DIT 637 Smart and Secure Systems
TT01: Experiencing Cloud Computing – Full-Stack Web Application
6/14/2024 Developed by Clark Ngo
6/19/2024 Reviewed by Sam Chung
07/30/2025 Reviewed by Rajdeep Singh Golan
School of Technology & Computing (STC) @ City University of Seattle (CityU)

Key Concepts and Tools for Web Development

- **Frontend**: To create the visual part of websites and applications that users interact with, like the layout of an e-commerce site.
- **React**: To efficiently build dynamic and responsive user interfaces, such as the interactive components of Facebook.
- **Movie Search**: To practice and demonstrate the capabilities of React in creating real-world applications, like searching for movies on Netflix.
- **Codespaces**: To provide a cloud-based development environment that's easy to set up and use from anywhere, similar to Google Docs but for coding.
- **GitHub as Repo**: To store and manage code with version control, making collaboration easier, just like using Google Drive for sharing documents.

1) User Case – Movie List Display & Search

💡 *A user story helps by clearly defining the user's needs and goals, ensuring that development efforts are focused on delivering value to the end-user.*

As a movie enthusiast, **I want to** search and browse a list of movies with details such as title, genre, and year, **so that I can** easily find information about movies I am interested in.

Features Included

1. **Movie List Display**:
   - **What**: Shows a list of movies with their titles, genres, and release years.
   - **Why**: To provide users with a comprehensive view of available movies.
2. **Search Functionality**:
   - **What**: An input field that filters the movie list based on the user's search term.
   - **Why**: Allows users to quickly find specific movies by typing part of the title.

Flow of Interaction

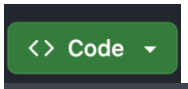
3. **Initial View**:
   - The user sees a list of all available movies along with a search bar at the top.
4. **Using the Search Bar**:
   - As the user types into the search bar, the list of movies dynamically updates to show only those that match the search term.

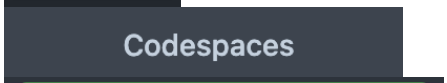
Technical Implementation

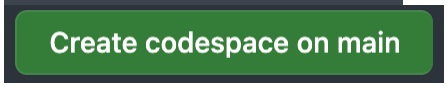
- **State Management**: Uses React's useState hook to manage the search term and filtered movie list.
- **Search Handling**: Updates the displayed movie list in real-time based on user input.
- **Styling**: Utilizes Material-UI components and custom styles for a clean and responsive user interface.

This user story ensures that users have a seamless and interactive experience when searching and discovering movies in the application.

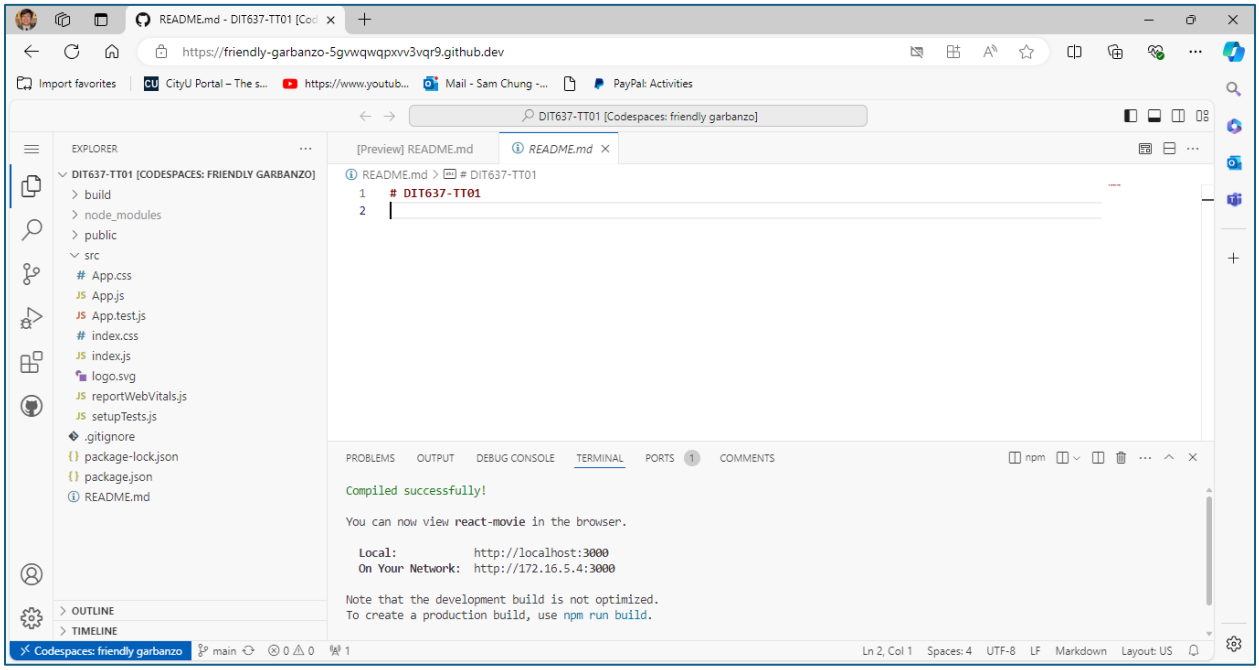
2) Setup

1. Open the project in GitHub.
2. Create a cloud-based development project with GitHub Codespaces:
   a. 

   b. 

   c. 

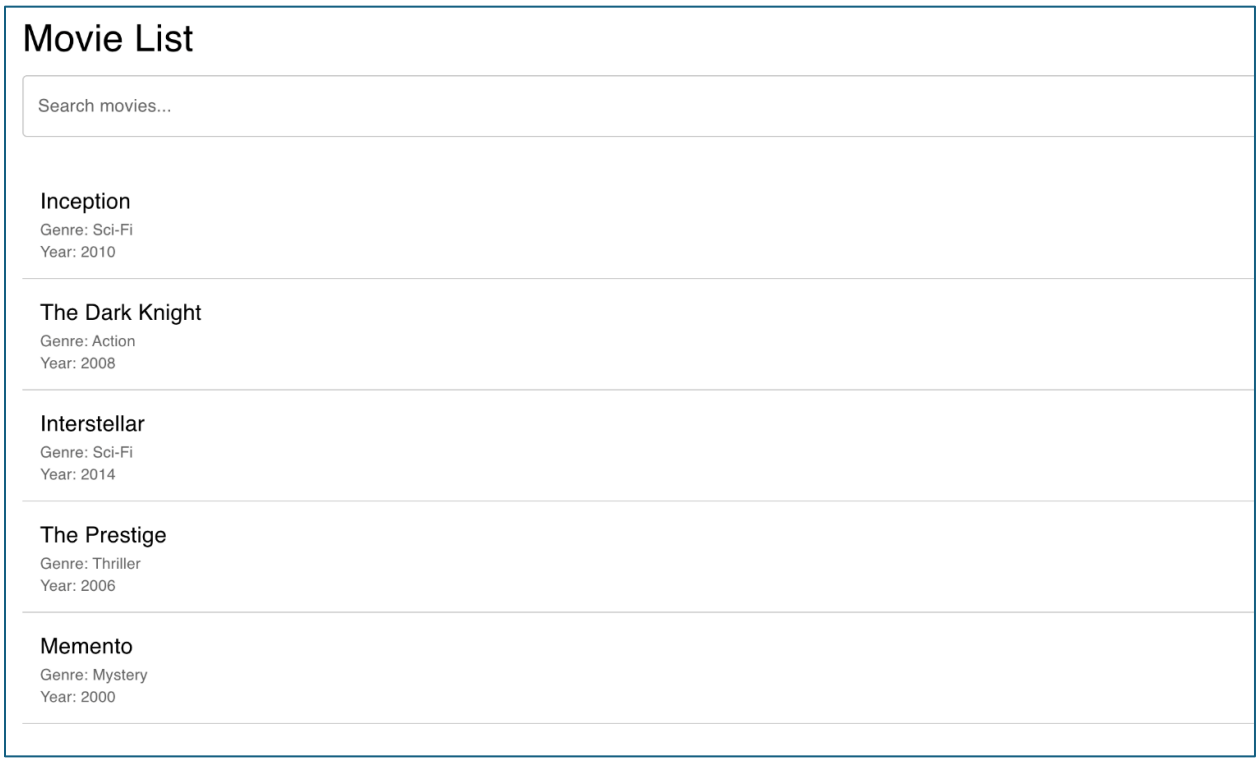
3. In the terminal type:
   a. npm install
   b. npm start
4. It should automatically open a new browser (If prompted, click “Retry”)



3) Screenshot Summary

In the provided screenshot, we see:

1. **Movie List Title**: Displayed using Typography with the variant "h4".
2. **Search Bar**: A TextField at the top for users to type in their search queries.
3. **Movie Items**: Each movie is displayed in a ListItem, showing the title, genre, and year.
4. **Separation**: Each movie item is visually separated by a border, providing a clean layout.



Take the screenshot of your 'React App' as 'firstname_lastname_react_app.png' by using your first and last name and upload/push to current GitHub repository.

4) Components Overview

1. **App Component**:
   - The main component that contains the entire application structure.
   - It maintains the state for the movie list and search term.
   - It renders the movie list and search input.
2. **Container (Material-UI)**:
   - Wraps the entire content to provide consistent spacing and layout.
   - Styled using styled from Material-UI's system for custom styling.
3. **Typography (Material-UI)**:
   - Used for consistent text styling.
   - Here, it is used to display the title "Movie List" and movie details.
4. **TextField (Material-UI)**:
   - An input field for searching movies.
   - It updates the search term state and filters the movie list based on the input.
   - Styled using styled from Material-UI's system.
5. **List (Material-UI)**:
   - A container for the list of movies.
   - It ensures the movies are displayed as a list.
6. **ListItem (Material-UI)**:
   - Represents each movie in the list.
   - Styled using styled from Material-UI's system for custom borders and layout.
7. **ListItemText (Material-UI)**:
   - Displays the movie title, genre, and year within each ListItem.
   - Uses Typography for consistent text styling.

5) Code Overview

This code imports necessary React hooks and Material-UI components for building and styling the movie list application. The following code is present under **src/App.js**

- **React Hooks**: Enable functional components to use state and other React features, making the code more concise and easier to manage.
  - **Functional Components:** Simplified React components defined as functions, offering a more concise syntax and easier testing compared to class components.
  - **State in React:** A built-in object for managing component-specific data that can trigger re-rendering when updated, enabling dynamic and interactive UIs.

- **Material-UI**: Provides a comprehensive set of components and styling tools to create a responsive and visually appealing user interface efficiently.

```
1   import React, { useState } from 'react';
2   import { Container, TextField, List, ListItem, ListItemText, Typography } from '@mui/material';
3   import { styled } from '@mui/system';
```

Sample Data for Movies: Provides a predefined list of movies with their titles, genres, and release years, used to populate the initial movie list and demonstrate search functionality.

```
5   // Sample data for movies
6   const moviesData = [
7     { title: 'Inception', genre: 'Sci-Fi', year: 2010 },
8     { title: 'The Dark Knight', genre: 'Action', year: 2008 },
9     { title: 'Interstellar', genre: 'Sci-Fi', year: 2014 },
10    { title: 'The Prestige', genre: 'Thriller', year: 2006 },
11    { title: 'Memento', genre: 'Mystery', year: 2000 }
12  ];
```

Styled Components Using Material-UI: Enhance the appearance and layout of the React application by adding custom styles to Material-UI components, ensuring a consistent and visually appealing design.

```
14  // Styled components using Material-UI
15  const ContainerStyled = styled(Container)({
16    marginTop: '20px',
17  });
18
19  const TextFieldStyled = styled(TextField)({
20    marginBottom: '20px',
21  });
22
23  const ListItemStyled = styled(ListItem)({
24    borderBottom: '1px solid #ccc',
25  });
```

App Component: The main functional component that uses state to manage the search term and movie list, providing real-time filtering and rendering of movie data to create an interactive user interface.

```
const App = () => {
  const [searchTerm, setSearchTerm] = useState('');
  const [movies, setMovies] = useState(moviesData);

  // Function to handle search input
  const handleSearch = (event) => {
    setSearchTerm(event.target.value);
    const filteredMovies = moviesData.filter((movie) =>
      movie.title.toLowerCase().includes(event.target.value.toLowerCase())
    );
    setMovies(filteredMovies);
  };

  return (
    <ContainerStyled>
      <Typography variant="h4" gutterBottom>Movie List</Typography>
      <TextFieldStyled
        label="Search movies..."
        variant="outlined"
        fullWidth
        value={searchTerm}
        onChange={handleSearch}
      />
      <List>
        {movies.map((movie, index) => (
          <ListItemStyled key={index}>
            <ListItemText
              primary={<Typography variant="h6">{movie.title}</Typography>}
              secondary={
                <>
                  <Typography variant="body2">Genre: {movie.genre}</Typography>
                  <Typography variant="body2">Year: {movie.year}</Typography>
                </>
              }
            />
          </ListItemStyled>
        ))}
      </List>
    </ContainerStyled>
  );
};

export default App;
```

6) Breakdown:

App Component Definition:

```
27    const App = () => {
```

- **What**: Defines the main functional component of the application.
- **Why**: Serves as the entry point for rendering the user interface.

State Declarations:

```
28      const [searchTerm, setSearchTerm] = useState('');
29      const [movies, setMovies] = useState(moviesData);
```

- **What**: Declares state variables for managing the search term and movie list.
- **Why**: Enables dynamic updates and re-rendering when the search term changes or the movie list is filtered.

Handle Search Function:

```
31      // Function to handle search input
32      const handleSearch = (event) => {
33        setSearchTerm(event.target.value);
34        const filteredMovies = moviesData.filter((movie) =>
35          movie.title.toLowerCase().includes(event.target.value.toLowerCase())
36        );
37        setMovies(filteredMovies);
38      };
```

- **What**: Function to update the search term and filter the movie list based on user input.
- **Why**: Provides real-time search functionality, allowing users to find movies by title.

Render Method:

```
40    return (
41      <ContainerStyled>
42        <Typography variant="h4" gutterBottom>Movie List</Typography>
43        <TextFieldStyled
44          label="Search movies..."
45          variant="outlined"
46          fullWidth
47          value={searchTerm}
48          onChange={handleSearch}
49        />
50        <List>
51          {movies.map((movie, index) => (
52            <ListItemStyled key={index}>
53              <ListItemText
54                primary={<Typography variant="h6">{movie.title}</Typography>}
55                secondary={
56                  <>
57                    <Typography variant="body2">Genre: {movie.genre}</Typography>
58                    <Typography variant="body2">Year: {movie.year}</Typography>
59                  </>
60                }
61              />
62            </ListItemStyled>
63          ))}
64        </List>
65      </ContainerStyled>
66    );
67  };
```

- **What**: Renders the search input, movie list, and individual movie details.
- **Why**: Displays the user interface elements and ensures they update dynamically based on the search term and movie list state.

Export Default App:

```
69  export default App;
```

- **What**: Exports the App component as the default export.
- **Why**: Allows the App component to be imported and used in other parts of the application.