

# **FULL STACK WEB DEVELOPMENT**

An internship report submitted in partial fulfilment of the requirement for the award of the Degree  
of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE AND ENGINEERING

Submitted by

Balla Bharadwaja Siva Subrahmanya Souri

(21021A0542)



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING  
UNIVERSITY COLLEGE OF ENGINEERING KAKINADA (A)  
JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY KAKINADA  
KAKINADA-533003(A.P)

2021-2025

## **DECLARATION**

I hereby declare that work embodied in this internship entitled 'FULL STACK WEB DEVELOPMENT, which is being submitted by me in requirement for the award of the degree of the 'BACHELOR OF THE TECHNOLOGY IN COMPUTER SCIENCE AND ENGINEERING' from Jawaharlal Nehru Technological University Kakinada, is the result of internship training by MAIN FLOW SERVICES AND TECHNOLOGIES Pvt. Ltd. in collaboration with Andhra Pradesh State Council of Higher Education.

With gratitude,

Balla Bharadwaja Siva Subrahmanya Souri

21021A0542

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING  
UNIVERSITY COLLEGE OF ENGINEERING KAKINADA  
JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY KAKINADA  
KAKINADA – 533003, ANDHRA PRADESH, INDIA**



**CERTIFICATE**

This is to certify that this project report entitled **“FULL STACK WEB DEVELOPMENT”** is a bonafide record of the work being submitted by **Balla Bharadwaja Siva Subrahmanya Souri** bearing the roll number **21021A0542**, in the partial fulfillment of the requirements for the award of the degree of Bachelor of Technology in **COMPUTER SCIENCE AND ENGINEERING** to the **UCEK(A), JNTUK, Kakinada, Andhra Pradesh, India**. It has been found satisfactory and hereby found satisfactory and hereby approved for submission.

**Signature of Head of the Department**

Dr N.Ramakrishnaiah

Professor & HOD

Department of CSE

UCEK (A)

JNTU KAKINADA

# CERTIFICATE OF COMPLETION

ID- 11268

## CERTIFICATE Of Internship



This is to certify that – **Balla Bharadwaja Siva Subrahmanya Souri**

**Has Attended** Full Stack Web Development

Duration: (25 April 2024 to 25 June 2024)

“During the internship, he/she has demonstrated exceptional dedication, enthusiasm, and a strong willingness to learn. They actively engaged in various projects and tasks assigned to them, exhibiting remarkable skills and a high level of professionalism.”

*Gauravkumar*  
**Director**



MAIN FLOW SERVICES AND TECHNOLOGIES

# **ABSTRACT**

The Full Stack Web Development project is an in-depth initiative designed to demonstrate a holistic understanding of web technologies and development processes. This project aims to create a responsive, user-friendly, and dynamic web application by integrating both front-end and back-end technologies. The application addresses real-world user requirements by offering a seamless and interactive experience, emphasizing scalability, performance, and security. Through this endeavour, the project illustrates the developer's ability to bridge the gap between user interface design and server-side functionality.

The front-end of the application is built using HTML, CSS, and JavaScript, with additional support from modern libraries like React.js to create an interactive user experience. It employs responsive design principles, ensuring compatibility across multiple devices, including desktops, tablets, and smartphones. Advanced CSS techniques and JavaScript frameworks are used to implement dynamic content rendering and visually engaging interfaces. This allows users to interact intuitively with the application, enhancing usability and accessibility. The front-end also integrates APIs to fetch and display data dynamically, providing real-time updates to the users.

On the backend, the project utilizes technologies such as Node.js or PHP, coupled with databases like MySQL, to handle server-side operations and manage data securely. This ensures robust performance under varying loads while maintaining data integrity and security. The back-end architecture is designed to be modular, making the system scalable and maintainable for future expansions. Features like user authentication, role-based access control, and secure data handling are implemented to protect sensitive information and enhance trustworthiness.

This project serves as a comprehensive demonstration of full-stack web development skills, from coding and debugging to deploying a fully functional application. It reflects a deep understanding of the software development lifecycle, including planning, implementation, testing, and deployment phases. Additionally, the project emphasizes collaboration tools such as Git for version control and project management, showcasing the developer's ability to work in multi-developer environments. By addressing practical problems and creating scalable solutions, this project not only highlights technical expertise but also the ability to think critically and deliver user-centric designs.

**Introduction** **8****Week-1** **9-16**

## ➤ LANDING PAGE USING HTML, CSS

- Introduction to Website Working, History of Websites.
- Introduction to HTML, HTML Semantics, HTML Tags
- Introduction to Document Object Model.
- Introduction to Styling, Cascading Style Sheets (CSS).
- Types of CSS style inclusion.
- Types of CSS Selectors, Properties and Attributes.
- General structure of Website using HTML and CSS.
- Developing a Landing Page for any Website using HTML, CSS.

**Week-2** **17-18**

## ➤ SIMPLE WEBSITE USING RESPONSIVE DESIGN

- Project Overview
- Technologies Used
- Features
- Code Analysis
- Challenges and Solutions
- Conclusion

**Week-3** **19-22**

## ➤ LOGIN SIGN UP PAGE with JS, PHP, MySQL

- Project Overview
- Technologies Used
- Features
- Code Analysis
- Challenges and Solutions
- Conclusion

**Week-4** **23-26**

## ➤ TO-DO LIST APPLICATION USING REACT JS.

- Project Overview
- Technologies Used
- Features
- Code Analysis
- Challenges and Solutions
- Conclusion

**Week-5 & 6****27-29**

➤ SIMPLE CALCULATOR USING REACT JS.

- Project Overview
- Technologies Used
- Features
- Code Analysis
- Challenges and Solutions
- Conclusion

**Week-7 & 8****30-34**

➤ GALLERY APPLICATION USING REACT JS.

- Project Overview
- Technologies Used
- Features
- Code Analysis
- Challenges and Solutions
- Conclusion

# **INTRODUCTION**

Full Stack Web Development has emerged as a crucial skill set in the rapidly evolving world of technology. It encompasses the development of both the front-end, which focuses on the user interface and experience, and the backend, which handles server-side logic, database management, and application functionality. This combination of skills enables developers to build complete, scalable, and efficient web applications that cater to a wide range of user needs. The ability to work on the entire stack of technologies allows developers to streamline processes, reduce dependencies, and deliver cohesive solutions.

The objective of this project is to design and develop a fully functional web application that showcases the integration of front-end and back-end technologies. The application aims to provide a responsive, interactive, and user-friendly experience while handling data securely and efficiently. The project emphasizes real-world applicability by incorporating features such as dynamic content rendering, secure user authentication, and database-driven functionality. These elements demonstrate the developer's ability to address practical challenges in web development while delivering high-quality solutions.

The project employs modern technologies for both front-end and back-end development. On the front-end, tools like HTML, CSS, JavaScript, and frameworks such as React.js are utilized to create a visually appealing and interactive user interface. The back-end leverages server-side technologies like Node.js or Python and integrates databases such as MySQL or MongoDB to manage and process data effectively. This combination of technologies ensures that the application is robust, scalable, and capable of handling complex operations seamlessly.

This introduction sets the stage for understanding the significance of full-stack web development and the purpose of the project. It highlights the importance of integrating various technologies to create cohesive solutions that meet user expectations. By delving into this project, the developer not only demonstrates technical proficiency but also showcases problem-solving skills, adaptability, and the ability to deliver impactful web applications that can be scaled for broader use cases.



# **Week-1 Report**

## **TASK 1: Landing Page**

- Introduction to Website Working, History of Websites.
- Introduction to HTML, HTML Semantics, HTML Tags
- Introduction to Document Object Model.
- Introduction to Styling, Cascading Style Sheets (CSS).
- Types of CSS style inclusion.
- Types of CSS Selectors, Properties and Attributes.
- General structure of Website using HTML and CSS.
- Developing a Landing Page for any Website using HTML, CSS.

## **INTRODUCTION:**



- A website (also written as a web site) is one or more web pages and related content that is identified by a common domain name and published on at least one web server.
- Websites are typically dedicated to a particular topic or purpose, such as news, education, commerce, entertainment, or social media.
- Hyperlinking between web pages guides the navigation of the site, which often starts with a home page. The most-visited sites are Google, YouTube, and Facebook.
- All publicly-accessible websites collectively constitute the World Wide Web. There are also private websites that can only be accessed on a private network, such as a company's internal website for its employees.
- While "web site" was the original spelling (sometimes capitalized "Web site", since "Web" is a proper noun when referring to the World Wide Web), this variant has become rarely used, and "website" has become the standard spelling.
- After reaching 1 billion websites in September 2014, a milestone confirmed by Netcraft in its October 2014 Web Server Survey and that Internet Live Stats was the first to announce as attested by this tweet from the inventor of the World Wide Web himself, Tim Berners-Lee the number of websites in the world have subsequently declined, reverting to a level below 1 billion.

## INTRODUCTION TO HTML:



- Hypertext Markup Language (HTML) is the standard markup language for documents designed to be displayed in a web browser.
- It defines the content and structure of web content. It is often assisted by technologies such as Cascading Style Sheets (CSS) and scripting languages such as JavaScript, a programming language.
- Web browsers receive HTML documents from a web server or from local storage and render the documents into multimedia web pages. HTML describes the structure of a web page semantically and originally included cues for its appearance.
- HTML elements are the building blocks of HTML pages. With HTML constructs, images and other objects such as interactive forms may be embedded into the rendered page.
- HTML provides a means to create structured documents by denoting structural semantics for text such as headings, paragraphs, lists, links, quotes, and other items.
- HTML elements are delineated by tags, written using angle brackets. Browsers do not display the HTML tags but use them to interpret the content of the page.

## SYNTAX:

```
<!DOCTYPE html>
<html>
  <head>
    <title>Page Title</title>
  </head>
  <body>
    <h1>Hello, World!</h1>
    <p>This is a paragraph.</p>
  </body>
</html>
```

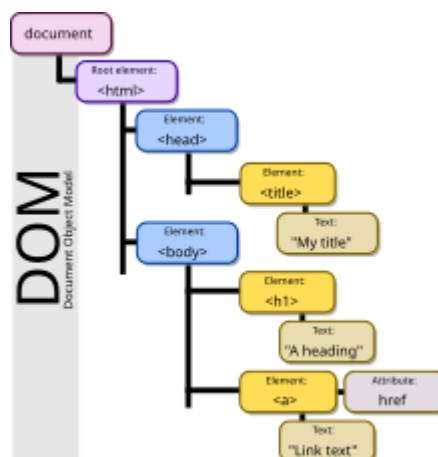
## OUTPUT:



**Hello, World!**

This is a paragraph.

## DOCUMENT OBJECT MODEL:



- The Document Object Model (DOM) is a cross-platform and language-independent interface that treats an HTML or XML document as a tree structure wherein each node is an object representing a part of the document.
- The DOM represents a document with a logical tree. Each branch of the tree ends in a node, and each node contains objects.
- DOM methods allow programmatic access to the tree; with them one can change the structure, style or content of a document. Nodes can have event handlers (also known as event listeners) attached to them. Once an event is triggered, the event handlers get executed.

In HTML DOM (Document Object Model), every element is a node:

- A document is a document node.
- All HTML elements are element nodes.
- All HTML attributes are attribute nodes.
- Text inserted into HTML elements are text nodes.
- Comments are comment nodes.

## INTRODUCTION TO CSS:



- Cascading Style Sheets (CSS) is a style sheet language used for specifying the presentation and styling of a document written in a markup language such as HTML or XML (including XML dialects such as SVG, MathML or XHTML).
- CSS is a cornerstone technology of the World Wide Web, alongside HTML and JavaScript.
- CSS is designed to enable the separation of content and presentation, including layout, colors, and fonts.
- This separation can improve content accessibility, since the content can be written without concern for its presentation; provide more flexibility and control in the specification of presentation characteristics.
- The CSS enables multiple web pages to share formatting by specifying the relevant CSS in a separate .css file, which reduces complexity and repetition in the structural content; and enable the .css file to be cached to improve the page load speed between the pages that share the file and its formatting.

### SYNTAX:

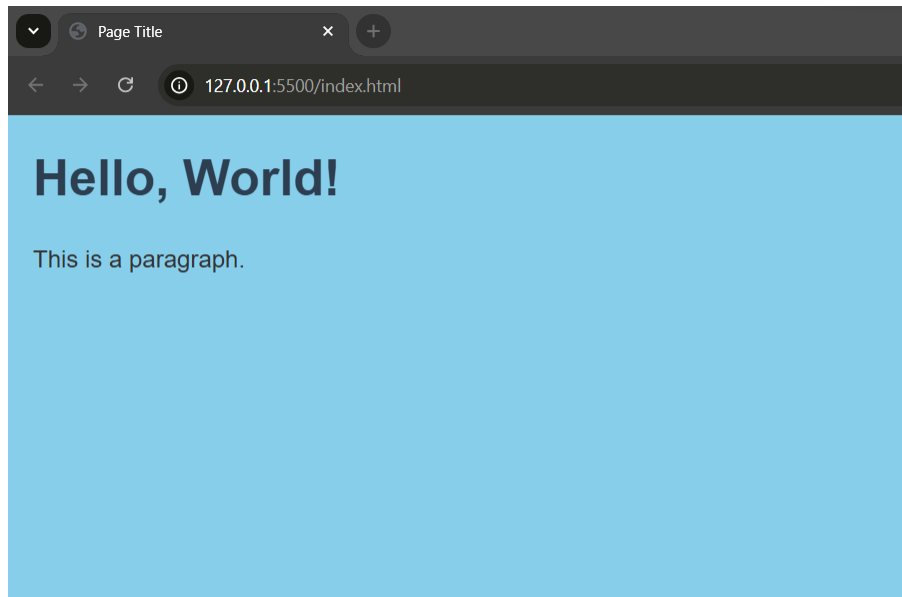
#### index.css:

```
body {  
    font-family: Arial, sans-serif;  
    margin: 20px;  
    background-color: skyblue;  
    color: #333;  
}  
  
h1 {  
    color: #2c3e50;  
    font-size: 2.5em;  
}
```

#### index.html:

```
<body>  
    <h1>Hello, World!</h1>  
    <p>This is a paragraph.</p>  
</body>
```

## OUTPUT:



## CSS SELECTORS:

The following is a list of the most common and well-supported CSS selectors. There are many, many more, but these are the ones you should know well.

- Element Type Selectors
- Class selectors
- Id Selectors

### a) Element Type Selectors:

The most basic CSS selectors are Element Type Selectors. That's a fancy name for simply using an HTML tag, without the angle braces.

For example, to make all paragraphs have green text, we would use the following CSS rule:

```
p { color: green; }
```

### b) Class Selectors:

To match a specific class attribute, we always start the selector with a period, to signify that we are looking for a class value. The period is followed by the class attribute value we want to match.

For example, if we wanted all elements with a class of "highlight" to have a different background color, we would use the following CSS rule:

```
.highlight { background-color: #ffcccc; }
```

### c) Id Selectors:

To match a specific id attribute, we always start the selector with a hash symbol (#), to signify that we are looking for an id value. We can only use the same id attribute value once, so the id selector will always only match one element in our document.

For example, element with an id of "content", we would use the following CSS rule:

```
#content { border: 2px solid green; }
```

## **TASK:** Developing a Landing Page for any Website using HTML, CSS.

### **Objective:**

The primary objective of this project was to design and develop a responsive and visually appealing landing page using HTML and CSS. This landing page serves as a personal portfolio for showcasing professional achievements, projects, and contact details.

### **Introduction:**

A personal portfolio website is an effective way to highlight one's skills, educational background, and accomplishments. The aim of this project was to create a simple yet functional landing page that provides a brief overview of the creator's profile in an organized and aesthetically pleasing manner.

### **Features of the Landing Page:**

#### **1. Overview Section:**

- Introduces the creator as an aspiring programmer and developer with a keen interest in machine learning and web development.

#### **2. Education Details:**

- Displays educational qualifications in a tabular format, including the institution, duration, and academic performance.

#### **3. Projects:**

- Lists significant projects with descriptions and links to the respective GitHub repositories.

#### **4. Achievements:**

- Highlights professional and academic accomplishments such as competitive coding milestones and rankings.

#### **5. Contact Details:**

- Includes clickable links for email, LinkedIn, phone number, and GitHub profile for seamless communication.

### **Technologies Used:**

#### **1. HTML (HyperText Markup Language):**

- Structured the content of the landing page using semantic tags for better readability and SEO.

#### **2. CSS (Cascading Style Sheets):**

- Styled the page to enhance visual appeal, including:
  - Background color: Antiquewhite for a subtle and clean look.
  - Center-aligned headings for symmetry.
  - Underlined headings and hover effects on links for interactivity.

## **Code Highlights:**

### **1. HTML:**

- Used <div> elements to organize content into logical sections.
- Incorporated hyperlinks with hover effects to improve navigation.
- Applied semantic tags like <h1>, <h2>, <h3>, and <table> for structure and hierarchy.

### **2. CSS:**

- Enhanced usability and design through:
  - Link hover effects to indicate interactivity.
  - Table styling with borders and padding for a clean layout.
  - Global styling for consistency across the page.

## **Challenges Faced:**

- Ensuring the layout remains consistent across different devices.
- Structuring content in an easy-to-read format while maintaining visual appeal.

## **Solutions Implemented:**

- Used a responsive design by applying relative units and consistent styling.
- Tested the webpage on multiple screen sizes to ensure compatibility.

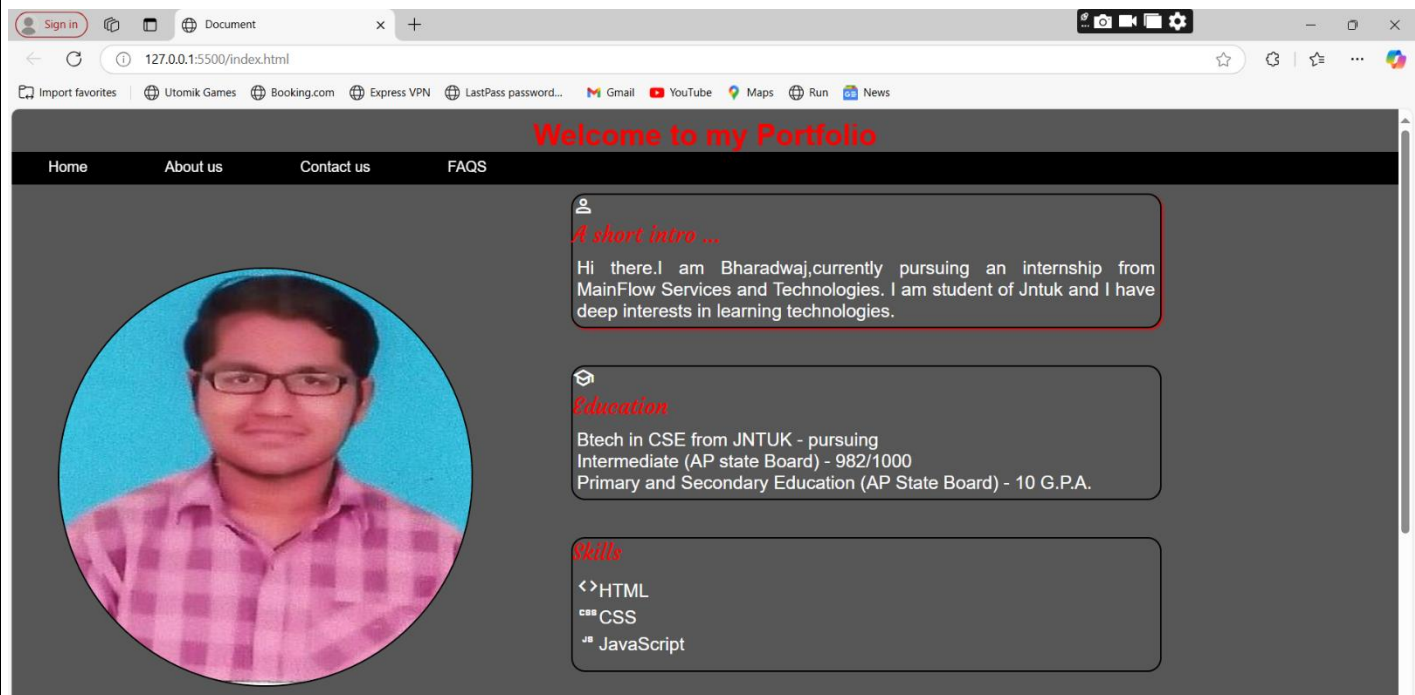
## **Outcomes:**

- Successfully developed a portfolio landing page that effectively showcases professional and personal information.
- Demonstrated proficiency in HTML and CSS for creating a user-friendly interface.
- Enhanced understanding of web development principles and best practices.

## **Conclusion:**

This project provided hands-on experience in web development and reinforced the importance of clear structure and aesthetic design in creating an impactful landing page. The resulting webpage is a practical tool for showcasing skills and achievements, establishing a professional online presence.

## OUTPUT:





# **WEEK-2 REPORT**

## **TASK: SIMPLE WEBSITE**

### **Description:**

Develop a Website with DropDowns and make sure that you can navigate between the pages.

### **Responsibilities:**

- Arrange the drop down to the content of the web page to make it responsive.
- Establish the connection between the pages through navigation.

### **1. Project Overview:**

This project is an HTML and CSS-based webpage created to showcase the life and teachings of Swami Vivekananda. The webpage includes several sections highlighting key aspects of his life, including his birth, education, teachings, influence, works, and death. The design emphasizes responsiveness, interactivity, and user engagement, adhering to modern web development practices.

### **2. Technologies Used:**

#### **Frontend:**

#### **a) HTML**

- Used for structuring the content of the webpage.
- Contains headings, paragraphs, lists, links, and images.

#### **b) CSS**

- Provides styling and layout design.
- Includes animations, hover effects, and responsive design using media queries.

### **3. Features:**

#### **Content Structure**

- Divided into distinct sections:
  - Birth and Childhood
  - Education
  - Teachings and Philosophy
  - Influence and Legacy
  - Works
  - Death

### **4. Navigation:**

- Each section contains clickable links leading to relevant Wikipedia pages for additional information.

### **5. Design Highlights:**

#### **a) Typography:**

- Fonts imported from Google Fonts (David Libre).
- Hierarchical font sizes for headings and paragraphs.

#### **b) Colors:**

- The background and border colors were carefully chosen to reflect a calm and readable design (e.g., sky blue for sections, gold for borders).

#### **c) Images:**

- An image of Swami Vivekananda is included with proper styling and alt text for accessibility.

## 6. Responsive Design:

- Media queries ensure that the layout adapts gracefully to different screen sizes:
  - **Desktop View:** Multiple columns with equal-width sections.
  - **Tablet View:** Single-column layout with sections stacked vertically.
  - **Mobile View:** Simplified layout, reduced hover effects for a better mobile experience.

## 7. Code Analysis:

- Semantic elements (e.g., <h1>, <p>, <ul>, <li>) ensure content is organized and accessible.
- External links open in new tabs (target="\_blank") to maintain user focus on the webpage.
- Class-based Styling.
- Hover Effects.
- Responsive Design.
- Flexbox.

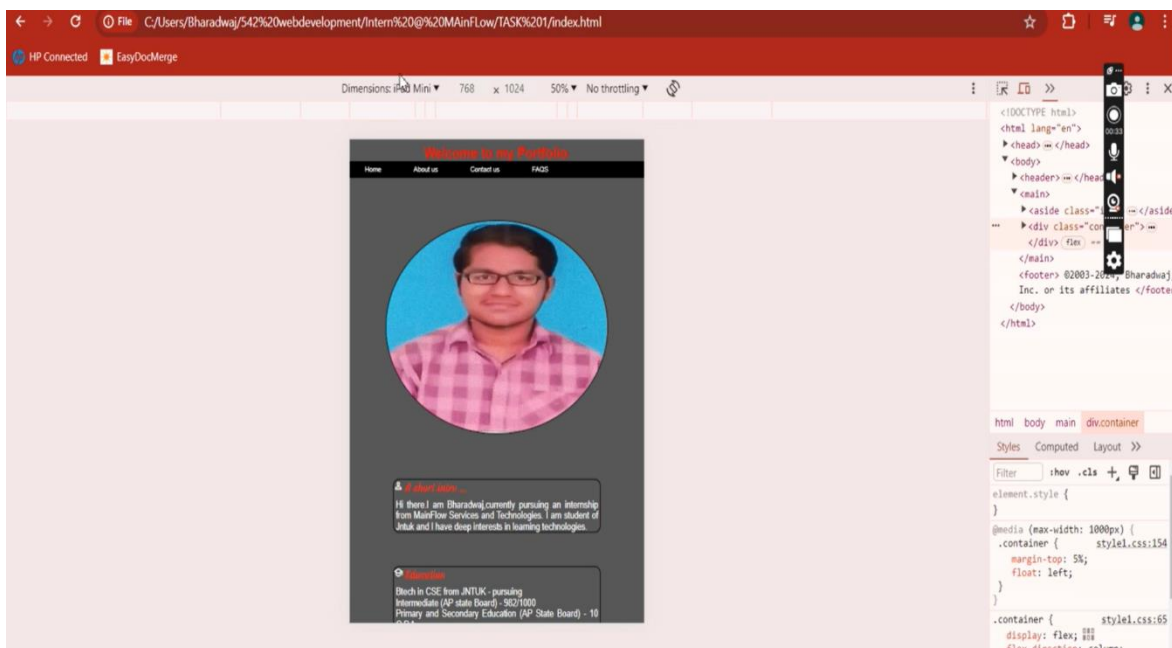
## 8. Challenges and Solutions:

- a) **Challenge:** Ensuring proper scaling and alignment of images.
  - **Solution:** Applied flexbox and media queries to manage alignment and scaling dynamically.
- b) **Challenge:** Maintaining hover effects across devices.
  - **Solution:** Disabled scaling effects for smaller screens to improve usability.
- c) **Challenge:** Ensuring the design remains readable and engaging on all devices.
  - **Solution:** Adjusted font sizes and section widths based on screen sizes.

## 9. Conclusion:

The project effectively demonstrates fundamental web development skills, including structuring content with HTML, styling with CSS, and creating a responsive and interactive design. The focus is on the Swami Vivekananda's life makes the webpage both informative and inspiring for users.

## OUTPUT:



# **Week-3 Report**

## **TASK: LOGIN SIGN UP PAGE**

### **Description:**

Build a Login and Signup Page using PHP

### **Responsibilities:**

- Use HTML, CSS for frontend
- Use MYSQL as Database
- Use PHP to connect the frontend with backend

### **Title: Login and Signup Page Using PHP**

#### **Objective:**

The primary objective of this project is to build a fully functional and user-friendly Login and Signup page using PHP, HTML, CSS, JavaScript, and MySQL for database integration. This ensures users can securely register and log in to a web application.

#### **Technologies Used:**

##### **1. Frontend:**

- HTML: To structure the web pages.
- CSS: For styling and creating a responsive design.
- JavaScript: For client-side validation of inputs.

##### **2. Backend:**

- PHP: To handle server-side logic and communicate with the database.

##### **3. Database:**

- MySQL: For storing user information such as username, email, and password.

#### **Features:**

##### **1. Signup Functionality:**

- Users can create a new account by providing a username, email, and password.
- Validations for input fields to ensure proper data entry (e.g., email format, password length).
- Secure password storage using hashing (encrypt).
- Validation to prevent duplicate email registrations.

##### **2. Login Functionality:**

- Users can log in with their registered email and password.
- Validation of credentials against the database.

### 3. Input Validation:

- Client-side validation using JavaScript.
- Server-side validation in PHP to ensure security and proper data entry.

### 4. Database Integration:

- MySQL is used to store and retrieve user data.
- Proper database schema to handle unique constraints for emails.

## Implementation Details:

### Frontend Development

- **HTML:** The structure of the Signup and Login forms includes fields for username, email, password, and confirm password.
- **CSS:** A responsive and modern UI design is created using CSS. The background has a gradient effect, and form components are styled for consistency.
- **JavaScript:** JavaScript validates input fields in real-time and provides user feedback for errors (e.g., invalid email or mismatched passwords).

### Backend Development

- **PHP:** PHP scripts handle form submissions, validate inputs, hash passwords, and interact with the MySQL database for data storage and retrieval.
- **Database Operations:**
  - INSERT operations for user registration.
  - SELECT operations for login verification.
  - Passwords are securely hashed before storage to ensure data security.

### Database Design:

- **Database Name:** user\_database
- **Table Name:** users
- **Schema:**

```
CREATE TABLE users (  
    id INT AUTO_INCREMENT PRIMARY KEY,  
    username VARCHAR(50) NOT NULL,  
    email VARCHAR(100) NOT NULL UNIQUE,  
    password VARCHAR(255) NOT NULL  
);
```

## Workflow:

### 1. User Registration:

- A user fills out the signup form.
- The client-side validation checks for empty fields, valid email, and password match.
- The data is sent to the process.php script.
- PHP validates inputs, checks for existing email, and inserts a new user into the database after hashing the password.

### 2. User Login:

- A user enters their email and password on the login page.
- PHP fetches the stored hashed password for the email and verifies it using password\_verify().
- If valid, a session is started, and the user is authenticated.

## Code Overview:

### Frontend (HTML, CSS, JavaScript):

- **HTML:** Provides the structure for the Signup and Login forms.
- **CSS:** Styles the forms and ensures responsiveness.
- **JavaScript:** Handles real-time input validation and user feedback.

### Backend (PHP):

- Handles user input securely by validating and sanitizing data.
- Hashes passwords using password\_hash().
- Authenticates users during login using password\_verify().

### Database (MySQL):

- Stores user information securely.
- Ensures email uniqueness with a unique constraint.

## Testing:

### Test Cases

Test Scenario	Expected Outcome	Result
Signup with valid details	User is registered successfully	Passed
Signup with existing email	Email is already registered	Passed
Login with correct credentials	User is logged in successfully	Passed
Login with wrong credentials	Invalid email or password.	Passed
Password, confirm password	Passwords must be the same	Passed

## Challenges Faced:

1. **Securing User Data:** Ensuring passwords are stored securely using hashing.
2. **Validation:** Implementing both client-side and server-side validation to prevent invalid inputs and security threats like SQL injection.
3. **Error Handling:** Providing clear error messages for various user actions.

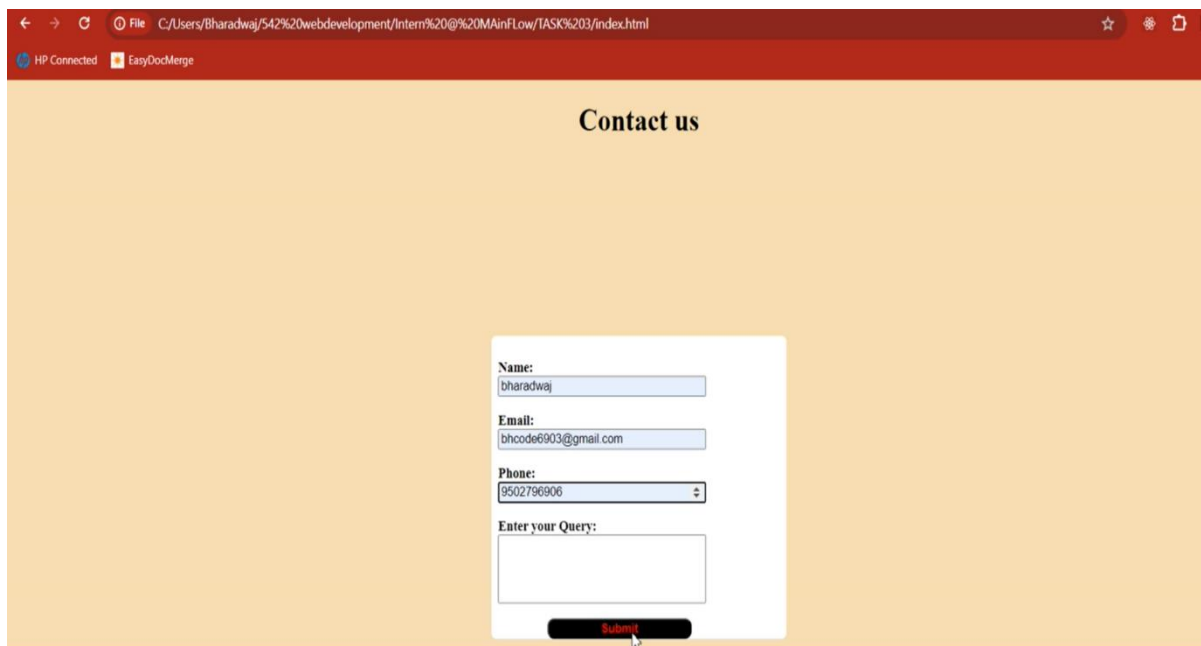
## Future Enhancements:

1. Add password recovery functionality.
2. Implement email verification during signup.
3. Enhance UI/UX with animations and advanced design.
4. Use a framework like Laravel for better scalability and maintainability.

## Conclusion:

This project demonstrates a simple yet effective implementation of a Login and Signup system. By combining frontend, backend, and database technologies, a secure and user-friendly system was created to handle user authentication seamlessly.

## OUTPUT:



The screenshot displays a web browser window with a red title bar. The address bar shows the file path: C:/Users/Bharadwaj/542%20webdevelopment/Intern%20@%20MainFlow/TASK%203/index.html. The browser's status bar at the bottom indicates 'HP Connected' and 'EasyDocMerge'. The main content area has a light orange background with the heading 'Contact us' in bold black text. Below the heading is a white form with the following fields: 'Name:' with a text input containing 'bharadwaj', 'Email:' with a text input containing 'bhcode6903@gmail.com', 'Phone:' with a text input containing '9502796906' and a dropdown arrow, and 'Enter your Query:' with a text area. A red 'Submit' button is located at the bottom of the form.

# **Week-4 Report**

## **Description:**

Develop a simple to-do list application where users can add tasks, mark tasks as completed, and delete tasks.

## **Implementation:**

1. Create a React component for the to-do list.
2. Use React state (useState hook) to manage the list of tasks.
3. Implement event handlers for adding, completing, and deleting tasks.
4. Use conditional rendering to display completed tasks differently.
5. Apply basic CSS for styling the to-do list and task items.

**Project Title:** To-Do List Application

## **Description:**

The To-Do List Application is a simple yet effective tool for managing daily tasks. The application allows users to add new tasks, mark tasks as completed, and delete tasks when no longer needed. It provides a clear and interactive user interface to help users organize their work efficiently.

## **Implementation Details:**

1. **Frontend Framework:** React.js
  - Utilized React's component-based architecture to build a modular and reusable user interface.
2. **State Management:**
  - Managed application state using the useState hook to store tasks, handle changes dynamically.
3. **Key Features:**
  - **Add Task:** Users can enter a task in input field, click the "Add Task" button to add it to the list.
  - **Mark as Completed:** Users can mark a task as completed by clicking the checkbox or task text. Completed tasks are visually distinguished.
  - **Delete Task:** Users can delete tasks by clicking the "Delete" button associated with each task.
4. **Event Handling:**
  - Added event listeners to handle actions such as adding, toggling completion, and deleting tasks.
  - Used functions to update the state and re-render the UI accordingly.
5. **Styling:**
  - Applied basic CSS for a visually appealing and responsive interface.
  - Used conditional styling to differentiate completed tasks (e.g., strikethrough text).

## Code Explanation:

### 1. React Component:

- ToDoList is the primary component managing all functionalities.
- It contains state variables tasks (an array of task objects) and taskInput (the current input value).

### 2. Add Task:

- A new task is added when the user types in the input field and clicks the "Add Task" button.
- Tasks are stored as objects with properties text (task description) and completed (boolean for task status).

### 3. Mark Task as Completed:

- Clicking the checkbox or task text toggles the completed property of the task object.
- This triggers a re-render to visually reflect the task's completion status.

### 4. Delete Task:

- Tasks are removed from the list using the filter method to create a new array excluding the selected task.

### 5. CSS:

- A completed class is applied to tasks marked as completed, applying strikethrough styling.

## Key Code Snippets:

### • Add Task Function:

```
const addTask = () => {  
  if (taskInput.trim() !== "") {  
    setTasks([...tasks, { text: taskInput, completed: false }]);  
    setTaskInput("");  
  }  
};
```

### • Toggle Task Completion:

```
const toggleTaskCompletion = (index) => {  
  const newTasks = tasks.map((task, i) => (  
    i === index ? { ...task, completed: !task.completed } : task  
  ));  
  setTasks(newTasks);  
};
```



- **Delete Task:**

```
const deleteTask = (index) => {  
  const newTasks = tasks.filter((_, i) => i !== index);  
  setTasks(newTasks);  
};
```

### **Styling Highlights:**

- Strikethrough styling applied to completed tasks:

```
.completed {  
  text-decoration: line-through;  
  color: grey;  
}
```

### **Challenges Faced:**

1. Managing state updates efficiently for multiple actions (add, toggle, delete).
2. Ensuring a seamless user experience with proper visual feedback for each action.

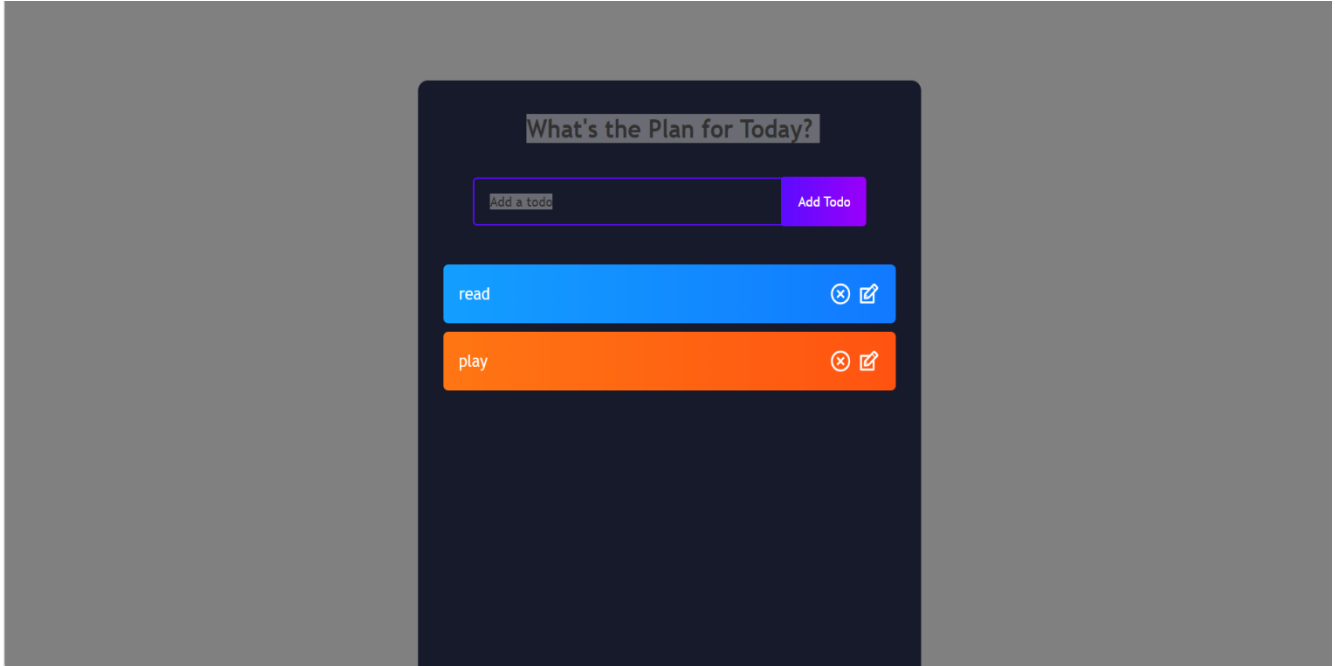
### **Future Enhancements:**

1. Add local storage to persist tasks between sessions.
2. Introduce categories or labels for better task organization.
3. Implement due dates and reminders for tasks.
4. Enhance styling and responsiveness for various screen sizes.

### **Conclusion:**

The To-Do List Application demonstrates the power and flexibility of React.js for building interactive and dynamic web applications. By implementing core functionalities like adding, completing, and deleting tasks, the project showcases essential React concepts such as state management, event handling, and component rendering.

Output:



## **Week- 5 & 6 Report**

### **TASK: SIMPLE CALCULATOR**

#### **Description:**

Build a basic calculator application that can perform simple arithmetic operations like addition, subtraction, multiplication, and division.

#### **Implementation:**

1. Create React components for the calculator interface, number, and operation buttons.
2. Use React state (useState hook) to manage the input and output of the calculator.
3. Implement event handlers for button clicks to update the input and perform calculations.
4. Style the calculator using CSS for a user-friendly interface.

#### **Overview**

This project is a simple calculator application built using React. It allows users to perform basic arithmetic operations such as addition, subtraction, multiplication, and division, as well as clearing inputs and evaluating expressions. The calculator also includes error handling for invalid or undefined calculations.

#### **Project Components:**

The application is structured into three main components:

**App.js** – The main container for the application, which includes state management and handles the logic for calculating results.

**Button.js** – Represents individual buttons in the calculator.

**ButtonPanel.js** – Manages the layout of the calculator buttons.

**Display.js** – Displays the current value of the calculation.

#### **Components Breakdown:**

##### **1. App.js**

This is the main component that contains the state for storing the value being calculated and the logic for evaluating the mathematical expressions.

State Management: The app uses the useState hook to manage the value, which stores the current input or result.

## Functions:

- equals(e): This function evaluates the expression stored in the value state using JavaScript's eval function. It also handles invalid operations and disables buttons when necessary.
- clearFunction(e): Resets the value and enables all buttons for a new calculation.

## 2. Button.js

This component represents a single button on the calculator.

### Props:

- value: The value or symbol that the button represents (e.g., numbers, operators).
- onClick: A function to be executed when the button is clicked.
- id: The unique identifier for each button.
- className: The CSS class used for styling the button.

## 3. ButtonPanel.js

This component organizes and renders the buttons on the screen.

It uses a grid-like structure to arrange buttons into rows, each containing a set of related calculator symbols.

It passes the handleClick function to each button to append the respective value to the current input.

## 4. Display.js

This component displays the current value in an input field.

The input is read-only, allowing users to only view the result or the expression they are typing.

## Features:

- Basic Arithmetic Operations: Users can perform addition, subtraction, multiplication, and division.
- Clear and Delete: The "AC" button resets the input, and the "DE" button deletes the last character.
- Error Handling: Invalid operations (e.g., dividing by zero or entering undefined values) are handled gracefully by disabling buttons and showing an error message.
- Responsive Interface: The layout is designed to work well on both desktop and mobile screens.
- Implementation Details
- State Management: The useState hook is used to manage the state of the input and result.
- Error Handling: The app uses a try-catch block to handle errors during calculation. If an invalid operation occurs (like division by zero or an invalid expression), the result is set to "Invalid", and all calculator buttons are disabled except for the "AC" button.

## Challenges Faced:

- Handling Edge Cases: Managing invalid operations such as division by zero and ensuring that the calculator only performs valid arithmetic operations was challenging.
- UI Responsiveness: Ensuring that the calculator's layout was responsive and user-friendly on both desktop and mobile devices required careful planning of the button grid.

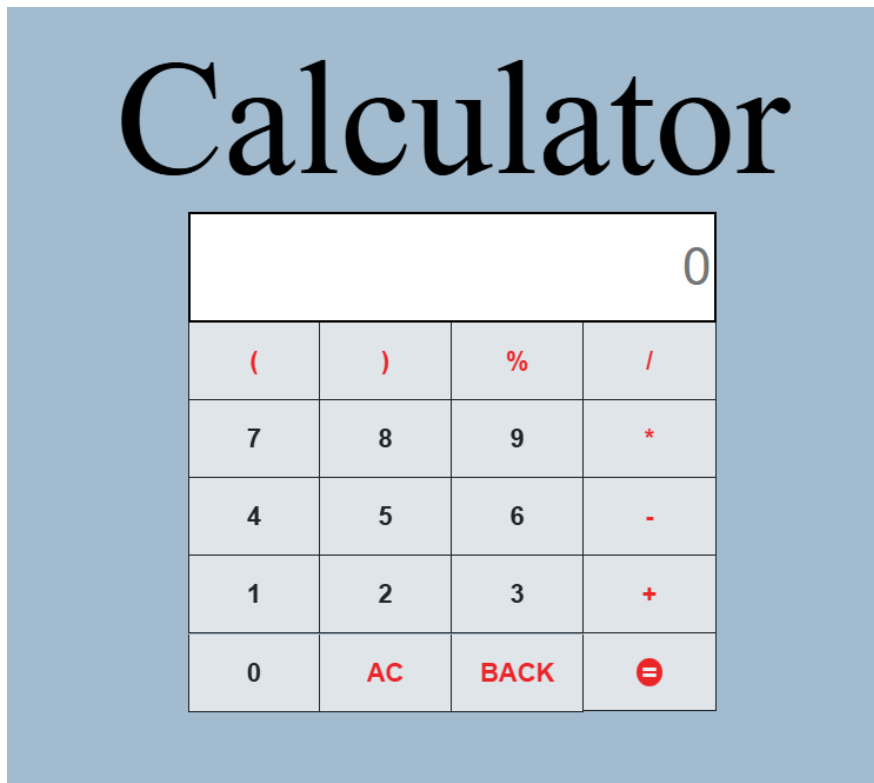
## Future Enhancements:

- Advanced Mathematical Operations: Adding support for additional operations like square roots, exponents, etc.
- History Feature: Implementing a history feature to display previously calculated results.
- Keyboard Support: Enabling keyboard input for a more interactive user experience.
- Theming: Adding dark/light theme toggle for better user experience.

## Conclusion:

The Calculator App demonstrates the power of React for building interactive and responsive web applications. Through simple state management and component organization, the app effectively handles basic arithmetic calculations, error handling, and user interactions. Future updates can further enhance its functionality and user experience.

## OUTPUT:



## **Week- 7 & 8 Report**

### **TASK: IMAGE GALLERY**

#### **Description:**

Develop a simple image gallery application that displays a grid of images with the ability to view each image in a modal or lightbox.

#### **Implementation:**

1. Create React components for the image gallery, individual images, and the modal/lightbox.
2. Use React state (useState hook) to manage the currently selected image in the modal.
3. Implement event handlers to open and close the modal and navigate between images.
- 4 Style the image gallery and modal using CSS for a visually appealing presentation.

#### **Overview:**

The **Image Gallery App** is a dynamic and interactive React application that allows users to view a collection of images in a grid. When an image is clicked, it opens in a modal window with the option to navigate between images using "Next" and "Previous" buttons. This project demonstrates the use of React components, state management, and event handling to create a smooth and user-friendly experience.

#### **Project Features:**

- **Image Gallery:** Displays a collection of images in a grid format.
- **Modal View:** Opens a modal to view images in larger format.
- **Image Navigation:** Users can navigate between images in the modal using "Next" and "Previous" buttons.
- **Close Modal:** The modal can be closed by clicking the close button or clicking outside the modal.
- **Responsive Design:** The application is designed to be fully responsive, ensuring a seamless experience across all screen sizes.

## Components:

The app is divided into the following key components:

### 1. App.js:

- The root component that initializes and renders the ImageGallery component.

### 2. ImageGallery.js:

- Displays the gallery of images and handles the logic for opening and closing the modal, as well as navigating between images.
- **State Management:** Manages the currently selected image using the useState hook.
- **Functions:**
  - openModal: Sets the clicked image as the selected image and opens the modal.
  - closeModal: Closes the modal by setting selectedImage to null.
  - nextImage and prevImage: Navigate between images in the gallery.

### 3. ImageItem.js:

- Renders each image as an individual item in the gallery. When an image is clicked, it triggers the openModal function to display the image in the modal.

### 4. Modal.js:

- Displays the selected image in a modal with "Next" and "Previous" buttons to navigate between images.
- **Functions:**
  - closeModal: Closes the modal when clicked.
  - nextImage and prevImage: Navigate through images in the modal.

### 5. CSS Files:

- ImageGallery.css, ImageItem.css, and Modal.css provide the necessary styles to ensure a polished, responsive, and visually appealing user interface.

## Implementation Details:

- **State Management:** React's useState hook is used to store and update the currently selected image in the ImageGallery component.
- **Event Handling:** The app uses onClick event handlers to trigger the opening of the modal and navigate between images.
- **CSS Styling:** The project uses custom CSS to ensure a clean and responsive layout for the gallery and modal components. The modal overlay is styled to cover the entire screen, with the image centered in the modal.

## Installation and Setup:

### Requirements:

- **Node.js:** Version 14.x or higher.
- **React:** React is the core library used to build this application.
- **Text Editor / IDE:** Example: Visual Studio Code, Sublime Text.
- **Web Browser:** A modern browser such as Google Chrome, Firefox, or Edge.

### Steps to Run the Application Locally:

1. Clone the repository or create a new React project.
2. Navigate to the project directory using `cd <project-directory>`.
3. Install the necessary dependencies using:
4. `npm install`
5. Start the development server using:
6. `npm start`
7. Open the browser to `http://localhost:3000` to view and interact with the app.

### Key Features and Functionality:

- **Image Gallery:** Displays images in a grid layout.
- **Modal:** Clicking on an image opens it in a modal with a larger view.
- **Navigation:** Users can navigate to the next or previous image using the buttons within the modal.
- **Close Modal:** The modal can be closed by either clicking the close button or clicking outside the modal window.
- **Responsive Design:** Ensures proper scaling and layout across various device sizes, from mobile phones to desktop screens.



### Challenges Faced:

1. **Image Navigation:** Implementing logic to ensure smooth navigation through images in the modal while handling edge cases such as the first and last images.
2. **Event Handling:** Preventing modal closure when clicking inside the modal while allowing it when clicking outside of it.
3. **Responsiveness:** Ensuring the layout adapts well to different screen sizes and maintaining the integrity of the design.

### Future Enhancements:

1. **Lazy Loading:** Implement lazy loading for images to optimize performance when dealing with large collections.
2. **Zoom Feature:** Add a zoom-in feature to allow users to view images in more detail.
3. **Fullscreen Mode:** Enable a fullscreen mode for better image viewing experience.
4. **Pagination:** Introduce pagination to split the gallery into multiple pages for better organization.
5. **Image Captions:** Add captions or descriptions for each image in the modal view for added context.

### Conclusion:

The **Image Gallery App** is a user-friendly and visually appealing application that allows users to browse images in an interactive way. By leveraging React components, state management, and event handling, the app provides a seamless experience for users. Future improvements can further enhance the functionality and performance of the app.

## OUTPUT:

