```
window.onload = cs445Project;
function cs445Project() {
  const displayPage = document.getElementById("outlet");
  let token;
  let timerId;
  const geokey = `1jwCRc01HYt6VS4GQSVQTMxsHTAIHqGt`;

  //let tokenSatus;
  const loginTemplate = `<div class="log-form">
    <h1 >Please Login</h1>
    Username : <input type="text" value="mwp" id="username" /><br><br>
    Password : <input type="text"  value="123" id="password"/><br><br>
    <button type="button" class="btn" id="login">Login</button><br>
  </div><!--end log form -->`;

  //animation template
  const animationTemplate = `<div id="location"> welcome to SPA Animation</div>
  <textarea id="loding" rows="20" cols="50"></textarea><br>
  <button id="refresh" >Refresh Animation</button>
  <button id="logOut"> LogOut </button>`;

  //default template for login template
  displayPage.innerHTML = loginTemplate;

  history.pushState("login", null, "?/login");

  // adding listener and adding state to history
  const loginButon = document.getElementById("login");
  loginButon.addEventListener("click", (_) => {
    history.pushState(
      {
        page: "login",
      },
      null,
      "?animationPage"
    );
    loginPage();
  });

  async function loginPage() {
    try {
      let username = document.getElementById("username").value;
      let password = document.getElementById("password").value;
      let logObject = {
        username: username,
```

```
    password: password,
  };

  //  change into animation template
  displayPage.innerHTML = animationTemplate;

  //  1st route to fetch token
  let response = await fetch(
    " https://shrouded-badlands-76458.herokuapp.com/api/login ",
    {
      method: "POST",
      headers: {
        "Content-Type": "application/json",
      },
      body: JSON.stringify(logObject),
    }
  );

  // result  from the Json response
  let result = await response.json();
  token = result.token;
  console.log(token);

  // start animation onlogin
  getAnimation();
  // displaying location
  geoLocation();

  //adding lisinner to refresh the page loding new animation each time
  document.getElementById("refresh").addEventListener("click", (_) => {
    if (timerId) clearInterval(timerId);
    getAnimation();
  });
  //  history push after click refresh
  const refresh = document.getElementById("refresh");
  refresh.addEventListener("click", (_) =>
    history.pushState(
      {
        page: "refresh",
      },
      null,
      "?/refrashAnimation"
    )
  );
```

```
    //throwing error message
  } catch (error) {
    console.log(`Error message : ${error}`);
  }
}

async function getAnimation() {
  // 2nd rout to fetch animation string
  try {
    let getAnimation = await fetch(
      "https://shrouded-badlands-76458.herokuapp.com/api/animation",
      {
        headers: {
          Authorization: `Bearer ${token}`,
        },
      }
    );

    // response with animation string
    let animation = await getAnimation.text();

    // creating animation
    let animationArray = animation.split("=====\n");
    let index = 0;
    timerId = setInterval((_) => {
      document.getElementById("loding").value = animationArray[index];
      index++;
      // looping again the animation
      if (index == animationArray.length) {
        index = 0;
      }
    }, 200);

    // adding lisinner into logout button
    document.getElementById("logOut").addEventListener("click", (_) => {
      displayPage.innerHTML = loginTemplate;

      // adding history pushstate into login after logout
      const logOutButton = document.getElementById("logOut");
      logOutButton.addEventListener("click", (_) =>
        history.pushState(
          {
            page: "logOut",
          },
          null,
```

```
            "?/login"
          )
        );

        // clear the token after each animation
        token = null;
        // clear animation list
        animationArray.splice(index, animationArray.length - 1);
        // clear timeInterval
        clearInterval(timerId);
      });
    } catch (error) {
      console.log(`Error message : ${error}`);
    }
  }

  async function geoLocation() {
    try {
      navigator.geolocation.getCurrentPosition(success);

      async function success(position) {
        let long = position.coords.longitude;
        let lat = position.coords.latitude;

        let geoResponse =
          await fetch(`http://www.mapquestapi.com/geocoding/v1/reverse?key=${geok
ey}
                      &location=${lat},${long}&includeRoadMetadata=true&includeN
earestIntersection=true`);
        let location = await geoResponse.json();
        const city = location.results[0].locations[0].adminArea5;
        const state = location.results[0].locations[0].adminArea3;
        const country = location.results[0].locations[0].adminArea1;

        document.getElementById(
          "location"
        ).innerHTML = `Welcome all from ${city},${state},${country}`;
      }
      //  throwing error
    } catch (error) {
      document.getElementById(
        "location"
      ).innerHTML = `Welcome all to  SPA !! Your location is not defined`;
      console.log(error);
    }
```

```
  }
  // managing the back forward arrows
  window.addEventListener("popstate", (_) => {
    if (history.page == "login") {
      displayPage.innerHTML = loginTemplate;
    } else if (history.page == "animation") {
      displayPage.innerHTML = loginTemplate;
    } else if (history.page == null) {
      displayPage.innerHTML = loginTemplate;
    }
  });
}
```