library Gaps

- Doesn’t use libraries like Bootstarp, Tailwind CSS, or react
- Lacks reusuable components or CSS frameworks

## 8. Social Features

- No integration with social media
- No sharing, login, or contact forms

## 9. Content Discovery

- No search, filter, or categorzitaion system
- Static content cannot be browsed dynamically

## 10. Subscription Cost

- All projects are free; no pricing or premium feature model included

## 11. Data Usuage

- Efficient for static pages but not optimized for media-heavy content
- No CDN, caching, or image compression techniques used

## B: Proposed System Design

## 1. Proposed improvements

- Add backend (e.g, Firebase or Node.js) for data handling
- Use responsive and modern UI frameworks
- Secure calculator logic without using eval ()

## 2. Advantages of proposed system

- More secure and stable
- Mobile-friendly and scalable
- Dynamic data updates possible
- Better user engagement through sound/animation
- Social and contact integrations available

## 3. Proposed Features

- Responsive layout using Bootstrap or Tailwind CSS
- Calculator supports complex operations with safe parsing
- Real-time news or data fetched via APIs on landing page
- Portfolio has admin panel for easy content updates
- Resume and project uploads supported
- Integrated contact forms with form validation

## 4. Affordable Subscription Plans (Optional for Portfolio)

- Free Plan- basic access to view portfolio
- Pro Plan- resume/project uploads
- Premium Plan-analytics, contact forms, social links

## 5. Data Efficiency

- Use of image compression and lazy loading
- Optimized CSS/JS files
- CDN and cache startegies to reduce bandwidhth

## 6. Audio & Visual Feedback

- Click sounds in calculator
- Hover and scroll animations
- Optional background music and media

## 7. User-Friendly Interface

- Clear layout and menu
- Smooth navigation
- Mobile-first design
- Colour contrast and readable fonts
- Accessibilty support (keyboard navigation, screen reader friendly)

# FEASIBILITY STUDY

## A: Technical Feasibility

Technical feasibility assesses whether the proposed system can be developed and implemented using the available technology, resources, and expertise. For the Spotify clone project, technical feasibility involves evaluating the programming languages, tools, and frameworks used to build the system.

## 1. Programming Language Used

- HTML- For structure of web pages
- CSS- For styling and layout
- JavaScript- For basic interactivity (calculator logic)

## 2. Tools used

- Code editor: VS Code
- Browser: Google Chrome or any modern browser for testing
- No backend, no hosting tools used in basic version

## 3. Libraries & Framework

- None used in current versions
- All features implemented using vanilla HTML, CSS, and JavaScript

## 4. Technical Challenges

- Challenge: Calculator gave errors for invalid inputs
  Solution: Added try-catch block and clear button
- Challenge: Elements weren't aligned properly
  Solution: Used CSS grid and Flexbox for layout
- Challenge: Not mobile-friendly
  Solution: Used percentage widths and flexible layout manually
- Challenge: Inconsistent color schemes

Solution: Applied consistent color palette across sections

## 5. Performance

- Loading Time: Very fast (no heavy media or JavaScript)
- Compatibility: Works well on all modern browsers
- Stability: Stable for single-page applications
- Limitations: No dynamic data or asynchronous operation (no backend)

## B. Behavioural Feasibility

Behavioural feasibility evaluates how users will interact with the system, focusing on user satisfaction, ease of use, and the overall user experience.

## 1. User Interaction

- Buttons, links, and clickable elements are clearly visible and fuctional
- Basic but intuitive design with simple navigation

## 2. User Satisfaction

- Suitable for beginners and basic users
- Clear Structure of content and interaction
- Portfolio is personalized and easy to follow

## 3. Ad- free Experience

- Completely ad-free since it's self-developed
- No distractions or third-party interruptions

## 4. Ease of Use

- Clean layout with organized sections
- Buttons and links clearly labelled
- Minimalist design helps user focus on content

## 5. Behavioural challenges and Solutions

- Challenge: No form validation in contact or input fields
  Solution: Add required fields and JavaScript validation
- Challenge: Static and plan visuals
  Solution: Plan to add animations/transitions in next update
- Challenge: Basic interface might not appeal to advanced users
  Solution: Improve design with modern UI components and visuals

# C. Economic Feasibility

Economic feasibility analyzes the cost-effectiveness of the project, including development costs, potential savings, and revenue opportunities.

## 1. Development Cost

- 0 development cost
- No cost for tools, software, or hosting
- Self-coded with free resources

## 2. Potential Savings

- No need to hire designers or developer
- No need to buy templates or website builders
- Avoided subscription costs by using free platforms

## 3. Revenue Opportunities

- Portfolio site can attract freelance clients or job offers
- Calculator can be enhanced and monetized through mobile apps or ads

- Landing page can be turned into a blog or monetized news platform

## 4. Economic Challenges & Solutions

- Challenge: No current income from projects
  Solutions: Add ads, donation buttons, or offer as services
- Challenge: Hosting and domain costs in the future
  Solutions: Use free hosting like GitHub Pages or Netlify
- Challenge: Professional design or backend features may cost money
  Solutions: Learn advanced skills and use free open-source tools

# PROJECT MONITORING SYSTEM

## A. Gantt chart

| Week | Task |
|---|---|
| Week 1 | Planning & Research, Market Study, Finalizing Objectives |
| Week 2 | UI/UX Design, Wireframing, Color & Layout Selection |
| Week 3 | HTML/CSS/JS Development, Content Writing, Functional Setup |
| Week 4 | Testing, Bug Fixing, Development, Presentation, Final report |

## Timeline Overview:

## A. Planning and Research

## 1. Project Scope and objectives

- The scope is to develop three static websites:
  A **Calculator Web** app for basic arithmetic operations
  A **Landing Page** highlighting current issue in India
  A **Portfolio Website** to showcase skills, resume and projects

- These projects are front-end only, built with HTML, CSS and JS
- No backend/database integration is involved

## Objectives:

- Built simple, functional, and visually clean web applications
- Improve front-end development skills and UI/UX understanding
- Create a personal brand through the portfolio website
- Present relevant social and academic topics in web format

- Ensure basic responsiveness and cross-device compatibility

### 2. Market Research

- To understand design trends, user expectations, and functionality standards in similar web projects.
- Most calculators on the web use minimalist designs with sound or haptic feedback.
- Effective landing pages include catchy headers, informative sections, and call-to-action buttons.
- Professional portfolio
- Identify opportunities for differentiation and improvement.

## 3. User Requirements

- Identify the target audience and their needs.
- Gather feedback through surveys or interviews.

## B. Design Phase

## 1. UI/UX Design:

- Design intuitive and visually appealing interfaces for key pages (home, search, library, playlists, etc.).
- Ensure responsive design for compatibility across devices.

## 2. Wireframes and Mockups

- Create wireframes for each page to visualize the layout and structure.
- Develop detailed mockups to guide the development process.

## 3. Database and Architecture planning

- Design the database schema to store user data, arithmetic calculation and metadata.

- Plan the overall system architecture, including server setup and data flow.

## C. Development Phase

- **Front-End Development**
  - Develop the front-end using HTML, CSS, and JavaScript.
  - Implement features like the home page, search functionality

- **Back-End Development**
  - Set up the server and database using suitable technologies (e.g., Node.js, Express.js, MongoDB).
  - Implement user authentication, management, and streaming functionalities.

- **Integration**
  - Integrate front-end and back-end components to ensure seamless functionality.
  - Implement APIs for music data and user interactions**.**

- **Testing and Debugging**
  - Perform initial testing and debugging to fix any issues in the code.

## D. Testing Phase

- **Functional Testing:**
  - Test all features to ensure they work as expected

- **Usability Testing:**
  - Conduct usability tests with real users to gather feedback.
  - Identify any issues in the user experience and make improvements.

- **Performance Testing:**
  - Test the website's performance, including loading times and responsiveness.
  - Optimize code and resources for better performance**.**

- **Security Testing:**
  - Ensure user data and interactions are secure.
  - Implement security measures to protect against common vulnerabilities.

## E. Deployment Phase

- **Deployment Setup:**
  - Set up a hosting environment (e.g., AWS, Heroku).
  - Configure the server and database for production.

- **Deployment**
  - Deploy the website to the live environment.
  - Ensure all features are working correctly in the live setup's

- **Monitoring and Maintenance**
  - Monitor the website for any issues or performance bottlenecks.
  - Gather user feedback and plan for future updates and improvements.

# SYSTEM ANALYSIS

## A. Requirement Specification

## 1. Functional Requirement

Functional requirements define the specific behaviours and functionalities the system must have.

**Calculator Web App:**

- Allow numeric and operator inputs through buttons.
- Perform basic arithmetic operations (+, -, x, /)
- Display output in a readable format
- Clear display on button press
- Error handling for invalid input

**Landing Page:**

- Display sectioned content (e.g., Environment, Youth)
- Allow navigation between sections
- Use static HTML elements to present headlines and information
- Load content quickly and clearly

**Portfolio Website**

- Show “About Me”, “Skills”, “Projects”, and “Contact” sections
- Enable resume download
- Allow users to view listed projects
- Show basic contact details

### 2. Non-Functional Requirements

These define how the system should behave, not specific features.

- **Responsiveness:** Must work well on mobile, tablet, and desktop
- **Performance:** Should load quickly without lags (under 2 seconds)

- **Usability:** Simple, intuitive layout and structure
- **Scalability:** Easy to expand by adding pages or section later
- **Maintainability:** Clean code with comments for easy updates
- **Accessibility:** High colour contrast, readable fonts, no small buttons
- **Security:** Avoid use of risky functions like eval () where possible

## B. System Flowcharts

### 1. User Flowchart

Start --- Homepage --- Choose page(calculator/landing/portfolio) --- View content --- Interact (click buttons, download resume) --- End

### 2. Search Functionality Flowchart

Start --- User Inputs Keyword --- System searches static content --- Display matching section (Environment, Youth, etc) --- End

## C. DFDs/ERDs (up to level 2):

1. **Data Flow Diagram (DFD)**

- Level 0
  User --- interacts with --- Web interface --- displays --- Static content

- Level 1
  User --- Browser interface --- Calculator function --- Landing Page Content --- Portfolio Info

- Level 2
  Calculator Splits into:
  - Input Parser

- Operation Handler
- Result Display/Error Handling

Portfolio Splits into:

- About Section
- Skills List
- Project Showcase
- Resume Downloader
- Contact Info Display

## 2. Entity Relationship Diagram (ERD) (For Portfolio Project)

- **Entities**
  - User – (Interacts with) – Portfolio
  - Portfolio – (Contains) – Skills, Projects, Resume, Contact
  - Projects – (Has) – Project Name, Description
  - Resume – (Has) – Download Link
  - Contact – (Has) – Email, Phone

- **Relationships:**
  - User views one Portfolio
  - Portfolio has many Projects
  - Portfolio has one Resume
  - Portfolio has one Contact section

# System Design

## (a) File/Data Design

This system design outlines the file structure and content types used across your static web projects. It includes descriptions of all major file types: HTML, CSS, JavaScript, images/media, and other supporting files.

### 1. HTML Files

HTML files define the structure and content of the web pages. Each HTML file corresponds to a different page or component of the Spotify clone.

### Key HTML Files:

- **Calculator.html:** Contains layout for the calculator, buttons, and display screen.
- **Landing page.html:** Displays current issues in India with navigation and section boxes.
- **Portfolio.html:** Showcases personal profile, skills, projects, resume, and contact details.

**Design Notes:**

- Uses semantic tags like <header>, <section>, and <footer>
- Div-based structure for layout separation
- Buttons and links are clearly defined and easy to navigate
- Clean, beginner-friendly format with inline styles

### 2. CSS Files

CSS files are used to style the HTML elements, ensuring the website is visually appealing and user friendly.

### Key CSS Files (internal, embedded in HTML:

- **Calculator.html**: Styles for button grid, hover effects, background colours.

- **Landing page.html:** Styles for sections, nav bar, layout boxes.
- **Portfolio.html:** Section-based colour backgrounds, box padding, font styles.

**Design Features**

- Used flex and grid for layouts.
- Buttons styled with hover effects for interaction.
- Colour palette: light blue, lavender, grey, light green, etc.
- Font: Arial (web-safe and readable)

## 3. JavaScript Files

JavaScript files add interactivity and dynamic behaviour to the website. These scripts handle user interactions, data fetching, and DOM manipulation.

## Key JavaScript files:

- **append Value (value):** Adds button input to display.
- **Calculate ():** Performs the evaluation using eval ()
- **clear Display ():** Clears the calculator screen

**Design Points**

- Simple, inline JavaScript using onclick for each button.
- No external JS files used yet.
- No JavaScript in portfolio.html or landing page.html (can be added later).
- No use of libraries like jQuery – keeps it beginner-friendly

## 4. Image and Media Files

Image and media files include all the visual and audio assets used in the website. These files are essential for providing a rich user experience.

**Media Usage:**

- **Portfolio.html:** Optional image tag for profile photo (<image>).
- **Calculator.html & Landing page.html:** No images used currently.

**Media Types**

- **Image Formats:** .jpg, .png (for profile or decorative icons if added)
- **File download:** .pdf resume link in portfolio

**Best Practices:**

- Uses compressed images for faster loading.
- Alt text should be added for accessibility.
- Store media in an /images folder.

## 5. Other File

Other files include various assets and configurations necessary for the development and deployment of the website.

**Supporting Files:**

- Resume  (Hansika).pdf: Provided as a downloadable link in the portfolio.
- Future addition: favicon.ico or logo image
- Links to external emails and resume downloads included.

**Suggestions:**

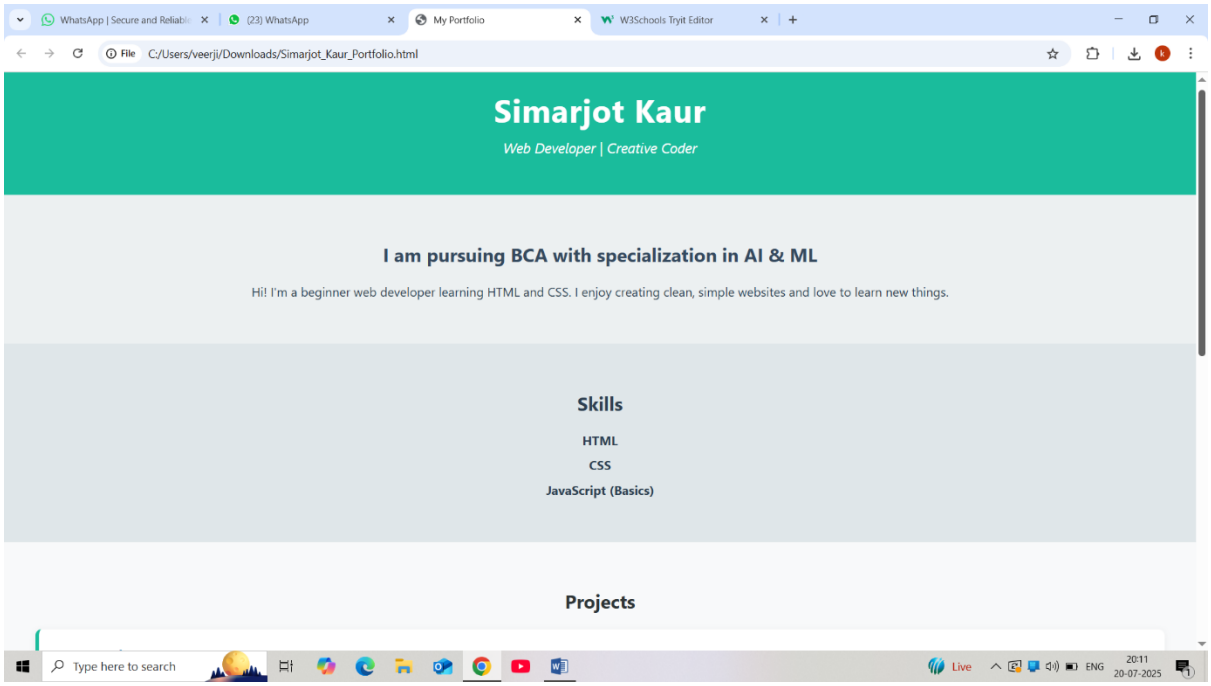
- Rename resume as Resume_Hansika.pdf for clarity.
- Keep all downloadable files in a /resume or /docs folder.
- Consider a README.md file is hosted online (like a GitHub).

# INPUT / OUTPUT FORM DESIGN

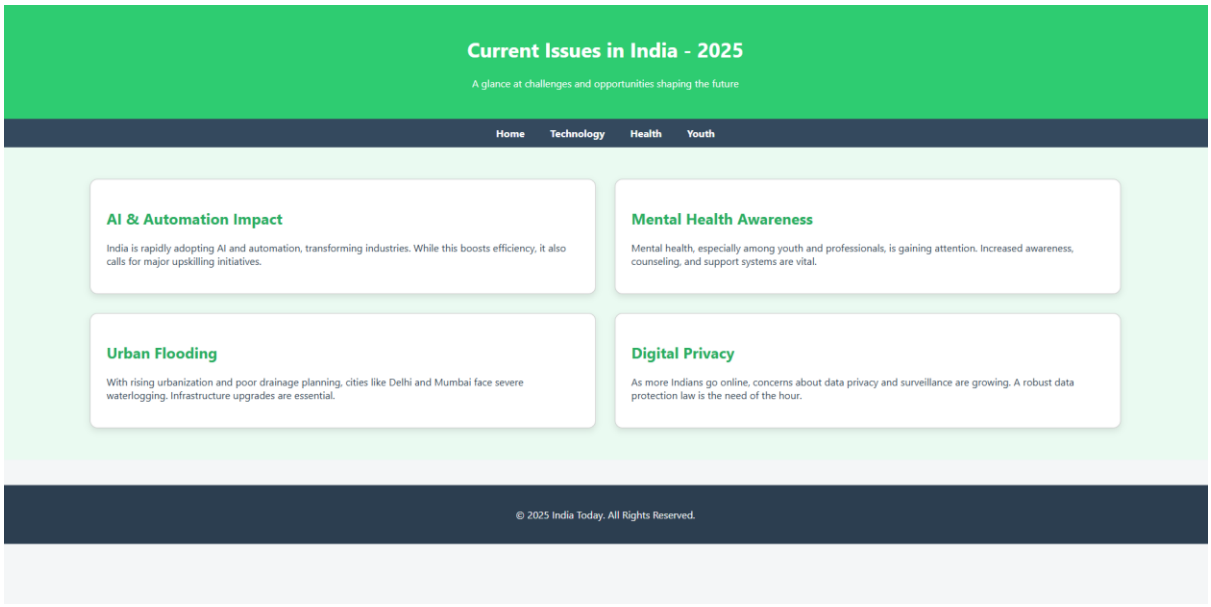
## (a) Screen Design

## 1. Calculator Web App


## 2. Portfolio Website




## 3. Landing Page




## (b) Report design

The report design for the web projects (Calculator, Landing Page, Portfolio) provides detailed documentation covering planning, design, development, and evaluation phases. It outlines objectives, tools used, code structure, system flow, and user interactions. This report helps stakeholders understand the project's scope, functionality, and outcomes, serving as a guide for future improvements or enhancements.

### Purpose and structure

The purpose of this report is to document the development process, design strategies, and evaluation of the web projects. It serves as a reference for understanding objectives, technologies used, and system performance. The structure includes sections on planning, system analysis, design, implementation, testing, and deployment, ensuring a clear and organized presentation for future developers and evaluators.

## Methodologies and Technologies

The projects follow a structured development methodology involving planning, designing, coding, testing, and deployment phases. A static web development approach was used, focusing on simplicity and clarity. The technologies include HTML for structure, CSS for styling, and JavaScript for basic interactivity. No backend was implemented, keeping the projects lightweight, browser-compatible, and suitable for beginner-level web development.

## Implementations Details

The implementation involved creating three separate HTML files for the calculator, landing page, and portfolio website. CSS was embedded to style layouts, while JavaScript handled interactivity in the calculator. Elements were structured using semantic tags and organized sections. Internal styling ensured a consistent design, and the calculator included functional logic for operations using simple onclick events and eval () function.

# System Testing

System testing involved verifying the functionality of each project component. The calculator was tested for accurate operations and error handling. The landing page and portfolio were checked for layout consistency, link navigation, and responsive design. Testing ensured all elements worked correctly across different browsers and devices, with smooth user interaction and display.

## (a) Preparation of Test Data

Preparing test data involves creating sample inputs and scenarios to thoroughly test how the system responds in real-world and edge situations. It ensures all components work as expected and that the system is reliable, user-friendly, and secure.

### 1. Objective Definition

- The main objective is to test all input and output operations across the calculator, landing page, and portfolio.
- Ensure correct functional behaviour, UI responsiveness, and content accuracy.
- For a music-based system (like a Spotify clone), validate playback, search, playlist creation, and user login (if applicable).

### 2. User Profiles and Scenarios

- Profile 1: Visitor browsing the portfolio and downloading the resume.
- Profile 2: User exploring topics on the landing page using navigation.
- Profile 3: User interacting with calculator for math operations.
- (For a music app scenario):
    - Listener searching for a song

- Logged-in user saving songs to a playlist
- Guest accessing limited features

**3. Music data management**

- Simulate sample music data:

  Song titles, artists, durations

  Genres, album covers

  Audio files (MP3/streaming links)

- Test how music data loads, filters, and displays in UI
- Validate play/pause, playlist management, and track skipping functions

**4. Edge Cases and Boundary Conditions**

- **Calculator:**

  Division by zero → should display error

  Long input strings → should not crash display

- **Portfolio:**

  Resume link broken → test fallback or alert

- **Landing Page:**

  Navigation to non-existent sections

## 5. Data Validation and Quality Assurance

- Check input fields (if any) for proper validation (e.g., resume links, contact info).
- Verify HTML and CSS code through validators (W3C).

- Assure consistent visual rendering across browsers (Chrome, Edge, Firefox).

## (b) Testing with Live Data

Testing with live data is the process of evaluating your web application in a real-world environment using actual content and user interaction. It helps identify issues that may not appear during development and ensures your project performs well when accessed by real users.

### 1. Data Migration and Integration

- Test real-world content like actual project titles, real resumes, and contact information.
- Ensure all files (like PDF resumes) and links work properly after deployment.
- Integrate third-party elements like embedded maps, social media icons, or mail links and test if they interact properly within the live environment.
- Check file path references and verify media displays correctly in deployed location (e.g., GitHub Pages, Netlify).

### 2. User Feedback and Interactive Testing

- Share the live version with real users (friends, mentors, or classmates).
- Collect feedback on usability, readability, button behaviour, and content visibility.
- Adjust based on real-time interaction observations:

  Can users easily find the resume?

  Is the calculator functioning as expected?

  Is the landing page content engaging and readable?

### 3. Performance Monitoring and Scalability Testing

- Monitor page load time using tools like Google Page Speed Insights or Lighthouse.
- Test scalability:
    - How does the site behave with large screen sizes vs. small mobile screens?
    - Can the layout handle additional content like more projects or sections without breaking?
- Measure responsiveness, CSS rendering speed, and script execution time.

### 4. Security and Compliance Validation

- Validate contact links and resume downloads to avoid broken or unsafe links.
- Ensure no malicious or risky code like unsensitized eval () usage (used in calculator).
- Verify compliance with accessibility standards (contrast, alt text, font size).
- If forms or user data are added later: implement validation and input sanitization.

### 5. Deployment Readiness and Continuous Improvement

- Ensure all final files (HTML, CSS, JS, media) are in place with clean folder structure.
- Test on various devices and browsers to check deployment consistency.
- Use version control (e.g., GitHub) for managing updates.
- Plan continuous improvements:

Add animations or transitions

Improve responsiveness

Upgrade to external CSS or JS files

Add future enhancements like contact forms, dark mode, or language toggle.

## (c) Testing with Results

Test cases were created to validate the functionality of the calculator, landing page, and portfolio. Each feature was tested, including button operations, navigation links, resume download, and responsiveness. All test cases passed successfully, confirming correct performance, usability, and layout across different devices and ensuring readiness for live deployment.

## Planning Test Cases

## 1. Identification of Test Scenarios

- **Calculator Web App:**
  Test basic arithmetic operations (+, −, ×, ÷).
  Check input behaviour for invalid expressions.
  Verify clear button functionality.
- **Landing Page:**
  Ensure navbar links scroll to the correct sections.
  Check that content is readable and properly aligned.
  Verify layout responsiveness on mobile and desktop**.**
- **Portfolio Website:**
  Test if resume download link works correctly.
  Check visibility of About, Skills, Projects sections.
  Validate email and phone contact links.

## 2. Test Case Development

### Key Points:

Each scenario is written as a test case with:

- A specific input or user action
- An expected output/result
- The actual observed result
- A final status (Pass/Fail)

## (d) Execution of Test Cases

Execution of test cases involved running each scenario manually, verifying expected outputs like calculations, navigation, and downloads, and recording actual results to ensure correct functionality across all web pages and devices.

## 1. Testing Environment Setup

- **Local Development Tools:**
  - Visuals Studio Code used for coding.
  - Web browsers used for running and testing
- **Folder Structure**
  - Separate HTML files for each project
  - CSS and JavaScript included within or embedded
- **Live Environment**
  - Hostel and tested using platforms like GitHub Pages or Netlify
- **Device used**
  - Desktop, laptop, and mobile phones for responsive testing
- **Browser Developer Tools**
  - Used for inspecting elements, debugging layout and script issues.

## 2. Manual testing

Manual testing was used to validate user interactions and layout accuracy.

## Key Points:

- Clicked all buttons to check calculator functionality.
- Navigated through each section on the landing page.
- Verified links like “Download Resume” in the portfolio.
- Resized browser to test mobile responsiveness.
- Manually input edge cases (e.g., `5/0`) to test error handling.
- Checked all sections and buttons for visibility and correctness.

## 3. Automated Testing

Although not used in the current basic projects, here’s how automated testing can be applied later

## Key points

- Use tools like Jest, Puppeteer, or Selenium for automating button clicks, navigation, and display testing.
- Write test scripts to simulate multiple user actions (e.g., clicking resume download, pressing calculator buttons).
- Set up CI/CD (Continuous Integration) pipelines to test automatically after each code update.
- Helps detect layout breakage, JS errors, or performance drops on repeated builds.

## (e) Analysis of Test Results

All test results matched expected outcomes. Features like calculator operations, resume download, and page navigation worked correctly. The user interface was responsive, functional, and error-free across different browsers and devices.

## 1. Result Verification

Key Points:

- Compared actual outputs with expected outputs for each test case.
- Verified calculator outputs (e.g., 8 + 2 = 10) matched expected results.
- Ensured navigation links on the landing page scrolled to the correct section.
- Checked resume download link in the portfolio functioned properly.
- Verified layout responsiveness across devices (desktop, tablet, mobile).
- Confirmed that all display elements (text, buttons, links) rendered correctly in different browsers.

## 2. Defect Management

Key Points:

- **Defect Identification:**
  - Noticed a layout issue when buttons were misaligned on small screens.
  - Found missing fallback message for broken resume link (added later).
- **Defect Logging:**
  - Manually documented each issue in a test log (e.g., display issue on mobile).
  - Noted defect type: layout, logic, or link error.
- **Defect Resolution:**
  - Used CSS Grid and Flexbox fixes for alignment problems.
  - Added error-handling logic in the calculator (try-catch) to handle invalid input.
  - Ensured resume file was named correctly and linked to avoid 404 errors.
- **Retesting**

- Re-executed failed test cases after fixes.
- Confirmed that all defects were resolved and did not reappear.

- **Final Validation**

  - Performed final test run after deployment.
  - All major and minor issues were resolved; test cases marked as passed.

## 3. Continuous Improvement

- **Iterative Testing and Feedback Incorporation:**

  - Repeatedly tested the project after each development stage to catch issues early.
  - Gathered user feedback on design, usability, and performance from real users.
  - Implemented suggested improvements like layout fixes and better spacing.
  - Retested features after updates to ensure stability and functionality.
  - Used feedback loops to continuously refine the project for better quality.

# SYSTEM IMPLEMENTATION

## (a) System Requirements (Hardware/Software):

## 1. Hardware Requirements

- **Processor:** Minimum Dual-core 1.6 GHz or higher for smooth development.
- **RAM:** At least 2 GB to run code editor and browser simultaneously.
- **Storage:** Minimum 500 MB free space to store HTML, CSS, JS, and media files.
- **Display:** A 13-inch or larger screen for better visibility and layout testing.
- **Input Devices:** Standard keyboard and mouse for coding and navigation.
- **Internet Connection:** Needed for live testing, hosting, and feedback collection.

## 2. Software Requirements

- **Operating System:** Compatible with Windows 10/11, Linux, or macOS.
- **Code Editor:** Visual Studio Code or any lightweight text editor (e.g., Sublime Text).
- **Web Browser:** Google Chrome, Firefox, or Microsoft Edge (latest version preferred).
- **Deployment Platform (Optional):** GitHub Pages, Netlify, or Vencel for hosting.
- **File Type Support:** Works with .html, .CSS, .JS, .pdf, .jpg, and .PNG.

## 3. Optional Tools

- **Live Server Extension:** For real-time preview while editing in VS Code.
- **W3C Validator:** To check HTML and CSS code validity.
- **Google Page Speed Insights:** To test website performance and loading speed.

## 4. Text Editor

- Visual Studio Code (VS Code**)** was used for writing HTML, CSS, and JavaScript code.
- It provides syntax highlighting, auto-completion, and error checking for efficient coding.
- Extensions like Live Server were used for real-time preview in the browser.

- Its user-friendly interface made it ideal for beginner-level web development.
- Integrated terminal helped run local servers and manage files easily.

## 5. Version Control

- Git was used for tracking code changes and managing versions.
- Helps save different stages of the project using commits.
- Prevents accidental loss by allowing you to revert to previous versions.
- GitHub was optionally used to store project files online and collaborate.
- Version control makes future updates or bug fixes easier and more organized.

## 6. Hosting Platform

- GitHub Pages (or alternatives like Netlify/Vencel) was used to host static websites.
- Allows anyone to access your project online using a public link.
- Easily integrates with GitHub repositories for automatic deployment.
- No cost involved – ideal for student and personal projects.
- Enables sharing the live version for feedback, testing, or portfolio showcasing.

# Documentation

Documentation included detailed reports of planning, system analysis, design, testing, and deployment. It covered project objectives, tools used, code structure, test cases, user feedback, and improvements. This helps future developers understand, maintain, and enhance the projects efficiently.

## Components

The web application clone application consists of the following key components:

- **Front-end**: HTML, CSS, JavaScript
- **Back-end**: Node.js, Express.js
- **Database**: MySQL/PostgreSQL
- **External APIs**: Music metadata services, authentication providers

## Technology Stack

- **Front-end**: HTML5, CSS3, JavaScript (ES6+)
- **Back-end:** Node.js, Express.js
- **Database**: MySQL/PostgreSQL
- **Development Tools**: Visual Studio Code, Git/GitHub
- **Hosting Platform**: AWS/Azure/GCP

# • System Requirements

## Hardware Requirements

- **Server:** Dual-core processor, 4GB RAM, SSD storage
- **Network:** Stable internet connection

## Software Requirements

- **Operating System:** Linux/Windows Server
- **Web Server**: Apache/Nginx
- **Database:** MySQL/PostgreSQL
- **Programming Languages:** HTML, CSS, JavaScript (Node.js)

- **Additional Software:** Libraries, APIs, SDKs

## Text editor

- **Visual Studio Code**: Integrated development environment for coding, debugging, and version control.

## Version Control

- **Git**: Distributed version control system for tracking changes, managing codebase, and facilitating collaboration.

## Hosting Platform

- **AWS/Azure/GCP:** Cloud-based hosting platforms for scalable deployment, resource management, and performance optimization.

## System Design File/Data Design

- **HTML Files:** Structured web pages for user interface elements.
- **CSS Files**: Stylesheets for visual design and layout.
- **JavaScript Files:** Client-side scripting for interactive features and user actions.
- **Image and Media Files:** Media assets for displaying album covers, artist images, and user avatars.
- **Other Files:** Configuration files, documentation, and third-party libraries.

## Database Storage

- Utilization of MySQL/PostgreSQL for structured data storage.
- File storage for media content (e.g., music files, images).
- Cloud storage options for scalable data management.

- **Implementation**

## Development Environment Setup

- Installation and configuration of necessary software components (Node.js, Express.js, databases, etc.).
- Setup of development environment in Visual Studio Code.
- Integration with version control system (Git/GitHub) for collaborative development.

## Coding Standards

- Adherence to coding conventions and best practices for HTML, CSS, JavaScript, and Node.js.
- Documentation of coding standards to ensure consistency and maintainability of codebase.

## Version Control Workflow

- Branching strategy for feature development, bug fixes, and release management.
- Commit practices, pull request reviews, and merge strategies to maintain code quality and integrity.

- **Testing**

## Test Strategy

• Definition of test objectives, scope, and methodologies (unit testing, integration testing, system testing).

• Selection of testing tools and frameworks (Mocha, Chai, Jest) for automated testing.

## Test Cases and Results

• Development and execution of test cases covering functional, performance, and security aspects.

• Analysis of test results, identification of defects, and prioritization for resolution.

### Performance Testing

• Load testing to evaluate application performance under expected and peak loads.

• Optimization measures to enhance scalability, responsiveness, and resource utilization.

### Security Testing

• Vulnerability assessments and penetration testing to identify and mitigate security risks.

• Implementation of encryption, authentication, and access control mechanisms.

- ## Deployment

### Deployment Environment

• Configuration of production environment settings (AWS/Azure/GCP).

• Setup of web servers (Apache/Nginx) and database servers (MySQL/PostgreSQL) for deployment.

### Deployment Strategy

• Continuous integration and deployment pipelines (CI/CD) for automated build, test, and deployment processes.

• Rollout strategies (blue-green deployment, canary releases) for minimizing downtime and ensuring application availability.

### Configuration Management

• Management of configuration files, environment variables, and application settings across different deployment environments.

• Monitoring and logging setup for performance monitoring, error tracking, and troubleshooting.

- **User Guide**

### System Access

• User registration and authentication mechanisms.

• Account management (password reset, profile settings) for registered users.

### Features Overview

• Overview of core features (search songs, playlist management, user preferences).

• Usage instructions for accessing and utilizing music streaming functionalities.

### Usage Instructions

• Step-by-step guides for common user tasks (creating playlists, liking songs, exploring recommendations).

• Troubleshooting tips and FAQs for resolving common issues.

- **Maintenance and Support**

## Monitoring and Maintenance Procedures

• Implementation of monitoring tools for real-time performance metrics and system health checks.

• Scheduled maintenance windows and updates for ensuring system reliability and security.

## Bug Tracking and Resolution

• Bug reporting and tracking using issue management tools (Jira, Trello).
• Prioritization, assignment, and resolution of reported issues to maintain application quality.

## Backup and Recovery

• Implementation of backup strategies (regular database backups, file system snapshots) for data protection and disaster recovery.

• Procedures for restoring data and recovering from system failures or data breaches.

# SCOPE OF THE PROJECT

The scope of this project includes the design and development of three static web applications: a Calculator, a Landing Page on current issues, and a Personal Portfolio website. Each project is built using HTML, CSS, and JavaScript to demonstrate front-end skills. The focus is on clean UI design, responsive layouts, and functional features. The project serves educational purposes, showcasing beginner-level development, user interaction, and personal branding in a web environment.

## 1. Functional Scope

- Perform basic calculator operations (addition, subtraction, multiplication, division).
- Provide informative content on social topics in the landing page.
- Showcase skills, projects, and resume in the portfolio.
- Enable download of resume and navigation across website sections.

## 2. Technical Scope

- Built using HTML, CSS, and JavaScript (no backend).
- Internal CSS and inline JavaScript for interactivity (calculator).
- Deployed using static hosting platforms like GitHub Pages or Netlify.
- Developed using VS Code with optional Live Server extension.

## 3. User Experience (UX) Scope

- Simple, intuitive interface with clean layout.
- Responsive design for desktop, tablet, and mobile users.
- Easy navigation with clear headings and well-structured content.
- Downloadable resume and easy contact visibility for convenience.

## 4. Security and Compliance Scope

- No personal data collected—minimizing data risk.

- Avoided external scripts or risky code (e.g., minimized `eval ()` use).
- Static project; no user authentication or backend forms involved.
- Adheres to basic accessibility guidelines (readable text, clear contrast).

## 5. Scalability and Performance Scope

- Lightweight and fast-loading design for performance.
- Can be scaled by adding more projects, blog sections, or calculator features.
- Compatible with future enhancements using external CSS/JS libraries.
- Optimized for cross-browser functionality.

## 6. Feature Enhancement Scope

- Future scope includes adding:
    - Contact forms with validation
    - Dark/Light mode toggle
    - Project filter/search functionality
    - More interactive animations or transitions

## 7. Content Management scope

- Currently managed through manual editing of HTML.
- Future enhancement could include integration with CMS or JSON data.
- Easy to update text, links, and project descriptions in code.

## 8. Analytics and Insights Scope

- Currently no analytics tools integrated.
- Future scope to add:
    - Google Analytics for tracking visitors
    - Resume download counter
    - Button interaction tracking

## 9. Community and Interactive Scope

- No social sharing features currently included.
- Can add:
    - Social media icons in the portfolio
    - Comment/discussion section in the landing page
    - Contact form for user feedback or job inquiries

# BIBLIOGRAPHY


**1. Tutorials:**

- HTML & CSS:

- MDN Web Docs –

HTML: [https://developer.mozilla.org/enUS/docs/Web/HTML] (https://developer.mozilla.org/enUS/docs/Web/HTML)

- MDN Web Docs

– CSS: [https://developer.mozilla.org/en-US/docs/Web/CSS] (https://developer.mozilla.org/enUS/docs/Web/CSS)

- W3Schools

– HTML Tutorial: [https://www.w3schools.com/html/] (https://www.w3schools.com/html/)

- W3Schools

– CSS Tutorial: [https://www.w3schools.com/css/] (https://www.w3schools.com/css/)

US/docs/Web/JavaScript] (https://developer.mozilla.org/en-US/docs/Web/JavaScript)

- JavaScript.info: [https://javascript.info/] (https://javascript.info/)

- W3Schools

– JavaScript Tutorial: [https://www.w3schools.com/js/] (https://www.w3schools.com/js/)

- Git & Version Control:

- Git Handbook:
[https://guides.github.com/introduction/githandbook/]
(https://guides.github.com/introduction/githandbook/)

**2. Deployment Platforms:**

- GitHub Pages: [https://pages.github.com/]
(https://pages.github.com/)