Milestone - 10 **Frontend Development** **Codrel**

Implementation of Hooks

1. **useState**: useState is used to manage state within functional components in React. It allows you to declare and update component-level state variables.

Task: Create a simple component to increment or decrement count using display using useState hook.

2. useEffect: useEffect is used to perform side effects in functional components. It can be used for tasks like data fetching, DOM manipulation, and setting up subscriptions.

Task: Build a weather app that fetches weather data from an API and displays it in a user-friendly manner using useEffect hook.

3. useContext: useContext provides a way to pass data through the component tree without having to pass props down manually at every level. It's often used for themes, authentication, or user preferences.

Task: Create a dark mode toggler to manage website theme

4. useReducer: useReducer is a React hook that is used for managing more complex state logic where state transitions depend on the previous state. It is often used to manage complex component logic

Task: Create a form in order to validate and change state using useReducer hook.

5. useMemo: useMemo is a React hook used for memorization. It's useful when you want to optimize the performance of your application by memorizing the result of expensive calculations or operations.

Task: Create a factorial function to input a value and calculate factorial with memorization.

6. useCallback: useCallback is used to memorize functions, preventing unnecessary re-renders when passing functions as props to child components.

Task: Increment and decrement a value by useCallback hook.

7.useLocalStorage: A custom hook that simplifies reading and writing data to the browser's local storage. It's useful for persisting user settings or data between sessions.

Task: Create a form where users can input information and use the hook to save and retrieve data.

8.useOnlineStatus: A custom hook that tracks the user's online/offline status. This can be helpful for handling offline functionality in web applications.

Task: Check the status of user if online then display a green background else display red background

9.useHistory: This is a popular library for handling routing and navigation in React applications, used to store browsers navigation history into a stack and navigate to different routes.

Task: Use the hook to navigate to an URL by clicking on a button.

10.useParams: This hook can be used to access the URL parameters of a Route. In web applications, URL parameters are often the part of the URL that references the specific information/resource that is to be fetched when the page is to be rendered**.**

Task: use BrowserRouter, Route and Switch to provide routing to different pages and use a route to provide params in URL on routing.

Create another component where on routing to a specific URL with params, provides the value of param on the webpage.

Milestone - 10 **Backend Development** **Codrel**

File Conversion Project

In this assignment, you will design and implement a Node.js backend application that provides users with the capability to upload files of different formats, convert these files into other formats, and download the converted files. The project emphasizes the essential concepts of file management for processing and storage and the integration of external APIs for file format conversion.

Requirements

Your backend project must meet the following requirements:

File Upload:

Create an API endpoint that enables users to upload a variety of files. Supported file types should include images (e.g., JPEG, PNG) and textual documents (e.g., DOCX, PDF, TXT).

File Conversion:

Implement the ability to convert uploaded files from one format to another. Users should specify their desired output format.

Supported conversions may include image format conversion (e.g., from PNG to JPEG) and textual document format conversion (e.g., from DOCX to PDF or TXT).

API Integration for Conversion:

Integrate external APIs or libraries for file format conversion. You are encouraged to use the following **APIs and libraries:**

Image Conversion API: Utilize the CloudConvert API (https://cloudconvert.com/api) for image format conversions.

Textual Document Conversion: Implement document format conversion using a library like mammoth.js for DOCX to HTML or use a PDF generation library for DOCX to PDF.

File Storage:

Store both the uploaded files and the converted files on your server.

Maintain a record of each file in a database, including information such as the original filename, MIME type, size, and a unique identifier.

File Download:

Allow users to download the converted files. Implement an API endpoint for downloading files, such as /download/:fileId.

The endpoint should retrieve file metadata and serve the file to the client with the original filename, prompting the user to download it.

<u>Implementation:</u>

The implementation should consider robust file management, safe storage of user data, and secure API integration. Additionally, you should ensure proper error handling and validation throughout the project.

User Authentication in Task Management System

Task Management User Registration and Login:

User Registration and Login:

Implement user registration functionality, allowing users to create an account with a unique username and password.

Implement user login functionality, verifying the provided credentials and generating a session upon successful authentication.

Authentication Middleware:

Create an authentication middleware that checks for a valid session and verifies the user's identity for protected routes.

Implement this middleware on routes that require authentication, such as updating or deleting tasks.

Authorization Checks:

Enhance the authentication middleware to include authorization checks for specific user roles or permissions.

Implement authorization checks to ensure that only the user who created a task can perform operations like updating or deleting it.

Session Management:

Implement secure session management using cookies and the express-session middleware.

Configure session options, such as secret, cookie settings, and session storage.

Utilize session-based authentication to persist user sessions and handle user authentication across multiple requests.

User Profile Page:

Create a user profile page route where users can view and edit their profile information.

Implement functionality to retrieve and update user-specific data, such as their name, email, or profile picture or you can use a default image.