

# WEB AND MOBILE APPLICATION DEVELOPMENT LAB ACTIVITIES

## QUARTER 1

### WEEK 1

#### Lab Activity: Setting Up the Development Environment and Initiating a Git Repository

**Objective:** By the end of this lab, students should be able to set up their development environment, configure essential tools, and initiate a Git repository for version control.

**Materials Needed:**

1. Computer with Internet access
2. A text editor or Integrated Development Environment (IDE)
3. Git software (installed on your computer)
4. GitHub account (if not already created)

**Duration:** 2 hours

**Instructions:**

**Part 1: Development Environment Setup**

**1. Install a Text Editor or IDE:**

- a. Choose a text editor or integrated development environment (IDE) of your preference.
- b. Download and install the selected tool on your computer (e.g., Visual Studio Code, Sublime Text, IntelliJ IDEA).

**2. Install Git:**

- a. Download and install Git from the official website: <https://git-scm.com/downloads>.
- b. Download and install GitHub Desktop
- c. Follow the installation instructions for your operating system.

**3. Configure Git:**

- a. Open your terminal or command prompt.
- b. Configure your Git username and email using the following commands (replace **YourName** and **YourEmail** with your actual name and email):

#### 4. Create a GitHub Account (if you don't have one):

- a. Go to <https://github.com>
- b. Sign up for a free GitHub account.

### Part 2: Initiating a Git Repository

#### 1. Create a New Directory:

- a. Using your terminal or command prompt, navigate to the directory where you want to create your project folder.
- b. Create a new directory for your project (e.g., **my-web-app**) using the **mkdir** command.

#### 2. Navigate to the Project Directory:

- a. Move into your project directory using the **cd** command.

#### 3. Initialize a Git Repository:

- a. Initialize a new Git repository in your project directory.

#### 4. Create a Sample File:

- a. Create a sample text file (e.g., **index.html**) inside your project directory.

#### 5. Add and Commit the Sample File:

- a. Stage the **index.html** file for commit.
- b. Commit the staged changes with a meaningful commit message.

#### 6. Create a Remote Repository on GitHub:

- a. Go to <https://github.com/>
- b. Log in to your GitHub account.

#### 7. Push Your Local Repository to GitHub:

- a. Follow the instructions on GitHub to create a new repository. Make sure not to initialize it with a README file, as you already have one.

#### 8. Connect Your Local Repository to the Remote Repository:

- a. Follow the GitHub instructions to add your remote repository as a remote origin for your local repository. It typically involves running a command like:
- b. Replace **your-username** and **your-repository** with your GitHub username and the repository name.

#### 9. Push Your Code to GitHub:

- a. Push your local repository to GitHub using the **git push** command.

**Conclusion:**

By completing this lab, you should have successfully set up your development environment, initiated a Git repository, and pushed your code to GitHub. This activity lays the foundation for version control and collaborative development throughout the course.

## WEEK 2

### Lab Activity: Exploring Internet Fundamentals

**Objective:** The objective of this self-paced lab is to familiarize yourself with essential internet concepts, including the structure of the internet, protocols, and web browsers.

**Materials Needed:**

1. Computer with Internet access
2. Web browser (e.g., Chrome, Firefox)
3. Access to a text editor (optional)

**Duration:** Self-paced

**Instructions:**

**Part 1: Understanding the Internet**

**1. Introduction to the Internet:**

Begin by reading about the internet's role in modern communication, information sharing, and web development. Use online resources or textbooks for this.

**2. Internet Infrastructure:**

Explore the structure of the internet, consisting of interconnected networks. Consider looking for diagrams or visual representations to help you grasp the concept.

**3. Internet Protocols:**

Research the concept of protocols in networking and their role in facilitating communication. Focus on key internet protocols like TCP/IP and HTTP.

**Part 2: Web Browsing and Basic Navigation**

**1. Web Browsers:**

Investigate the role of web browsers as software applications for accessing and rendering web content. You can find information about popular web browsers and their market share online.

**2. Launching a Web Browser:**

If you aren't already using a web browser, open one on your computer.

**3. Navigating the Web:**

Experiment with entering URLs in the browser's address bar and loading websites. Pay attention to the use of HTTPS for secure web communication.

**4. Understanding URLs:**

Analyze the components of a URL (Uniform Resource Locator) - protocol, domain, path, and query parameters. Try dissecting and understanding a few example URLs.

### **Part 3: Practical Activity - Exploring Web Pages**

#### **1. Accessing Websites:**

Visit a variety of websites by entering their URLs in the browser. Explore different types of websites, such as news, e-commerce, and educational sites.

#### **2. Examining Web Pages:**

Take a closer look at the structure of web pages:

- a. Right-click on a page and select "Inspect" to view the HTML source code.
- b. Use the browser's Elements panel to understand the HTML document's structure.
- c. Experiment with inspecting and modifying CSS styles to see their impact on the page's appearance.

#### **3. Browser Developer Tools:**

Learn about browser developer tools (e.g., Chrome DevTools). Explore their features and understand how they can be used for debugging and analyzing web pages.

### **Part 4: Recap and Assignment**

#### **Recap and Assignment:**

1. Summarize the key points you've learned in this lab.
2. As an assignment, research and write a brief report on a specific internet protocol or a notable internet milestone (e.g., the creation of the World Wide Web, the first web browser).

#### **Conclusion:**

This self-paced lab activity is designed to provide you with a fundamental understanding of internet concepts and web browsing. You'll learn about the internet's infrastructure, protocols, and how web browsers enable you to access and interact with web content. It also encourages hands-on exploration of web pages and introduces you to developer tools, which will be valuable for future web development activities in the course.

## WEEK 3

### Lab Activity: Structuring Web Applications with HTML

**Objective:** This self-paced lab aims to understand the fundamentals of HTML (Hypertext Markup Language) and how it is used to structure web pages.

**Materials Needed:**

1. Computer with Internet access
2. Web browser (e.g., Chrome, Firefox)
3. Text editor (e.g., Notepad, Visual Studio Code)

**Duration:** Self-paced

**Instructions:**

**Part 1: Introduction to HTML**

1. **What is HTML?:**
  - a. Start by researching HTML and why it's crucial in web development. You can find introductory articles and videos online.
2. **Basic Structure of an HTML Document:**
  - a. Explore the basic structure of an HTML document, which includes HTML, head, and body elements. You can find examples and explanations in online tutorials.

**Part 2: Creating an HTML Document**

1. **Setting Up a Text Editor:**
  - a. If you don't already have a preferred text editor, choose one (e.g., Visual Studio Code, Sublime Text) and install it on your computer.
2. **Create a New HTML File:**
  - a. Open your text editor and create a new file with the **.html** extension (e.g., **index.html**).
3. **HTML Boilerplate:**
  - a. Use the HTML5 boilerplate code as a starting point for your HTML document. This includes the **<!DOCTYPE html>** declaration and the basic structure with **<html>**, **<head>**, and **<body>** elements.
4. **Adding Content:**
  - a. Inside the **<body>** element, add some basic content. This can include headings, paragraphs, and lists. Experiment with different HTML tags.

**Part 3: HTML Elements and Attributes**

1. **Text Formatting:**
2. Explore HTML text formatting elements such as `<em>`, `<strong>`, `<u>`, and `<br>`. Use these elements to format your text within the document.
3. **Images:**
  - a. Learn how to insert images using the `<img>` element. Include an image in your HTML document and provide alternative text using the **alt** attribute.

#### **Part 4: Creating a Simple Web Page**

1. **Linking Pages:**
  - a. Create a new HTML file (e.g., **about.html**) and link it to your main HTML file using the `<a>` element.
2. **Lists and Tables:**
  - a. Experiment with creating ordered lists (`<ol>`), unordered lists (`<ul>`), and tables (`<table>`) within your HTML document.
3. **Validation:**
  - a. Use online HTML validation tools to check the correctness of your HTML code.

#### **Part 5: Recap and Assignment**

1. **Recap and Assignment:**
  - a. Summarize what you've learned about HTML in this lab.
  - b. As an assignment, create a simple webpage (e.g., a personal bio or a hobby page) using HTML. Incorporate text, images, links, lists, and tables as needed.

#### **Conclusion:**

This self-paced lab activity provides you with hands-on experience in structuring web applications using HTML. You'll learn about the basic structure of an HTML document, common HTML elements, and how to create a simple web page. Completing this activity will set the foundation for your web development skills as you progress in the course.

# QUARTER 2

## WEEK 4

### Lab Activity: Styling Components with CSS3

**Objective:** The objective of this lab is to learn the fundamentals of CSS3 (Cascading Style Sheets) and how to apply styles to HTML elements to enhance the visual presentation of a web page.

**Materials Needed:**

1. Computer with Internet access
2. Web browser (e.g., Chrome, Firefox)
3. Text editor (e.g., Notepad, Visual Studio Code)

**Duration:** Self-paced

**Instructions:**

**Part 1: Introduction to CSS3**

**1. What is CSS3?:**

- ☐ Start by researching what CSS3 is and why it's important in web development. You can find introductory articles and videos online.

**2. Separation of Concerns:**

- ☐ Understand the concept of "separation of concerns" in web development, where HTML is responsible for content and structure, and CSS is used for presentation and styling.

**Part 2: Styling HTML Elements**

**1. Create a New HTML File:**

- ☐ Open your text editor and create a new HTML file (e.g., `index.html`), or use the existing one from Week 3.

**2. Linking CSS:**

- ☐ Inside the `<head>` section of your HTML document, link an external CSS file (e.g., `styles.css`) using the `<link>` element with the `rel` and `href` attributes.

**3. Basic Styling:**

- ☐ In your CSS file (e.g., `styles.css`), select a few HTML elements (e.g., headings, paragraphs) and apply basic styles like changing text color, font size, and background color.



#### 4. CSS Comments:

- ☐ Learn how to add comments to your CSS code to document your styles.

### Part 3: Selectors and Properties

#### 1. CSS Selectors:

- ☐ Explore different types of CSS selectors, including element selectors, class selectors (.classname), and ID selectors (#idname).

#### 2. Applying Styles:

- ☐ Experiment with applying styles to specific elements using various selectors. For example, style all <h2> elements differently from <h1> elements.

#### 3. Box Model:

- ☐ Understand the CSS box model, which includes properties like **margin**, **padding**, **border**, and **width/height**. Apply these properties to elements to control their layout and spacing.

### Part 4: Advanced Styling

#### 1. Text Styling:

- ☐ Style text elements using properties like **font-family**, **text-align**, **text-decoration**, and **line-height**.

#### 2. Backgrounds and Borders:

- ☐ Apply background colors, images, and borders to elements using relevant CSS properties.

### Part 5: Recap and Assignment

#### Recap and Assignment:

- ☐ Summarize what you've learned about CSS3 in this lab.
- ☐ As an assignment, create a simple webpage (e.g., a personal portfolio section) and style it using CSS3. Include text styling, backgrounds, borders, and responsive design concepts if possible.

#### Conclusion:

This self-paced lab activity provides you with hands-on experience in styling HTML components using CSS3. You'll learn about CSS selectors, properties, and the CSS box model. Completing this activity will enable you to enhance the visual presentation of web pages, a crucial skill in web development.

## WEEK 5

### Lab Activity: Exploring Interactivity with JavaScript - ECMAScript 6 (ES6)

**Objective:** The objective of this lab is to introduce students to JavaScript, with a focus on ECMAScript 6 (ES6) features, and how to add interactivity to web pages.

**Materials Needed:**

1. Computer with Internet access
2. Web browser (e.g., Chrome, Firefox)
3. Text editor (e.g., Visual Studio Code, Sublime Text)

**Duration:** Self-paced

**Instructions:**

#### Part 1: Introduction to JavaScript and ES6

**1. What is JavaScript?:**

- ☐ Start by researching what JavaScript is and why it's important in web development. You can find introductory articles and videos online.

**2. ECMAScript 6 (ES6):**

- ☐ Understand the significance of ECMAScript 6 (ES6) in modern JavaScript development and its enhanced features.

#### Part 2: Setting Up Your Environment

**1. Text Editor and Browser:**

- ☐ Ensure that you have a text editor (e.g., Visual Studio Code) and a web browser (e.g., Chrome) installed on your computer.

**2. Creating an HTML File:**

- ☐ Create a new HTML file (e.g., **index.html**) and include a basic structure with **<html>**, **<head>**, and **<body>** elements.

**3. Linking JavaScript:**

- ☐ Inside the **<head>** section of your HTML document, link an external JavaScript file (e.g., **script.js**) using the **<script>** element with the **src** attribute.

#### Part 3: Basic JavaScript Concepts

**1. Writing Your First JavaScript:**

- ☐ In your **script.js** file, write JavaScript code that displays a simple "Hello, World!" message in the browser's console using **console.log()**.

## 2. Variables and Data Types:

- ☐ Learn about variables and data types in JavaScript, including numbers, strings, and booleans. Declare and use variables in your JavaScript code.

### Part 4: ES6 Features

#### 1. Constants with `const`:

- ☐ Explore the use of `const` to declare constants in ES6. Declare a constant variable in your code.

#### 2. Template Literals:

- ☐ Learn how to use template literals to create dynamic strings in JavaScript. Create a template literal in your code.

#### 3. Arrow Functions:

- ☐ Understand how arrow functions work and how they simplify function declarations. Create an arrow function in your code.

#### 4. `let` vs. `var`:

- ☐ Explore the differences between `let` and `var` for variable declaration and scoping. Use `let` in your code to declare a variable within a block.

### Part 5: Basic Interactivity

#### 1. Alert and Confirm Boxes:

- ☐ Use JavaScript to create an alert box and a confirm box with custom messages.

#### 2. Prompt for User Input:

- ☐ Create a JavaScript prompt to ask the user for input and display the input value.

### Part 6: Recap and Assignment

#### Recap and Assignment:

- ☐ Summarize what you've learned about JavaScript and ES6 in this lab.
- ☐ As an assignment, create a simple interactive web page (e.g., a form with input fields) and use JavaScript to add functionality, such as form validation or interactive buttons.

#### Conclusion:

This self-paced lab activity provides you with a foundational understanding of JavaScript, focusing on ECMAScript 6 (ES6) features. You'll learn how to set up your environment, work with variables and data types, and add basic interactivity to web pages. Completing this activity will prepare you for more advanced JavaScript development in the course.

## WEEK 6

### Lab Activity: Introduction to React (States, Hooks)

**Objective:** The objective of this lab is to introduce students to the fundamentals of React, including state management and the use of React Hooks.

**Materials Needed:**

1. Computer with Internet access
2. A text editor or Integrated Development Environment (IDE)
3. Node.js and npm (Node Package Manager) installed

**Duration:** Self-paced

**Instructions:**

**Part 1: Introduction to React**

**1. What is React?:**

- ☐ Start by researching what React is and why it's essential in modern web development. You can find introductory articles and videos online.

**2. Setting Up Your Environment:**

- ☐ Ensure that you have Node.js and npm (Node Package Manager) installed on your computer. You can download them from <https://nodejs.org/>.

**Part 2: Creating a React Project**

**1. Creating a React App:**

- ☐ Open your terminal or command prompt.
- ☐ Use the following command to create a new React application. Replace my-react-app with your preferred project name.

**npx create-react-app my-react-app**

**2. Navigating to the Project Directory:**

- ☐ Move into your project directory using the cd command.

**cd my-react-app**

**Part 3: Understanding React States**

**1. React Component Structure:**

- ☐ Explore the structure of a React component, including the import statement, component definition, and render method.

**2. State in React:**

- ☐ Learn about state in React and how it enables components to manage and store data.
- ☐ Modify the default state of a component to include custom data.

### 3. Rendering State Data:

- ☐ Display the state data within your component's render method.

## Part 4: Introduction to React Hooks

### Introduction to Hooks:

- ☐ Understand the concept of React Hooks and how they simplify state management and side effects.
- ☐ Learn about common React Hooks like `useState()`.

### Using `useState()`:

- ☐ Implement the `useState()` Hook in your React component to manage and update state.

## Part 5: Creating a Simple Interactive Component

### 1. Create an Interactive Component:

- ☐ Develop a simple React component that includes a button and a counter.
- ☐ Use React state to manage and display the count value.

### 2. Adding Functionality:

- ☐ Implement the functionality to increment the count when the button is clicked.

## Part 6: Recap and Assignment

### 1. Recap and Assignment:

- ☐ Summarize what you've learned about React, states, and React Hooks in this lab.
- ☐ As an assignment, enhance your interactive React component by adding a decrement button and implementing its functionality.

### Conclusion:

This self-paced lab activity provides you with an introductory understanding of React, including state management and the use of React Hooks. You'll set up a React project, create a simple interactive component, and gain hands-on experience with React's core concepts. Completing this activity will prepare you for more advanced React development in the course.

# QUARTER 3

## WEEK 7

### Lab Activity: Advanced Concepts in React with Functional Components and Hooks

**Objective:** The objective of this lab is to explore advanced concepts in React using functional components and React Hooks, including lifecycle methods, making HTTP requests using Axios, and the FETCH API.

**Materials Needed:**

1. Computer with Internet access
2. A text editor or Integrated Development Environment (IDE)
3. Node.js and npm (Node Package Manager) installed
4. Axios library installed (`npm install axios`)

**Duration:** Self-paced

**Instructions:**

**Part 1: Recap of React Basics Using Functional Components and Hooks**

**1. Review of React Basics:**

- ☐ Briefly review the fundamental concepts of React, including functional components, state, and React Hooks (`useState`, `useEffect`).

**Part 2: Lifecycle Methods Using Functional Components and Effects**

**1. Understanding Component Lifecycle in Functional Components:**

- ☐ Learn about the lifecycle of a React functional component using React Hooks, specifically the `useEffect` Hook.

**2. Practical Use of Lifecycle Methods with `useEffect`:**

- ☐ Create a new functional React component and implement the `useEffect` Hook to mimic lifecycle behavior. Log messages to the console during different phases of the component's lifecycle.

**Part 3: Making HTTP Requests with Axios in Functional Components**

**1. Introduction to Axios:**

- ☐ Understand the role of Axios as a JavaScript library for making HTTP requests.

- ☐ Learn why Axios is commonly used for handling API calls in React applications.

## **2. Installing Axios:**

- ☐ If you haven't already, install the Axios library in your React project using the following command:

**npm install axios**

## **Part 4: Making API Requests with Functional Components and Axios**

### **1. Creating a Mock API:**

- ☐ Set up a simple mock API or use an online API for testing purposes. You can use JSONPlaceholder (<https://jsonplaceholder.typicode.com/>) for this lab.

### **2. Using Axios for GET Requests in Functional Components:**

- ☐ Create a functional React component that uses Axios to make a GET request to your mock API.
- ☐ Display the fetched data in your component using React Hooks.

## **Part 5: Using the FETCH API in Functional Components**

### **1. Introduction to the FETCH API:**

- ☐ Learn about the FETCH API, which is a built-in JavaScript API for making HTTP requests.

### **2. Making GET Requests with FETCH in Functional Components:**

- ☐ Create another functional React component that uses the FETCH API to make a GET request to the same mock API.
- ☐ Display the fetched data in your component using React Hooks.

## **Part 6: Comparison and Assignment**

### **1. Comparison and Analysis:**

- ☐ Compare and analyze the differences between using Axios and the FETCH API for making HTTP requests with functional components and React Hooks.
- ☐ Consider factors like syntax, ease of use, and compatibility with functional components.

## **Assignment:**

- ☐ As an assignment, choose a real-world API (e.g., weather, news, movies, TV shows, or a public API of your choice) and create a functional React component that fetches and displays data from that API using either Axios or the FETCH API with React Hooks. You may use <https://rapidapi.com/hub>.

## **Conclusion:**

This self-paced lab activity provides you with a deeper understanding of advanced React concepts, including component lifecycle methods and making HTTP requests using functional components and React Hooks. You'll gain hands-on experience with these concepts, preparing you for more complex React applications and data handling in your course while adhering to functional components and Hook principles.

## WEEK 8

### Lab Activity: Introduction to Mobile Application Development

**Objective:** The objective of this lab is to introduce students to the fundamentals of mobile application development, covering the basics of mobile app architecture, design considerations, and platform choices.

**Materials Needed:**

1. Computer with Internet access
2. Mobile device (optional)
3. A text editor or Integrated Development Environment (IDE)
4. Access to a web browser

**Duration:** Self-paced

**Instructions:**

**Part 1: Understanding Mobile App Development**

**1. Introduction to Mobile App Development:**

- ☐ Start by researching what mobile app development is and why it's crucial in today's technology landscape. You can find introductory articles and videos online.

**2. Mobile App Platforms:**

- ☐ Learn about different mobile app platforms, including iOS (Apple), Android (Google), and cross-platform development options like React Native and Flutter.

**Part 2: Choosing a Development Environment**

**3. Setting Up Your Environment:**

- ☐ Ensure that you have a development environment suitable for mobile app development. Depending on your choice of platform (iOS, Android, cross-platform), set up the necessary tools and SDKs.

**Part 3: Mobile App Architecture**

**1. Mobile App Architecture Overview:**



- ☐ Explore the fundamental architecture of a mobile app, including the components of user interface (UI), application logic, and data storage.

## **2. UI/UX Design Considerations:**

- ☐ Understand the importance of user interface (UI) and user experience (UX) design in mobile app development. Research best practices for designing mobile-friendly interfaces.

### **Part 4: Developing a Simple Mobile App**

#### **1. Creating a Hello World Mobile App:**

- ☐ Depending on your platform choice (e.g., Android Studio for Android, Xcode for iOS, React Native, Flutter), create a simple "Hello World" mobile app project.

#### **2. Building and Running Your App:**

- ☐ Use your development environment to build and run your "Hello World" app on an emulator or a physical device (if available).

### **Part 5: Testing and Debugging**

#### **1. Testing Your App:**

- ☐ Explore different methods for testing your mobile app, including manual testing on emulators or physical devices and automated testing using testing frameworks.

#### **2. Debugging:**

- ☐ Learn how to use debugging tools and techniques specific to your chosen platform for identifying and fixing issues in your app.

### **Part 6: Recap and Assignment**

#### **1. Recap and Assignment:**

- ☐ Summarize what you've learned about mobile app development in this lab.
- ☐ As an assignment, create a small mobile app project of your choice. It could be a simple to-do list, calculator, or a concept app. Document your app's features, design choices, and any challenges you faced during development.

### **Conclusion:**

This self-paced lab activity provides you with a foundational understanding of mobile application development. You'll learn about different mobile app platforms, set up a development environment, and create a simple mobile app project. Understanding the basics of mobile app development will be valuable as you explore more advanced mobile app concepts and technologies in the course.

## WEEK 9

### Lab Activity: Introduction to React Native

**Objective:** The objective of this lab is to introduce students to React Native, a popular framework for building mobile applications using JavaScript and React.

**Materials Needed:**

1. Computer with Internet access
2. A text editor or Integrated Development Environment (IDE)
3. Node.js and npm (Node Package Manager) installed
4. A mobile device (iOS or Android) for testing (optional but recommended)

**Duration:** Self-paced

**Instructions:**

**Part 1: Introduction to React Native**

**1. What is React Native?:**

- ☐ Start by researching what React Native is and why it's essential in modern mobile app development. You can find introductory articles and videos online.

**Part 2: Setting Up Your React Native Environment**

**1. Installing React Native:**

- ☐ Open your terminal or command prompt.
- ☐ Use the following command to install React Native globally on your system:

**`npm install -g react-native`**

**2. Creating a New React Native Project:**

- ☐ Create a new React Native project using the following command. Replace MyReactNativeApp with your preferred project name.

**`npx react-native init MyReactNativeApp`**

**Part 3: Building Your First React Native App**

**1. Exploring the Project Structure:**

- ☐ Familiarize yourself with the structure of a React Native project, including directories like **Android**, **iOS**, **src**, and **node\_modules**.

**2. Running Your App:**

- ☐ Use your development environment to run your React Native app on an emulator or a physical device (if available). Follow platform-specific instructions for running your app (e.g., Android Studio for Android, Xcode for iOS).

### **3. Understanding the App Entry Point:**

- ☐ Explore the app's entry point, usually located in the `index.js` file. Understand how the app is initialized and how components are rendered.

## **Part 4: Creating React Native Components**

### **1. Introduction to React Native Components:**

- ☐ Learn about React Native components, which are similar to React components but designed for mobile development.

### **2. Creating a Simple Component:**

- ☐ Create a new React Native component that displays a basic UI element (e.g., text, image, button) on the screen.

## **Part 5: Styling in React Native**

### **1. Styling React Native Components:**

- ☐ Understand how to apply styles to React Native components using JavaScript and the StyleSheet API.

### **2. Styling Your Component:**

- ☐ Style the component you created earlier using the StyleSheet API. Experiment with different styles and layouts.

## **Part 6: Recap and Assignment**

### **1. Recap and Assignment:**

- ☐ Summarize what you've learned about React Native in this lab.
- ☐ As an assignment, enhance your React Native app by adding more components and interactivity. Document your app's features, design choices, and any challenges you faced during development.

## **Conclusion:**

This self-paced lab activity provides you with a foundational understanding of React Native, a powerful framework for building mobile applications. You'll learn how to set up a React Native environment, create a simple mobile app, and work with React Native components and styling. This knowledge will serve as a solid foundation for your further exploration of React Native and mobile app development in the course.

# QUARTER 4

## WEEK 10 - EXTRA

### Lab Activity: Security and Deployment of Web Applications

**Objective:** The objective of this lab is to explore security practices and deployment strategies for web applications. Students will learn how to secure their web applications and deploy them to production environments.

**Materials Needed:**

1. Computer with Internet access
2. A text editor or Integrated Development Environment (IDE)
3. Node.js and npm (Node Package Manager) installed
4. Access to a web browser
5. GitHub account (for deployment)

**Duration:** Self-paced

**Instructions:**

**Part 1: Security Best Practices**

**1. Introduction to Web Application Security:**

- ☐ Research and understand the importance of web application security, including common threats and vulnerabilities.

**2. Authentication and Authorization:**

- ☐ Learn the difference between authentication and authorization and explore various methods to implement them in your web application.

**3. Data Validation and Sanitization:**

- ☐ Understand the significance of data validation and sanitization in preventing common security vulnerabilities like SQL injection and Cross-Site Scripting (XSS).

**Part 2: Secure Coding Practices**

**1. Secure Coding Guidelines:**

- ☐ Study secure coding guidelines for the programming languages and frameworks used in your web application.

**2. Code Review and Vulnerability Scanning:**

- ☐ Learn how to perform code reviews and use automated vulnerability scanning tools to identify and fix security issues in your code.

### **Part 3: Deployment Strategies**

#### **1. Introduction to Deployment:**

- ☐ Understand the deployment process and the importance of a well-structured deployment strategy.

#### **2. Version Control with Git:**

- ☐ Review the basics of version control with Git, as it's essential for deploying web applications.

### **Part 4: Deployment Tools**

#### **1. Deployment Platforms:**

- ☐ Explore different deployment platforms, including traditional web hosting, cloud services (e.g., AWS, Azure, Heroku), and serverless options.

#### **2. CI/CD Pipelines:**

- ☐ Learn about Continuous Integration (CI) and Continuous Deployment (CD) pipelines and how they automate the deployment process.

### **Part 5: Deployment and Security**

#### **1. SSL/TLS Certificates:**

- ☐ Understand the importance of SSL/TLS certificates for securing data transmission and enabling HTTPS.

#### **2. Web Application Firewall (WAF):**

- ☐ Learn about Web Application Firewalls and how they protect your web application from common security threats.

### **Part 6: Deploying Your Web Application**

#### **1. Prepare Your Application:**

- ☐ Ensure your web application is ready for deployment. This includes resolving any security issues, optimizing performance, and setting up environment variables.

#### **2. Deployment Options:**

- ☐ Choose a deployment option based on your project's requirements (e.g., traditional hosting, cloud platform, serverless).

#### **3. Setting Up CI/CD:**

- ☐ If applicable, set up a CI/CD pipeline to automate the deployment process whenever changes are pushed to your code repository.

## **Part 7: Recap and Assignment**

### **1. Recap and Assignment:**

- ☐ Summarize what you've learned about web application security and deployment best practices in this lab.
- ☐ As an assignment, secure and deploy a simple web application (such as a portfolio site or blog) following the best practices you've learned. Document your deployment process and any challenges you encountered.

### **Conclusion:**

This self-paced lab activity provides you with an understanding of essential web application security practices and deployment strategies. You've learned how to secure your web application's code, deploy it to different environments, and ensure it's protected in production. This knowledge will be valuable as you work on more complex web application projects and aim to maintain their security and availability.

**WEEK 11 - Web Application Development for Everybody**  
**Capstone**

**WEEK 12 - Mobile Application Development for Everybody**  
**Capstone**