PROJECT REPORT

Project Title:

Create Transition Effects from White Section to Black Section Using HTML, CSS, and React

Table of Contents

Web Development Capstone Project

Project Title: Create Transition Effects from White Section to Black Section Using HTML, CSS, and React

SUBMITTED BY:

DEBADATTA ROUT

routdebadatta22@gmail.com

1. Overview of the Project

This project uses HTML, CSS, and React to show how to create a seamless transition between two sections—a white area and a black section. A button that toggles between the two parts is clicked to initiate the transition. The project's goal is to demonstrate how to utilize React to manage user interactions effectively using CSS to create fluid animations. Moreover, this project's live webpage features phrases such as "This was created by Debadatta Rout" and "Welcome to my Major Project 1".

2. Employed Technologies

The webpage is structured using HTML (Hypertext Markup Language), which also defines the root container that React components will be presented in.

Cascading Style Sheets, or CSS, are used to style the information and produce the effect of a transition from the white to the black part.

React is a JavaScript library that is used to create the webpage's interactive element. Based on human interaction, it is utilized to dynamically switch between the black and white areas.

3. Explaining Folder Structure

public/: Holds any additional publicly accessible assets as well as the static HTML file index.html.

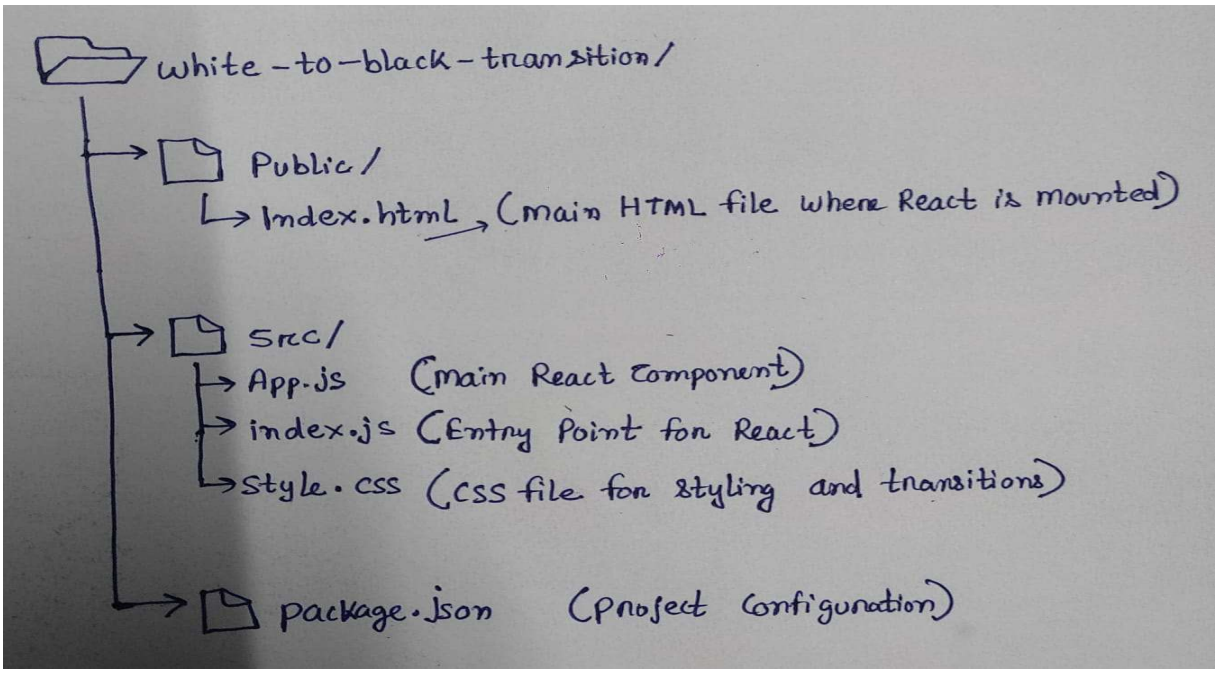
All of the JavaScript, CSS, and React components are located in the src/ directory.

App.js: The primary React component that manages the webpage's logic, including rendering the user interface and managing states.

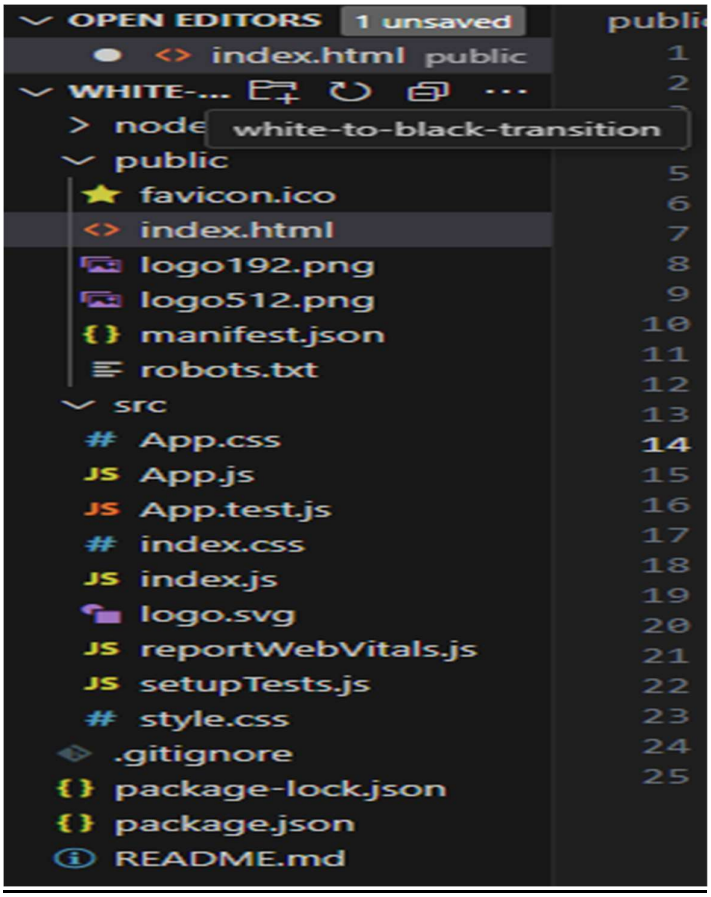
All of the webpage's styles and transition effects are contained in style.css.

The React application's entry point, index.js, renders the root React component into the index.html file.

The project folder structure is as follows:



IN VS CODE



Important Ideas and Phrases

The webpage is structured using **HTML**. The React app is rendered in the container ( ) defined by the HTML code in index.html of this project.

CSS

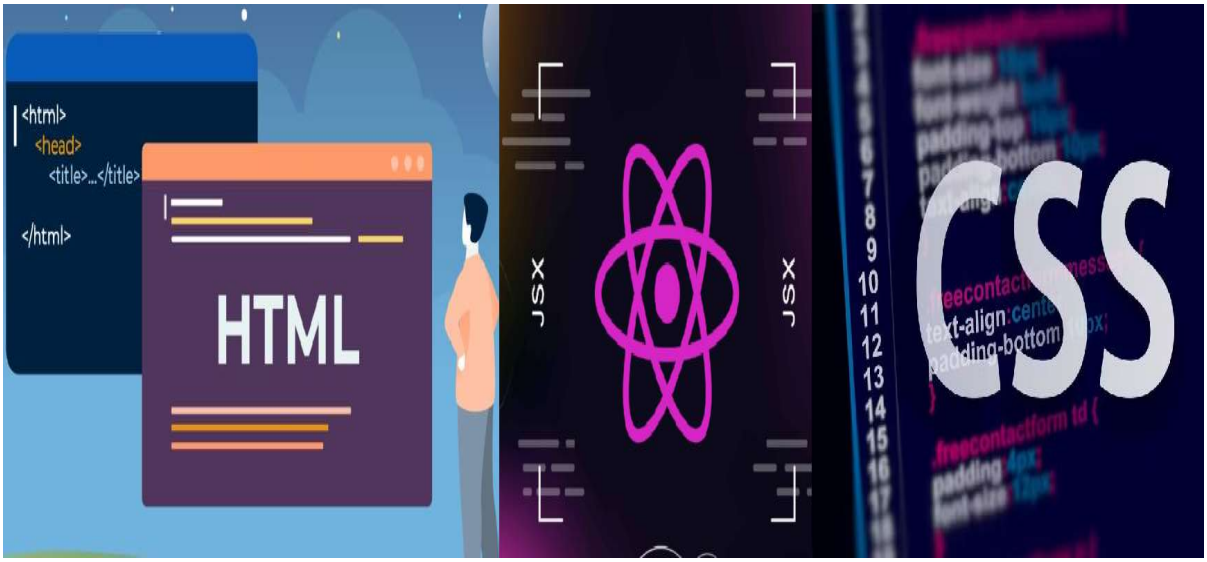
The webpage is styled and the seamless transition effect is produced using CSS. The CSS transition attribute animates the background color change over a one-second time interval.

React

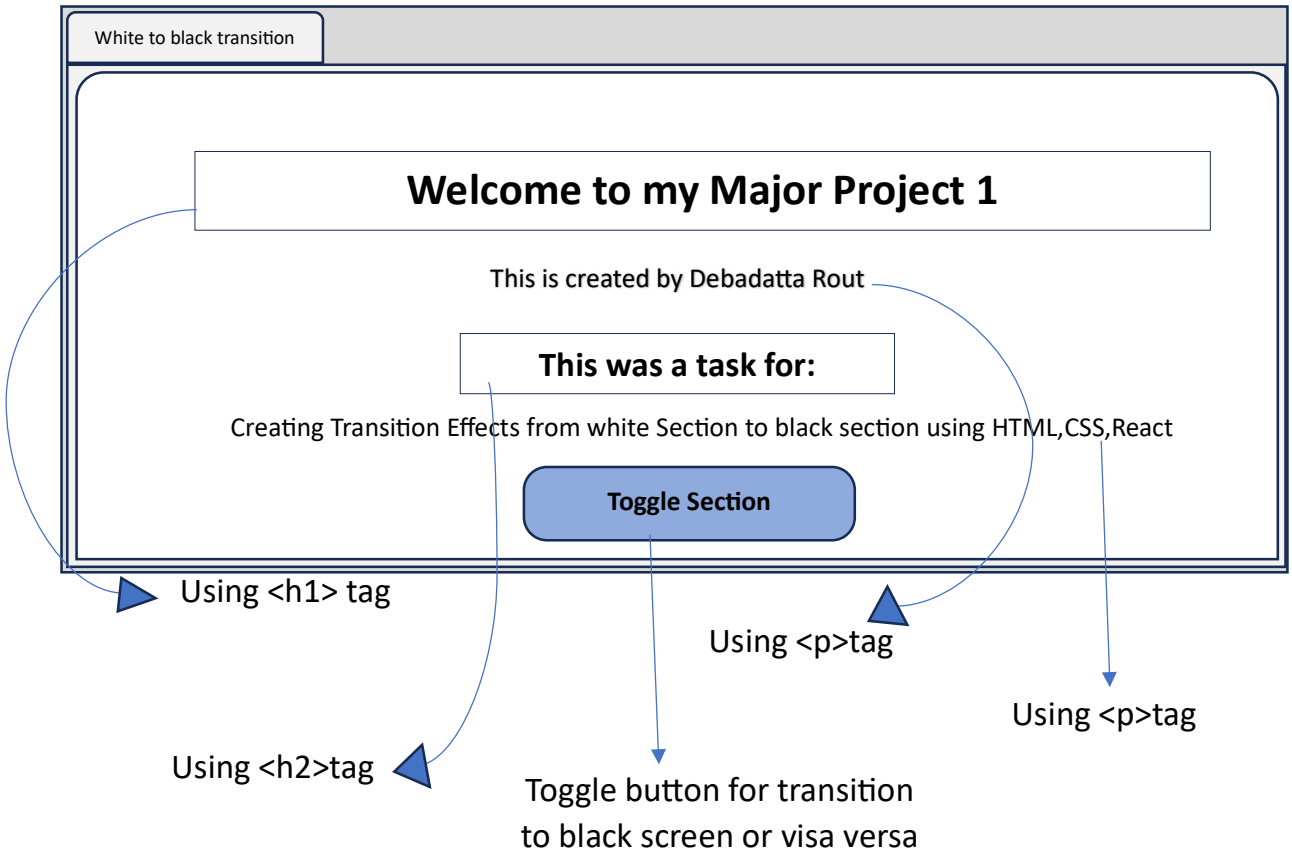
React is a JavaScript package used for constructing interactive UIs. Within this undertaking:

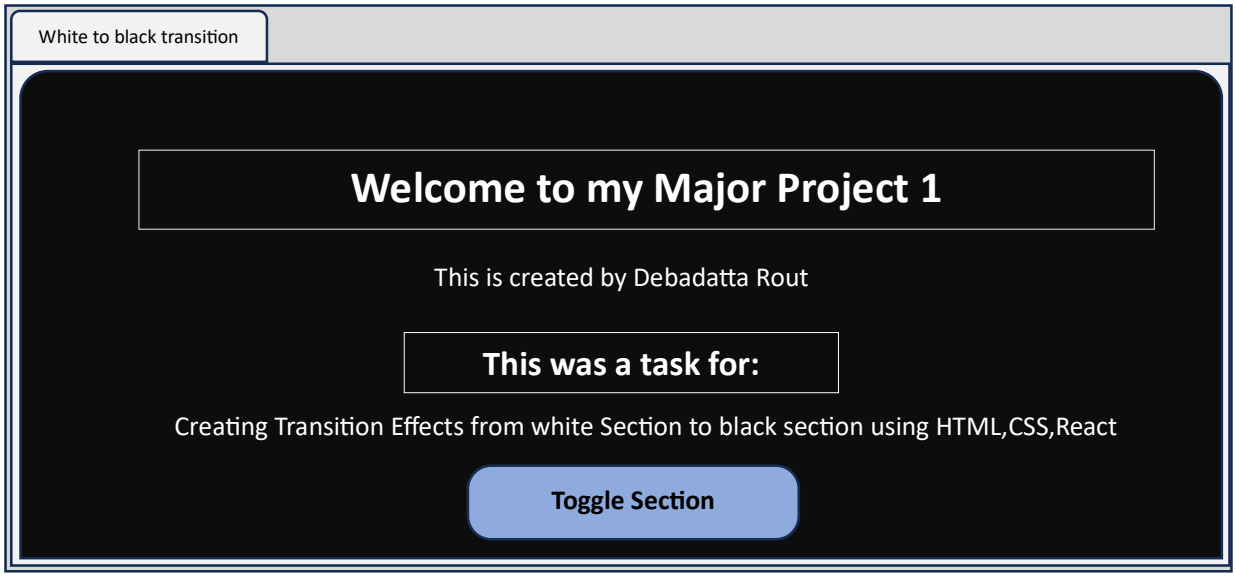
useState Hook: Controls which section (black or white) is active at any given time.

Event Handling: Toggling between the parts is accomplished by using React's onClick event handler.



My webpage design and transition plan:





TAGS/CODES

index.html in public

index.html

public > index.html > html > head

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1" />
    <meta name="theme-color" content="#000000" />
    <meta
      name="description"
      content="Web site created using create-react-app"
    />
    <link rel="icon" href="%PUBLIC_URL%/favicon.ico" />
    <link rel="apple-touch-icon" href="%PUBLIC_URL%/logo192.png" />
    <link rel="manifest" href="%PUBLIC_URL%/manifest.json" />
      <title>White to Black Transition</title>
  </head>
  <body>
    <noscript>You need to enable JavaScript to run this app.</noscript>

    <!-- This is the root div where React will mount the app -->
    <div id="root"></div>

    <!-- Your build process will place the bundled JS here -->
  </body>
</html>

```

EXPLANATION :

Any React project must include the index.html file since it provides the fundamental framework for the webpage. It is essential to the browser's ability to load the React application. These are the essential components:

:

The complete React app will be mounted in this primary container. This section of the HTML text serves as the root element for React, and it is where all React components are injected. This is accomplished by using React's ReactDOM.render() function through the index.js file..

Metadata:

Meta tags are mostly used by search engines and browsers to offer info about the HTML text. Within this document:

The character encoding for the webpage is defined by <meta charset="utf-8" />, which guarantees that special characters appear appropriately.

The element <meta name="viewport" with content="width=device-width, initial-scale=1" opens.Makes sure the website is responsive, meaning it can adjust to various screen sizes and device kinds (like desktops and mobile phones).

Title:

The transition from white to black. This configures the text that displays on the webpage tab of the browser, providing information about the page's content. Here, it alerts readers to the presence of a "White to Black Transition" on the page.

App.js (in the src/ folder)

```
  return (
    <div className={isWhiteSection ? 'white-section' : 'black-section'}>
      <div className="content">
        <h1>Welcome to my  Major Project 1 </h1>
        <p>This was created by Debadatta Rout</p>
        <br></br>
        <h2>This was a task for : </h2>
        <p>Creating Transition Effects from White Section to Black Section Using HTML, CSS, and React</p>

        <button className="transition-button" onClick={handleToggle}>
          Toggle Section
        </button>
      </div>
    </div>
  );
}

export default App;
```

EXPLANATION :

The App.js file is the main React component that handles the logic of toggling between the white and black sections. It includes:

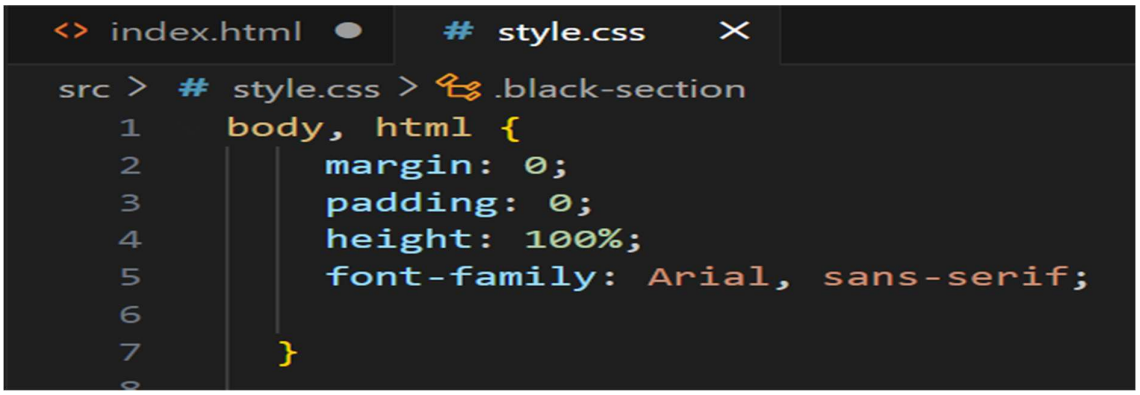
React state management: Using the useState hook to store whether the current section is white or black.

Event handling: The button triggers the handleToggle function, which changes the section color.

Conditional Rendering: Depending on the state, the section's class switches between white-section and black-section.

---

style.css (in the src/ folder)



index.html ● # style.css ×

src > # style.css > .black-section

```css
.white-section, .black-section {
  display: flex;
  justify-content: center;
  align-items: center;
  height: 100vh;
  transition: background-color 1s ease-in-out;
}

.white-section {
  background-color: white;
  color: black;  /* Text color for white section */


}

.black-section {
  background-color: black;
  color: white;  /* Text color for black section */
}

.content {
  text-align: center;
}

.transition-button {
  padding: 10px 20px;
  font-size: 18px;
  border: none;
  border-radius: 5px;
  cursor: pointer;
  color: white;
  background-color: #007BFF;
  transition: background-color 0.3s ease;
  margin-top: 20px;
}

.transition-button:hover {
  background-color: #0056b3;
}
```

EXPLANATION:

The layout, effects, and transitions on the webpage are all controlled by the style.css file. Below is a summary of the main ideas that are discussed in the file:

Flexbox Configuration:

Elements inside a container can be aligned using the Flexbox CSS layout concept. Here, Flexbox is used to center the content on the page in both the vertical and horizontal directions.

This guarantees that the button and text will always be in the middle of the page, no matter how big the screen gets.

justify-content: center; the content is centered horizontally.

align-items: center aligns the content vertically.

height: 100vh; this causes the section to fill the entire viewport.

Easy Transition:

Smooth transitions between an element's many states are made possible using the CSS transition attribute. Here, it's employed to produce a seamless transition in background color between the black and white areas.

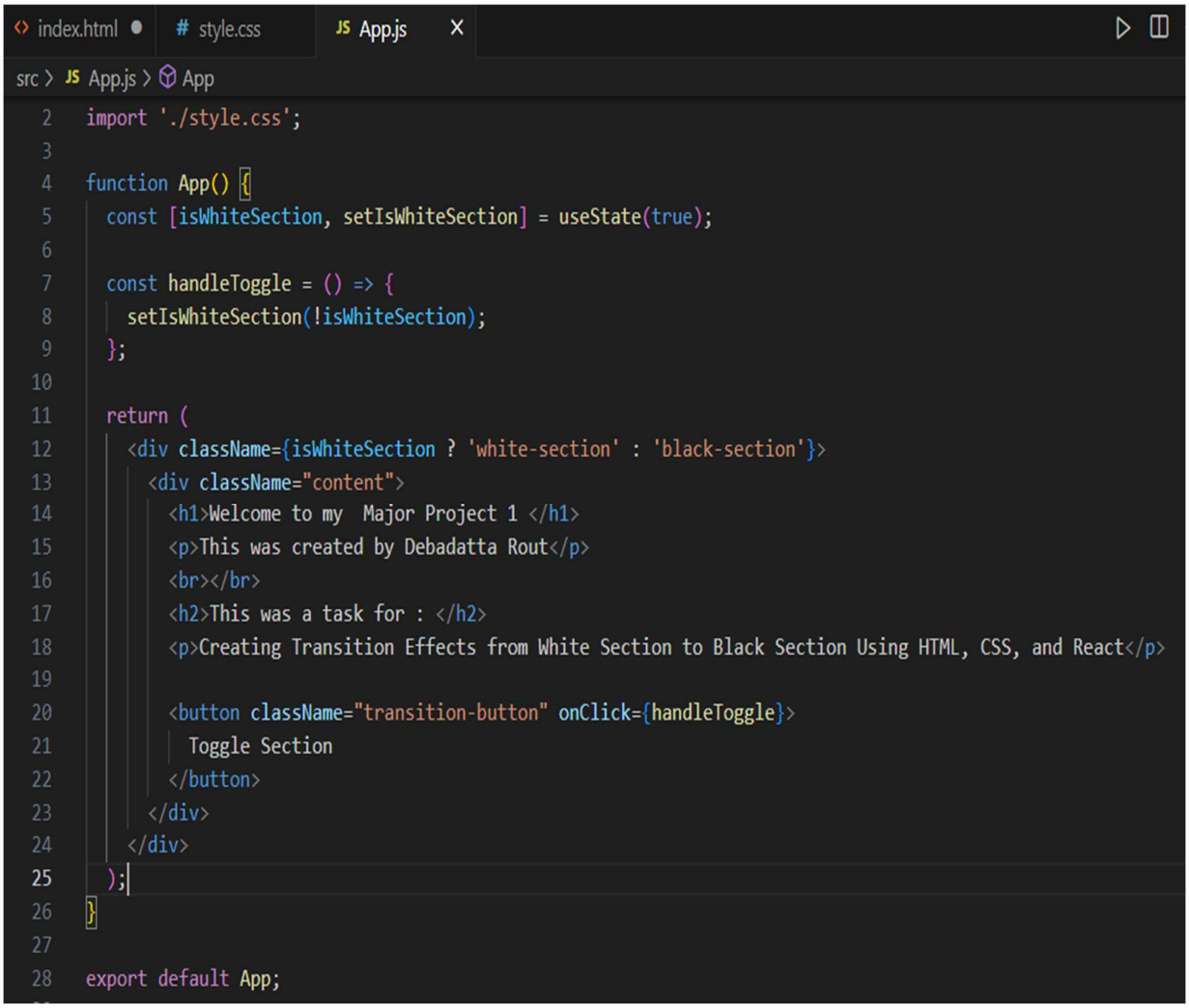
The backdrop gradually transitions from white to black over a predetermined amount of time rather than changing colors all at once.

transition: background-color 1s easy; specifies a background color transition that is gradual over a duration of one second.

Effect of Button Hover:

To improve user interaction, a hover effect is incorporated into the button style. The backdrop color shifts when the user hovers over the button, offering visual cues and enhancing the user experience.

App.js



EXPLANATION:

The core of your React application is the App.js file. It integrates the transition effects, maintains the layout, and handles the state. The main components of the code are broken down as follows:

1. IMPORT

To use JSX and React features, import the React library.

To manage state within the functional component, import the useState hook.

To apply the required styles, import the style.css file.

2. State Administration:

The useState hook is used to define a state variable called isWhiteSection, which is initialized to true. The current section's color—white or black—is determined by this variable.

This state is updated by the function setIsWhiteSection.

3.Toggle Function:

The handleToggle function is defined to change the state when the button is clicked. It toggles the value of isWhiteSection between true and false.

4.Render Method: A div that conditionally applies a CSS class depending on the isWhiteSection state is returned by the main component:

The class white-section, which has a white backdrop, is used if isWhiteSection is true.

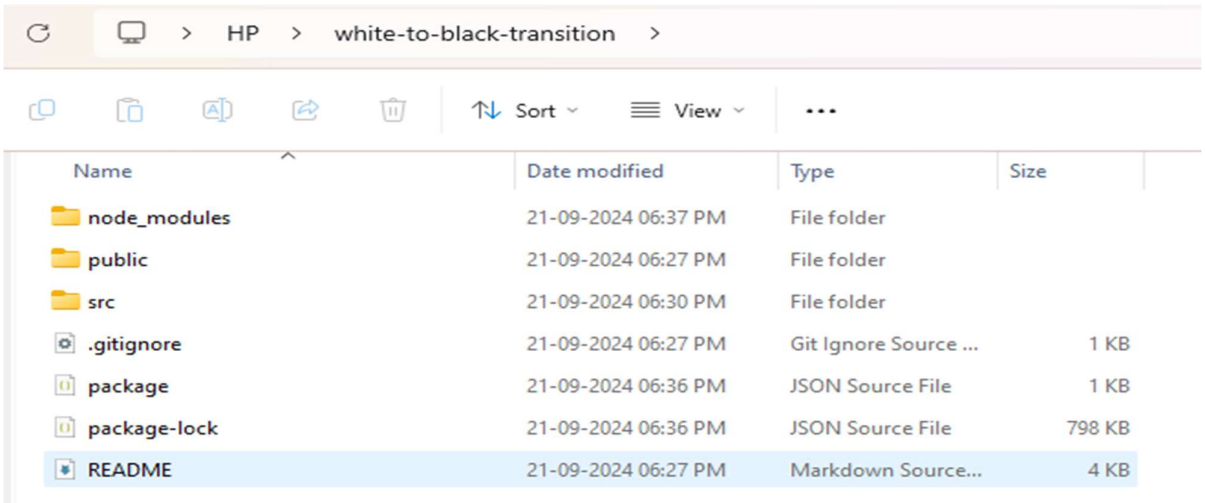
The background is changed to black using the class black-section if isWhiteSection is false.

Another div with the class content, which includes the button, paragraphs, and heading, is contained inside this one.

Users can flip between the white and black areas by clicking the button, which activates the handleToggle function.

5.Exporting the Part:

In order to use the App component in other areas of the program, particularly index.js where it will be rendered into the DOM, this line exports it.



EXECUTION:

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS C:\Users\HP> npx create-react-app white-to-black-transition
>> cd white-to-black-transition

Creating a new React app in C:\Users\HP\white-to-black-transition.

Installing packages. This might take a couple of minutes.
Installing react, react-dom, and react-scripts with cra-template...


added 1480 packages in 1m
```

```
added 150 packages, and audited 1693 packages in 11s
Compiled successfully!

You can now view white-to-black-transition in the browser.

  Local:            http://localhost:3000
  On Your Network:  http://192.168.0.100:3000

Note that the development build is not optimized.
To create a production build, use npm run build.

webpack compiled successfully
```

EXPLANATION

Procedure Flow

First Load: React renders the App component when the application loads for the first time in the browser.

The state variable isWhiteSection is initialized to true by the useState hook, indicating that the first part will be white.

Rendering: App.js's JSX code assesses isWhiteSection's status.

The component applies the white-section CSS class to the outer div since isWhiteSection is true.

As a result, the welcome message and toggle button appear on the screen against a white background.

User Interaction: The handleToggle function is called when the user selects the "Toggle Section" button.

SetIsWhiteSection(!isWhiteSection) within this function sets the status of isWhiteSection to false, indicating that the black part should be displayed in the subsequent render.

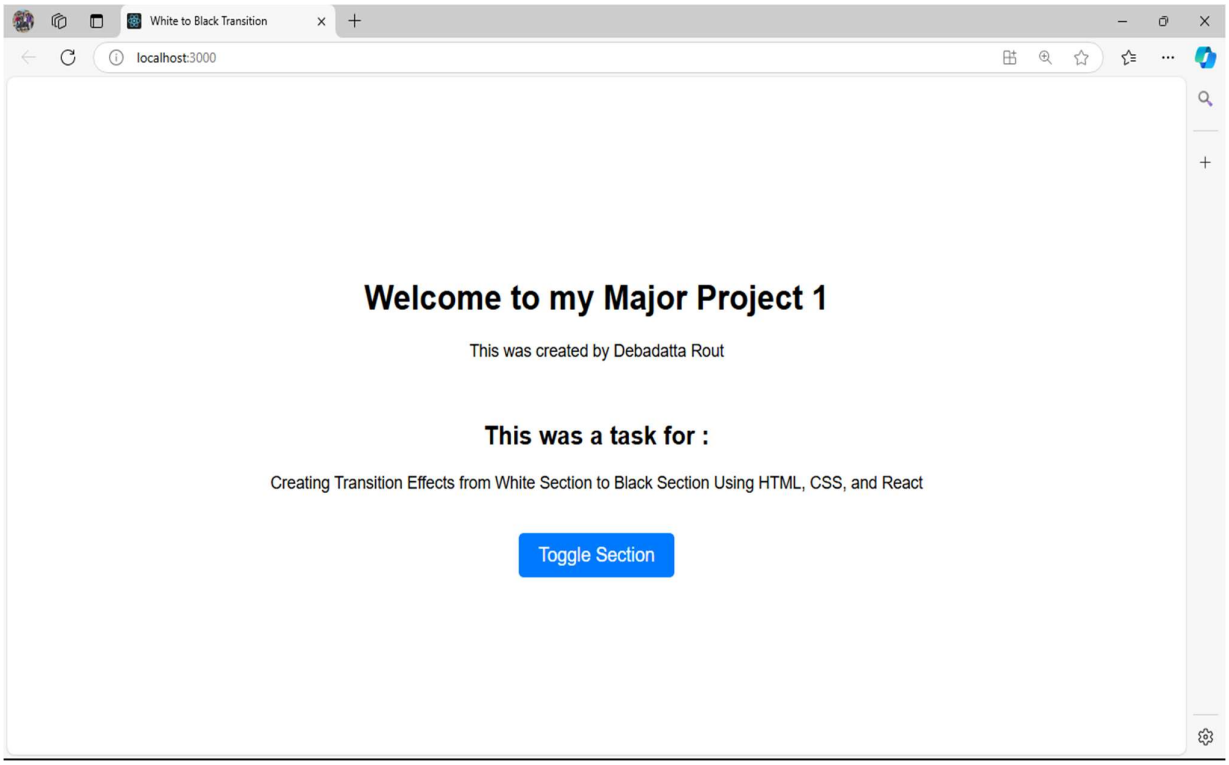
Easy Transition:

React renders the component again when the state changes. This time, the black-section CSS class is applied because isWhiteSection is false.

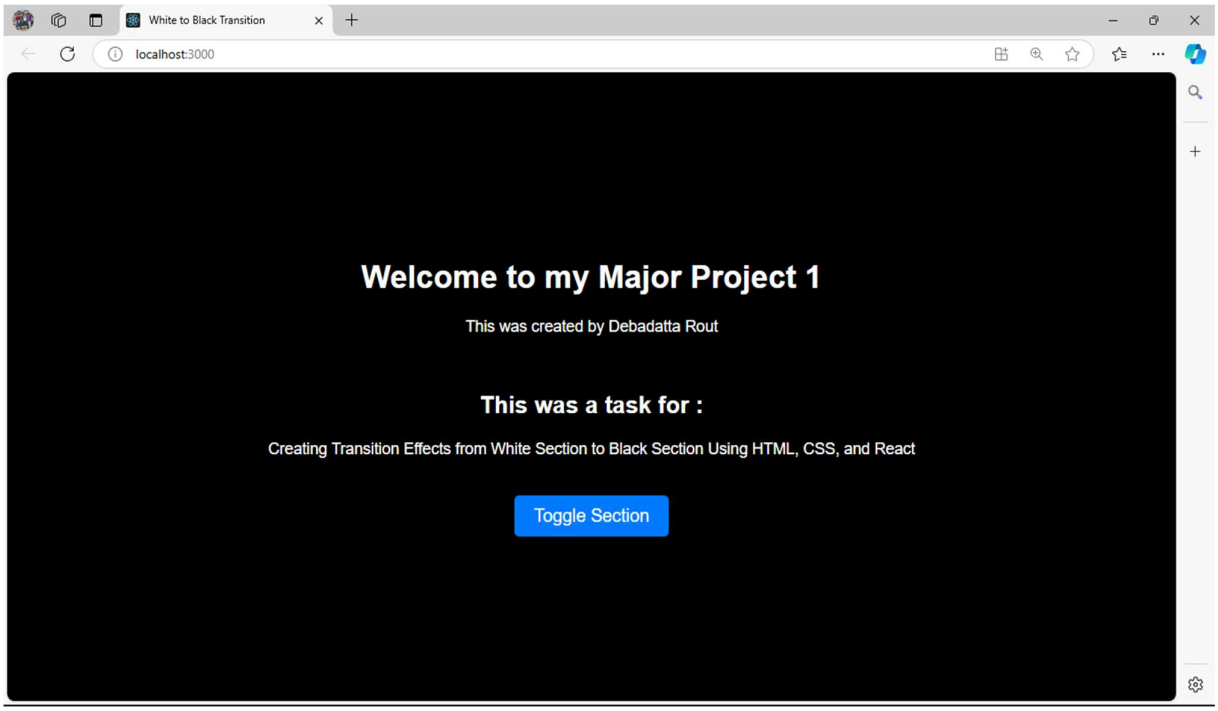
The style defines the transition attribute.The smooth transition from white to black background color over the designated length (e.g., 1 second) is guaranteed by the css file.

The consumer is thus treated to an eye-catching fading effect between the two portions.

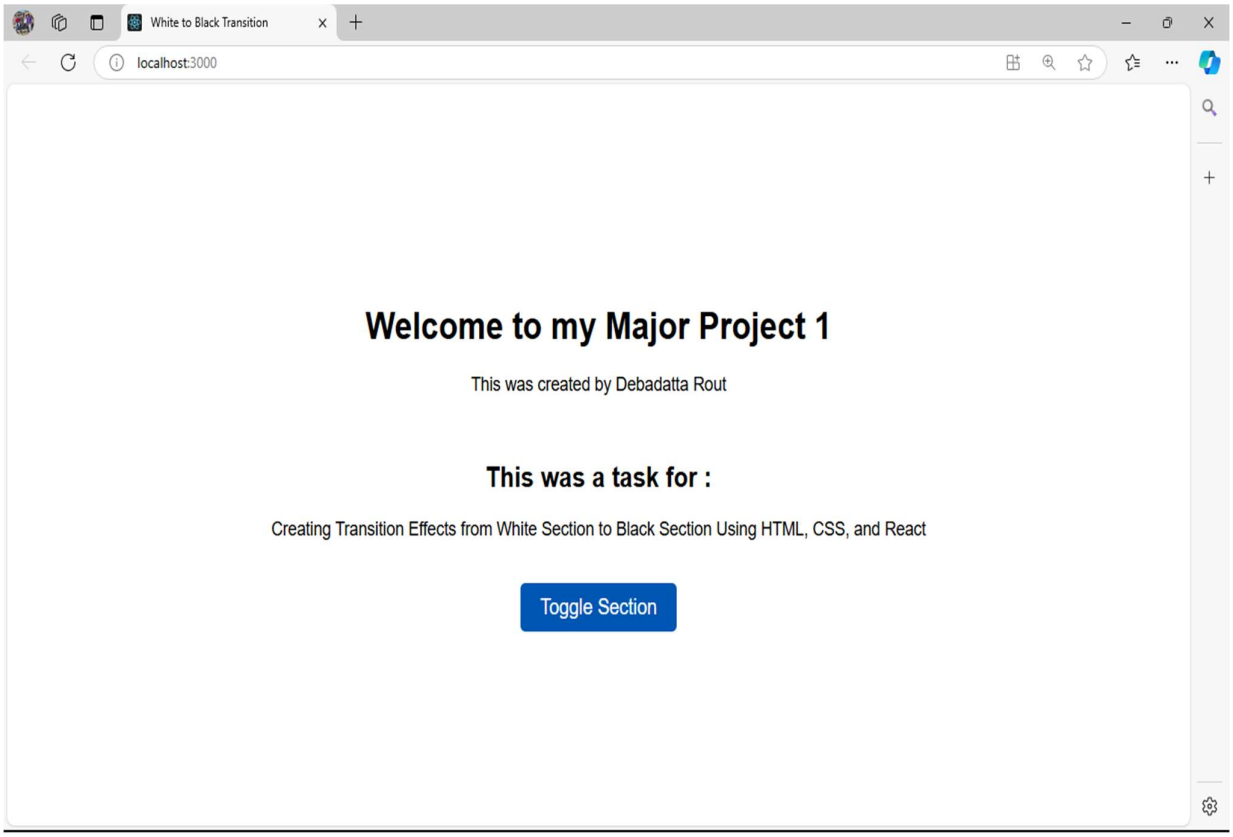
LIVE SERVER WEBPAGE



ON CLICKING ON “TOGGLE SECTION” BUTTON



AGAIN ON CLICKING ON IT



CONCLUSION

My name is Debadatta Rout, and I work with Technook as an intern in web development. I've successfully finished Major Project 1, which involved using HTML, CSS, and React to create a seamless transition effect from a white portion to a black section. This project shows off my proficiency with CSS to create aesthetically pleasing transitions in addition to my ability to handle dynamic interactions using state management in React. Through this effort, I have effectively merged these technologies to construct a dynamic and engaging online application, demonstrating the skills and knowledge I have gained during my internship.

Bibliography

Software used **- Vs code**

Course referred **– teachnook internship**