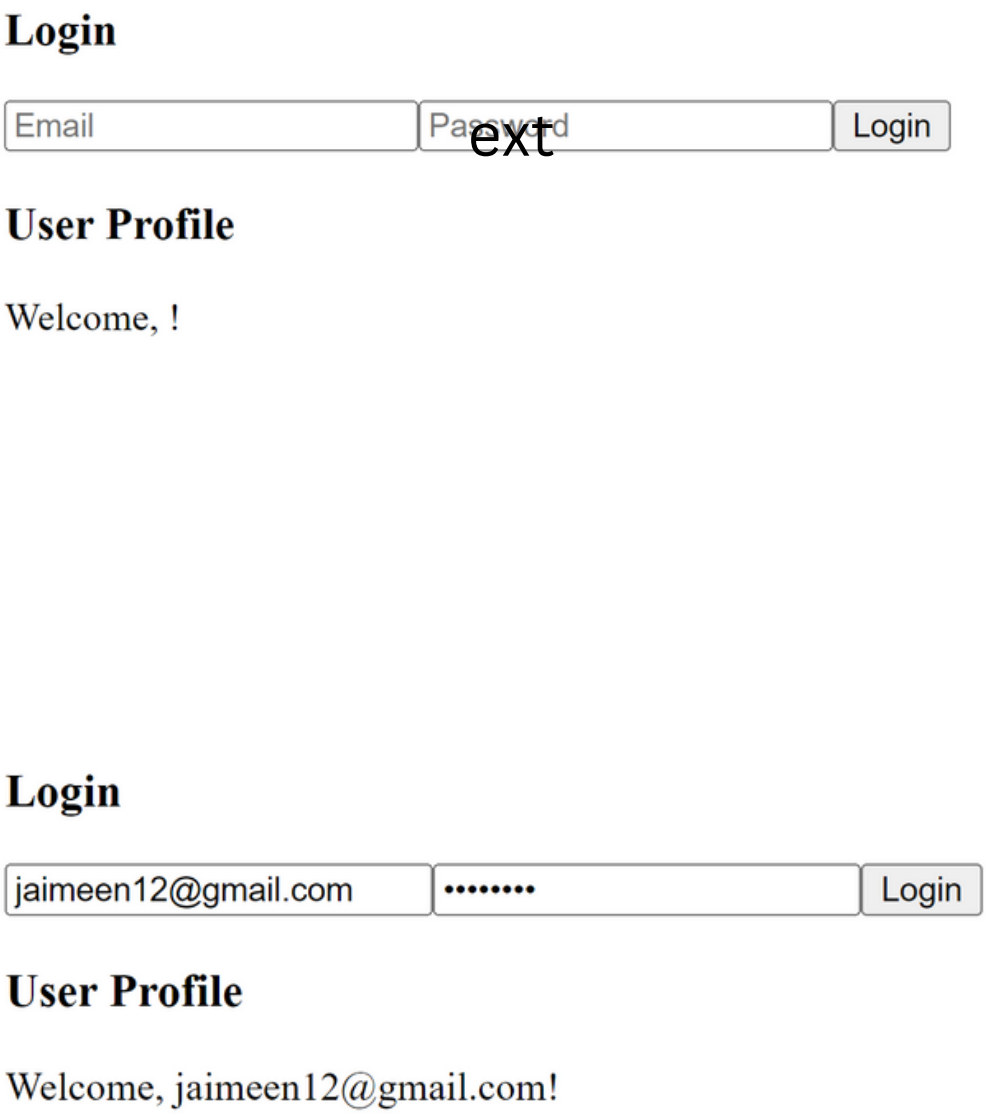
Output would look like this :



```jsx
//login-form component-1

import React, { useState } from 'react';
import { connect } from 'react-redux';//allows us to connect the component to the Redux store.
import { login } from '../redux/actions/authActions';//This action creator is responsible for dispatching(sending)
// the login action to the Redux store.

const LoginForm = ({ login }) => {
//login function is passed as a prop from the Redux store when the component is connected.
  const [email, setEmail] = useState('');
  const [password, setPassword] = useState('');
//These functions are responsible for updating the email and password state variables based on the user's input.
  const handleEmailChange = (e) => {
    setEmail(e.target.value);
  };

  const handlePasswordChange = (e) => {
    setPassword(e.target.value);
  };
//which is triggered when the form is submitted.
  const handleSubmit = (e) => {
//it prevents the default form submission behavior, which causes a page refresh.
    e.preventDefault();
// triggering the login action to be dispatched to the Redux store.
    login(email, password);
  };

  return (
    <div className="container">
      <h3>Login</h3>
      <form onSubmit={handleSubmit}>
        <input type="text" placeholder="Email" value={email} onChange={handleEmailChange} />
        <input type="password" placeholder="Password" value={password} onChange={handlePasswordChange} />
        <button type="submit">Login</button>
      </form>
    </div>
  );
};
//The LoginForm component is connected to the Redux store using the connect function from 'react-redux'.
//The first argument, null, is used to indicate that the component does not need to subscribe to any state updates from the R
//The second argument, { login }, provides the login action creator as a prop to the LoginForm component, allowing it to disp
export default connect(null, { login })(LoginForm);
```

created by:jaimeen makavana

```jsx
//login-form component-1

import React, { useState } from 'react';
import { connect } from 'react-redux';//allows us to connect the component to the Redux store.
import { login } from '../redux/actions/authActions';//This action creator is responsible for dispatching(sending)
// the login action to the Redux store.

const LoginForm = ({ login }) => {
//login function is passed as a prop from the Redux store when the component is connected.
  const [email, setEmail] = useState('');
  const [password, setPassword] = useState('');
//These functions are responsible for updating the email and password state variables based on the user's input.
  const handleEmailChange = (e) => {
    setEmail(e.target.value);
  };

  const handlePasswordChange = (e) => {
    setPassword(e.target.value);
  };
//which is triggered when the form is submitted.
  const handleSubmit = (e) => {
//it prevents the default form submission behavior, which causes a page refresh.
    e.preventDefault();
// triggering the login action to be dispatched to the Redux store.
    login(email, password);
  };

  return (
    <div className="container">
      <h3>Login</h3>
      <form onSubmit={handleSubmit}>
        <input type="text" placeholder="Email" value={email} onChange={handleEmailChange} />
        <input type="password" placeholder="Password" value={password} onChange={handlePasswordChange} />
        <button type="submit">Login</button>
      </form>
    </div>
  );
};
//The LoginForm component is connected to the Redux store using the connect function from 'react-redux'.
//The first argument, null, is used to indicate that the component does not need to subscribe to any state updates from the Redux store.
//The second argument, { login }, provides the login action creator as a prop to the LoginForm component, allowing it to dispatch the login action when needed.
export default connect(null, { login })(LoginForm);
```

created by:jaimeen makavana

```jsx
//user profile component-2

//imported for using JSX and creating React components.
import React from 'react';
//to connect the component to the Redux store.
import { connect } from 'react-redux';
//This line defines a functional component named UserProfile which takes in the user object as a
//destructured prop. The user object is obtained from the Redux store using mapStateToProps (explained later).
const UserProfile = ({ user }) => {
  return (
    <div className="container">
      <h3>User Profile</h3>
      <p>Welcome, {user.email}!</p>
    </div>
  );
};
//This function is called mapStateToProps, and it maps the Redux state to the component's props.
//It takes the state as an argument and returns an object that defines the user prop.
//In this case, state.auth.user indicates that the user object is located in the auth slice of the Redux store.
//Mapping the Redux state to the component's props means that you are selecting specific
// pieces of data from the Redux store and making them available as props within your component.
// This allows you to access and use that data in your component's rendering and logic.

const mapStateToProps = (state) => {
  return {
    user: state.auth.user
  };
};
//Here, state refers to the entire Redux state object. By accessing state.auth.user,
// it retrieves the user object stored in the auth slice of the Redux store.
export default connect(mapStateToProps)(UserProfile);
```

created by:jaimeen makavana

```
//actions->authAction.js

import { LOGIN_SUCCESS } from '../constants';
//This function takes in two parameters: email and password, representing the credentials used for login.
export const login = (email, password) => {
//The returned function is known as a "thunk" and is used in Redux to perform asynchronous operations or side effects.
  return (dispatch) => {
    setTimeout(() => {
//This code block dispatches an action to the Redux store. The dispatched action has a type of LOGIN_SUCCESS,
//indicating a successful login. It also includes a payload object containing the email value passed to the login function.
      dispatch({
        type: LOGIN_SUCCESS,
//  the payload is an optional property of an action object that carries any
//additional data associated with the action. It provides a way to pass data from
// the action creator to the reducer, allowing the reducer to update the state based on that data.
        payload: {
          email: email
        }
      });
    }, 1000);
  };
};
```

```
//reducers->authReducer.js

import { LOGIN_SUCCESS } from '../constants';
//Defines the initial state for the authentication slice
const initialState = {
  user: {}
};
//authReducer function:which is responsible for handling state updates for the authentication slice.
const authReducer = (state = initialState, action) => {
  switch (action.type) {
    case LOGIN_SUCCESS:
//return:Returns a new state object with the spread operator (...state)
// to preserve the existing state properties. The user property is updated
// with the value from action.payload, which represents the data associated with the login success.
      return {
        ...state,
        user: action.payload
      };
    default:
      return state;
  }
};

export default authReducer;
```

```
//redux->rootReducer

import { combineReducers } from 'redux';//function is used to combine multiple reducers into a single root reducer.
import authReducer from './reducers/authReducer';//responsible for managing the state related to authentication in the appl

const rootReducer = combineReducers({
  auth: authReducer
});
// combineReducers function takes an object as an argument,
//where the keys represent different slices of the state,
//and the values are the corresponding reducer functions.
//slices of the state:the application state is typically divided into multiple smaller parts,
// often referred to as "slices." Each slice represents a specific portion
// of the overall state that is managed by a separate reducer function.
export default rootReducer;
```

```
//redux->store

import { createStore, applyMiddleware } from 'redux'; //These functions are used to create the
// Redux store and apply middleware(It allows you to intercept and modify actions before
// they reach the reducers.It provides a way to perform extra operations on actions,
//such as asynchronous API calls, logging, or transforming actions, before they are processed by the reducers.
//) to the store, respectively.
import thunk from 'redux-thunk';//a middleware that allows you to write asynchronous logic in Redux actions.
import rootReducer from './rootReducer';//can be used to create the Redux store.

const store = createStore(rootReducer, applyMiddleware(thunk));

export default store;
```

```jsx
//app.js

import React from 'react';
import { Provider } from 'react-redux'; //provide access to the redux store to the components.
import store from './redux/store'; //The store object represents the Redux store, which holds the application state.
import LoginForm from './components/LoginForm';
import UserProfile from './components/UserProfile';

const App = () => {
return (
  <Provider store={store}>
  {/* allows any nested components to access the Redux store and dispatch actions.
  This allows those components to connect to the store, retrieve the state,
  and dispatch actions(means triggering an action to update the state in the Redux store.) as needed. */}
    <div className="App">
      <LoginForm />
      <UserProfile />
    </div>
  </Provider>
);
};

export default App;
```

```
//index.js

import React from 'react';
import { createRoot } from 'react-dom/client';
import App from './App';

createRoot(document.getElementById('root')).render(
  <React.StrictMode>
    <App />
  </React.StrictMode>
);
```