SUNSTONE

“Social Media Website And Application”

Task - 4

SUNSTONE

Social Media Website And Application

- This document serves as an introductory guide to social media websites and applications. It aims to provide readers with a comprehensive understanding of these digital platforms, their functions, and their significance in contemporary society.

LMS Username	Name	Batch
au310520205057	Sentamil kumaran B	04
au310520205053	Santhosh B	04
au310520205056	Selva kumar K	04
au310520205047	Rakesh R	04

Task 4 :: Dynamic Frontend

Integrate the Databse abd APIs to frontend to ensure the dynamic feature of website

- Connect app.js to firebase
- Implement api calls for each element
- Handle errors in the output
- Render output of apis to different low level components
- Secure content of post apis

Evaluation Metric:

100% Completion of the above tasks

Learning outcome

- Developing complicated UI using HTML components
- Using props drilling and context to pass variables
- Getting familiar with different type of api calls
- Handling different input data

Integrate the APIs to frontend to ensure the dynamic feature of website

Let's go step by step to integrate APIs into your React frontend for a dynamic experience.

❑ Set the firebase Base URL:

```
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.cloud.netflix.eureka.EnableEurekaClient;
import org.springframework.cloud.netflix.zuul.EnableZuulProxy;


@SpringBootApplication
@EnableEurekaClient
@EnableZuulProxy
public class InstaApiGatewayApplication {

		public static void main(String[] args) {
			SpringApplication.run(InstaApiGatewayApplication.class, args);
		}

}
```

❑ Design API Calls for Each Element:

Design API calls in your withDataFetching.js file for each element (e.g., restaurants, cart, etc.). Update the endpoints accordingly:

// higherOrderComponents/withDataFetching.js

```
package com.clone.instagram.instaapigateway.config;

import io.jsonwebtoken.Claims;
import io.jsonwebtoken.Jwts;
import org.springframework.security.authentication.UsernamePasswordAuthenticationToken;
import org.springframework.security.core.authority.SimpleGrantedAuthority;
import org.springframework.security.core.context.SecurityContextHolder;
import org.springframework.web.filter.OncePerRequestFilter;

import javax.servlet.FilterChain;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.io.IOException;
import java.util.List;

import static java.util.stream.Collectors.toList;
```

```
public class JwtTokenAuthenticationFilter extends OncePerRequestFilter {
private final JwtConfig jwtConfig;
public JwtTokenAuthenticationFilter(JwtConfig jwtConfig) { this.jwtConfig = jwtConfig; }
@Override protected void doFilterInternal(HttpServletRequest request, HttpServletResponse response,
FilterChain chain) throws ServletException, IOException {
// 1. get the authentication header. Tokens are supposed to be passed in the authentication header String
header = request.getHeader(jwtConfig.getHeader());
// 2. validate the header and check the prefix if(header == null || !header.startsWith(jwtConfig.getPrefix())) {
chain.doFilter(request, response); // If not valid, go to the next filter. return; }
// If there is no token provided and hence the user won't be authenticated. // It's Ok. Maybe the user
accessing a public path or asking for a token.
// All secured paths that needs a token are already defined and secured in config class. // And If user tried
to access without access token, then he won't be authenticated and an exception will be thrown
```

// 3. Get the token String token = header.replace(jwtConfig.getPrefix(), "");
try { // exceptions might be thrown in creating the claims if for example the token is expired
// 4. Validate the token Claims claims =
Jwts.parser() .setSigningKey(jwtConfig.getSecret().getBytes()) .parseClaimsJws(token) .getBody();
String username = claims.getSubject(); if(username != null) { List<String> authorities = (List<String>)
claims.get("authorities");
// 5. Create auth object
// UsernamePasswordAuthenticationToken: A built-in object, used by spring to represent the current authenticated / being authenticated user.
// It needs a list of authorities, which has type of GrantedAuthority interface, where SimpleGrantedAuthority is an implementation of that interface UsernamePasswordAuthenticationToken auth = new UsernamePasswordAuthenticationToken(username, null,
authorities .stream() .map(SimpleGrantedAuthority::new) .collect(toList()));

```
// 6. Authenticate the user
                // Now, user is authenticated
                SecurityContextHolder.getContext().setAuthentication(auth);
            }

        } catch (Exception e) {
            // In case of failure. Make sure it's clear; so guarantee user won't be authenticated
            SecurityContextHolder.clearContext();
        }

        // go to the next filter in the filter chain
        chain.doFilter(request, response);
    }
}
```

Html For Frontend

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1" />
    <meta
      name="description"
      content="Instagram clone built using ReactJS"
    />
    <title>Instagram Tribute</title>
  </head>
  <body>
    <noscript>You need to enable JavaScript to run this app.</noscript>
    <div id="root"></div>
  </body>
</html>
```

Updating Index.js

```
import React from 'react';
import ReactDOM from 'react-dom';
import App from './App';
ReactDOM.render(
  <React.StrictMode>
    <App />
  </React.StrictMode>,
  document.getElementById('root')
);
```

ow, we can begin to build our app frontend. We'll start by building the header which will hold the Instagram logo and buttons to log in, sign up, and log out.

We'll do this in App.js so that it looks like this:

```
import React from "react";
import "./App.css";

function App() {
  return (
    <div className="app">
      <div className="app__header">
        <div className="app__headerWrapper">
          <img
            src=" https://www.instagram.com/static/images/web/mobile_nav_type_logo.png/
735145cfe0a4.png"
            alt="Instagram original logo"
          />
          <button className="text__button">Logout</button>
          <div className="app__headerButtons">
            <button className="primary__button">Log in</button>
            <button className="text__button">Sign up</button>
          </div>
        </div>
      </div>
    </div>
```

Then add the base styling and styling for the header in App.css, so that the file looks like this instead:

```
* {
  margin: 0;
  padding: 0;
  box-sizing: border-box;
  font-family: -apple-system, BlinkMacSystemFont, "Segoe UI", Roboto, Helvetica,
    Arial, sans-serif;
}
body {
  color: #262626;
  font-size: 14px;
  line-height: 18px;
}
.app {
  background-color: #fafafa;
  min-height: 100vh;
```

```
.app__headerWrapper {
  max-width: 975px;
  margin: 0 auto;
  width: 100%;
  padding: 0 20px;
  display: flex;
  align-items: center;
  justify-content: space-between;
}
.app__headerButtons {
  display: flex;
  column-gap: 15px;
}
button {
  font-size: 14px;
  line-height: 18px;
  cursor: pointer;
  border: 1px solid transparent;
}
.primary__button {
  background-color: #0095f6;
  border-radius: 4px;
  }
```

```
.text__button {
  color: #0095f6;
  font-weight: 600;
  line-height: 28px;
  background-color: transparent;
}
.app__signUp {
  display: flex;
  flex-direction: column;
  margin: 28px 0;
  row-gap: 15px;
}
```

Connecting App.js with post component

Then, we connect the post component to App.js, by importing it, and also update App.js so that it becomes

```
import React, { useState } from "react";
import Post from "./components/post/Post";
import "./App.css";

function App() {
  const [posts, setPosts] = useState([
    {
      username: "blessingthebobo",
      caption: "Wow, I'm Amazing!",
      imageUrl:
        "https://images.unsplash.com/photo-1637014387463-a446e89abb68?
ixid=MnwxMjA3fDB8MHxlZGl0b3JpYWwtZmVlZHw1fHx8ZW58MHx8fHw%3D&ixlib=rb-
1.2.1&auto=format&fit=crop&w=500&q=60",
    },
    {
      username: "godtello",
      caption: "Oh, I'm a God!",
}
```

```
return (
    <div className="app">
      <div className="app__header">
        <div className="app__headerWrapper">
          <img
            src=" https://www.instagram.com/static/images/web/mobile_nav_type_logo.png/735145cfe0a4.png"
            alt="Instagram original logo"
          />
          <button className="text__button">Logout</button>
          <div className="app__headerButtons">
            <button className="primary__button">Log in</button>
            <button className="text__button">Sign up</button>
          </div>
        </div>
      </div>
      <div className="timeline">
        {posts.map((post) => (
          <Post
            username={post.username}
            caption={post.caption}
            imageUrl={post.imageUrl}
          />
```

Submission Github

https://github.com/SandeSanthosh/NM-DSCET-04

Thank you!