|  |  |  |  |
| --- | --- | --- | --- |
| Code Specification | Reference | Files changed | Changes made (Y/N) |
| Instagram  App id 1287139948354496 | App Secret  94d1d3c26ff6d7cc751e949348bd70bb | Instagram display name  Jyothsna\_Image\_Viewer |  |
| Login Page (Part-I)  You must create a folder named ‘login’ inside the ‘screens’ folder. Inside the ‘login’ folder, create ‘Login.js’ and ‘Login.css’ files. |  |  | Y |
| The header on the login page should look like this:    The background color of the header must be #263238, which is the hex color code for the dark grey color  The header must run across the entire width of the browser window. This means that there should be no padding or margin on any side of the header and the header’s black background must touch the boundaries of the browser window. |  |  | Y |
| The logo of your application must be displayed on the left side inside the header.  The logo must be in the form of text and must read “Image Viewer”  The logo of the application must be displayed with the following CSS properties:  The color of the text must be #fff, which is the hex color code for the white color.  The font size of the text must be 18px.  The text must be vertically in the center of the header. |  |  | Y |
| The Login card -    1. The login form must be displayed inside a card. You must use Material-UI ‘Card’ component on the login page. This card must be displayed horizontally in center of the page  2. Inside the login card, you must display ‘LOGIN’ (all uppercase) as the heading  3. Below the LOGIN heading, you must display two text fields - one for username and the other for password as shown in the screenshot above. You must use Material-UI ‘FormControl’ component for each text field. Inside each ‘FormControl’ component, you must use ‘InputLabel’ and ‘Input’ components for displaying these text fields |  |  | Y |
| Initially, the text field for username must display ‘Username’ as the placeholder text. |  |  | Y |
| Initially, the text field for password must display ‘Password’ as the placeholder text. |  |  | Y |
| Below the password text field, a button must be displayed for logging the user into the application. You must use Material-UI ‘Button’ component for this purpose.  This button must display ‘LOGIN’ as the label, must have the variant of type ‘contained’ and color of enum ‘primary’.    When a user hovers/focuses on this button, the cursor should change to a pointer, which displays a pointing hand. |  |  | Y |
|  |  |  | Implemented, but it has to be tested thoroughly |
|  |  |  | Implemented, changes required while integrating the backend. |
| The search bar and profile button should not appear on the login page. |  |  | N |
| To implement the idea explained above, inside the ‘onClick’ event handler of the ‘login’ button in the Login.js file, declare two variables with the name ‘username’ and ‘password’ and initialize them with the username and password respectively of the user. Note that this username or password does not have to be the one with which you created your account. It can be any random text. |  |  |  |
| You must also declare a variable containing the ‘access-token’ that you obtained from your Instagram developer account. |  |  |  |
| If the user enters the same username and password as mentioned in the variables inside the function, the user must be directed to the next page, which is the home page and the access-token must be set inside the session storage with the key as ‘access-token’. |  |  |  |
|  |  |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
| Code Specification | Reference | Files changed | Changes made (Y/N) |
| The home page must be displayed only after a user enters the correct credentials as per the ones (username and password) mentioned inside the Login component. Make sure that the URL for the home page is:  localhost:3000/home |  |  |  |
| Home Page (Part II)  The header on the home page should look like this: |  |  | Implemented but changes have to be made while integrating the backend |
| Apart from the logo, there are two elements, which are added to the header on this page. These elements are given below:  **Search Box**  The search box must be displayed on the right side of the header.  **User Profile Icon**  The icon displaying the profile picture of the currently logged-in user must be displayed on the right side of the search box inside the header.  Make sure that you maintain the position of these elements inside the header, as explained above. |  |  | Implemented but changes have to be made while integrating the backend |
| The search box must have the following CSS properties:  The border radius of the search box must be 4px.  The width of the search box must be 300px.  The background color of the search box must be #c0c0c0, which is the hex color code of grey color. |  |  | Y |
| The search box must be displayed vertically in center of the header |  |  | Y |
| The search box must display a microscope-shaped icon on the left side of the box, which is representative of search functionality. You must use the [SVG Material icon](https://material-ui.com/style/icons/#svg-material-icons) with the name ‘Search’ for this purpose. |  |  | Y |
| On the right side of the search icon, you must display an input box where a user can type in a text value. You must use Material-UI ‘Input’ component for this purpose. The placeholder of this component must be ‘Search…’. You must note that no underline must be displayed below the ‘Input’ component. |  |  | Y |
| The search box performs the functionality of searching an image based on its caption. The user may enter any substring of the image caption. As the user starts typing something inside the search box, the image(s) for which the caption matches or contains the value input by the user, is(are) shown in the result.  For an instance, there are three images with the caption - “This is image1”, “This is image2”, and “This is image3”. When a user enters “image”, all the three images must be shown but when the user types 1 after the “image” word, only the first image post must be displayed. If the user types a word which does not exist in the description of an image (let’s say that the user enters “image4” in the search box), then no image must be displayed |  |  | No, must be implemented |
|  |  |  |  |
| The user profile icon inside the header on the home page should look like this:    The user profile icon must display the profile picture of the user who is currently logged in. **This information must be displayed from a state variable which stores the property ‘profile\_picture’ fetched from the response data of the first API endpoint.** |  |  | Must display picture of the user inside the box |
| To display the user profile icon as a button, you must use Material-UI ‘IconButton’ component. |  |  | Used Select, try implementing IconButton |
| When a user hovers on this icon, the cursor must change to pointer, which displays a pointing hand. |  |  | Y |
| When a user clicks on this icon, a dropdown menu must be displayed which contains two menu items - ‘My Account’ and ‘Logout’ separated by a horizontal rule, as shown in the screenshot below: |  |  | Slight changes required |
| When the user clicks on ‘**My Account**’ menu item, **the user must be taken to the profile page,** which will be discussed later in the next part of the project statement |  |  | Must be implemented |
| When the user clicks on ‘Logout’ menu item, session storage must be cleared (i.e. the session storage must not contain ‘access-token’ key) and the user must be taken back to the login page. |  |  | Must be implemented |
| **Points to keep in mind while developing the ‘image post’ on the HOME page:**  The image post, which consists of an image and other relevant information associated with it, should look like: |  |  |  |
| Each image post must be enclosed inside a card. You must use Material-UI ‘Card’ component for this purpose. |  |  |  |
| Each card has two parts - ‘CardHeader’ & ‘CardContent’. These components are part of the Material-UI ‘Card’ component. |  |  |  |
| Inside the header of the card (‘CardHeader’ component), you must display three elements:  User’s profile picture  The user’s profile picture must be displayed as an icon on the left side inside the ‘CardHeader’ component.  Username  The username, who posted the image, must be displayed on the right side of the profile picture icon.  Date  The date on which the image is posted must be displayed on the right side of the user’s profile picture icon and below the username. |  |  |  |
| Make sure that the profile picture is treated as one entity and the username and date together are treated as another entity. Both these entities should be aligned to each other and should be vertically in center of the ‘CardHeader’ component. |  |  |  |
| Inside the content of the card (‘CardContent’ component), you must display the following elements or functionalities in the given order:  Image The image posted by the user must be the first element to be displayed inside the ‘CardContent’ component. A horizontal rule element must be displayed after displaying the image.  Caption of the image The caption of the image posted by the user must be displayed below the horizontal rule element.  Hashtags The hashtags associated with the image must be displayed below the image caption.  Like icon An icon which makes a user like an image must be displayed below the hashtags  Number of likes The number of likes for the image must be displayed to the right side of the like icon.  Comments All the comments for the image must be displayed below the number of likes for the image.  Add Comment Functionality for adding a comment must be displayed below all the comments for the image. |  |  |  |
|  |  |  |  |

Create the files userData.js, imageData.js & imageCaption.js under common folder, based on the information provided in **developers.facebook.com**.

|  |  |  |
| --- | --- | --- |
| Referred link - <https://developers.facebook.com/docs/instagram-basic-display-api/guides/getting-profiles-and-media#get-a-user-s-media> | User profile response –  {  "id": "17841405793187218",  "username": "jayposiris"  } | userData.js  let userData = [      {          "id" : "1",          "username" : "user1"      },      {          "id": "2",          "username" : "user2"      },      {          "id": "3",          "username" : "user3"      }  ]  export default userData; |
| Referred link-  <https://developers.facebook.com/docs/instagram-basic-display-api/guides/getting-profiles-and-media#get-a-user-s-media> | Image response –  {  "id": "17895695668004550",  "media\_type": "IMAGE",  "media\_url": "https://fb-s-b-a.akamaihd.net/...",  "username": "jayposiris"  "timestamp": "2017-08-31T18:10:00+0000"  } | //imageData.js file  let imageData= [  {        "id": "550",        "media\_type": "IMAGE",        "media\_url": "https://upload.wikimedia.org/wikipedia/commons/thumb/9/98/Al\_Pacino.jpg/220px-Al\_Pacino.jpg",        "username": "user1",        "timestamp": "2017-08-31T18:10:00+0000"      },  {        "id": "551",        "media\_type": "IMAGE",        "media\_url": "https://upload.wikimedia.org/wikipedia/en/0/08/Annabelle\_Creation.jpg",        "username": "user1",        "timestamp": "1972-03-15T00:00:00+05:30"      },  {        "id": "552",        "media\_type": "IMAGE",        "media\_url": "https://upload.wikimedia.org/wikipedia/commons/e/e7/Prabhleen\_Sandhu\_%28cropped%29.jpg",        "username": "user2",        "timestamp": "1972-03-15T00:00:00+05:30"      },  {        "id": "553",        "media\_type": "IMAGE",        "media\_url": "https://upload.wikimedia.org/wikipedia/en/c/cd/Shahid\_Poster\_%282013%29.jpg",        "username": "user3",        "timestamp": "2017-08-31T18:10:00+0000"      },  {        "id": "554",        "media\_type": "IMAGE",        "media\_url": "https://upload.wikimedia.org/wikipedia/commons/b/bd/Anne\_Hathaway\_in\_2017.png",        "username": "user3",        "timestamp": "2017-08-31T18:10:00+0000"      }  ]  export default imageData; |
| Captions of each image are stored in separate file. They are also retrieved as shown in the next column.  Referred link-  <https://developers.facebook.com/docs/instagram-basic-display-api/guides/getting-profiles-and-media#get-a-user-s-media> | {  "data": [  {  "id": "17895695668004550",  "caption": ""  },  {  "id": "17899305451014820",  "caption": ""  },  {  "id": "17896450804038745",  "caption": ""  },  {  "id": "17881042411086627",  "caption": ""  }  ],  "paging": {  "cursors": {  "after": "MTAxN...",  "before": "NDMyN..."  },  "next": "https://graph.faceb..."  }  } | imageCaption.js –  let imageCaption = [{  “data”: [  {  "id": "550",  “caption”: “Image1 Caption”  },  {  "id": "551",  “caption”: “Image2 Caption”  },  {  "id": "552",  “caption”: “Image3 Caption”  },  {  "id": "553",  “caption”: “Image4 Caption”  },  {  "id": "554",  “caption”: “Image5 Caption”  }],  "paging": {  "cursors": {  "after": "MTAxN",  "before": "NDMyN"  },  "next": "https://graph.facebook.com"  }  ]  export default imageCaption; |
| New file has to be created with User information like –  User full Name –  Number of posts –  Follows-  Followed By – |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
| Code Specification | Reference | Files changed | Changes made (Y/N) |
| Make sure that the URL for the profile page is: localhost:3000/profile |  |  |  |
| Also, note that if a user is not logged in and tries to go to the profile page by changing the URL, then the user must be taken back to the login page. |  |  | N |
| The header on the profile page looks like this:      The logo must be on the left side and the profile icon must be on the right side inside the header.  You must reuse the same logo and the same profile picture icon, which you developed on the home page. This means that you must reuse the header that you have developed on the login and the home page.  The logo ‘Image Viewer’ must be clickable. When a user clicks on the logo, the user must be taken to the home page.  The profile icon button, when clicked, must display a dropdown menu with only one item - ‘Logout’. |  |  | Y |
| When a user hovers on ‘Logout’ menu item, the cursor must become a pointer and when the ‘Logout’ menu item is clicked, the session variable for the ‘access-token’ must be cleared from the session storage and then the user must be taken back to the starting page of the application, which is the login page. |  |  | N  Changes need to be made for access token |
| This information section looks like this:     1. **User’s profile picture** 2. **Username** 3. **Number of posts** 4. **Number of users followed** 5. **Number of followed-by users** 6. **Full Name** |  |  | File must be created to generate user information with fields full-name, username, profile-picture, followed-by-count, following-count |
| Edit Icon button –  Material-UI ‘Button’ component for this purpose with the properties - variant as ‘fab’ and color as ‘secondary’. This button must display an edit icon, as shown in the screenshot. |  |  | Y |
| When this edit button is clicked, a modal opens which asks for the full name and looks like this: |  |  | Y |
| ‘image posts’ below the information section on the PROFILE page: These images must be displayed as a grid that consists of three images in one row. This should look like this:    You must use Material-UI ‘GridList’ (Image-only Grid list) for this purpose. |  |  | Y |
| ‘image modal’ popped up on clicking on an image post on the PROFILE page: |  |  | N |
| On the left side of the modal, the image which is clicked on must be displayed. You need to fetch this data from the API endpoint provided to you.  On the right side of the image and on the top of the modal, the profile picture icon of the username, who posted the image, must be displayed.  You will not get the user profile icon from the API endpoints. Hence, you must hard-code the same in your UI and may use an image of your own choice.  To the right of the profile picture icon, the username of the user, who posted the image, must be displayed. You need to fetch this data from the API endpoint provided to you.  Below the user information, a horizontal line, acting as a separator, must be displayed.  Below the horizontal line, the caption of the image must be displayed.   Below the caption of the image, the hashtags associated with the image must be displayed. You need to fetch the caption from the API endpoint provided to you. Your caption will consist of hashtags. You need to extract all the words which start with a # (pound) symbol and display all these words prefixed with the # symbol. Each hashtag should be separated by a space.  Below the hashtags, the user comments on the image (added using the input field and add button at the bottom of the modal) must be displayed. This will fetch all the comments, if any, added to all images on the home page. For example, there in an image I, on which you added a comment C on the home page. When you click on this image I in the profile page, you should get to see the comment C in the modal that pops up.  Below the user comments, a heart-shaped ‘like’ button must be displayed. When a user clicks on this icon, the icon must turn red in color, indicating that the image has been liked. Also, when a user again clicks on the red-colored icon, the icon must turn back to the original state with no color filled. Note that this functionality will not actually like the image post on Instagram. It will just like the image on the local application page, which you are developing.  Alongside the ‘like’ icon, the number of likes on the image must be displayed. This will fetch the number of likes, if any, added to all images on the home page. For example, there is an image I which was liked by you on the home page. When you click on this image I in the profile page, you should get to see the red-color filled heart-shaped icon indicating that the image is already liked by you and it should also show the incremented number of likes alongside the icon. Remember that when a user clicks on the heart-shaped like button, the number of likes increment by 1. When the user clicks on this button again, the number of likes decrement by 1.  Note that this functionality will not actually increment the number of likes on the image post on Instagram. It will just increment the number of likes on the local application page, which you are developing. When a user refreshes the page, the count of the number of likes is refreshed to the number that exists in the Instagram database.  Below the like icon, an input box must be displayed along with an ‘ADD’ button. A user can type in a comment inside the input box and get it displayed along with the list of other comments (under the hashtags) after clicking on the ‘ADD’ button. Note that this will not actually add a comment on the image post on Instagram. It will just comment on the image on the local application page, which you are developing. When a user refreshes the page, all the comments, which you added to the image in the local application vanish. |  |  |  |

Important points to generate Access Token –

<https://developers.facebook.com/docs/instagram-basic-display-api/guides/getting-profiles-and-media#get-a-user-s-media>

### Step 1: Get An Access token and Permissions

### Step 2: Query the User Media Edge

## Step3: Get Media Data

### Step 1: Get An Access token and Permissions

### Step 2: Query the Media node

Image response –

{

"id": "17895695668004550",

"media\_type": "IMAGE",

"media\_url": "https://fb-s-b-a.akamaihd.net/...",

"username": "jayposiris"

"timestamp": "2017-08-31T18:10:00+0000"

}