Login cases and flow:

1. On website reload or first time initialization display frontend but also check if the server is running, if server is running then check for connection establishments with the mongodb cluster, if not then ask the users to browse the pages and try to access hidden content after sometime as the server is not available at the moment.
2. Google react oauth screen retrieves the authorization code and sends it to the backend:
3. Scenario 1: Request not sent:
   * Backend connection was not available – if so then prompt the user to request after sometime as the servers are not available at the moment, log the issue in the backend.

* Middleware Setup Issue – prompt the user servers are unavailable try again later and log the issue in the backend.

1. Scenario 2: Request sent successfully:
   * Then backend validates the input request to have authorization code
     1. If Authorization Code is not available then prompt the user and log the issue in the backend using the handle\_exception and send response functions.
     2. If the Authorization code is available then use it to request the access token by making a call to the /token endpoint.
        + If the token request was unsuccessful then log the details of the error using the handle\_exception and send response functions.
        + If the request was successful then use the access token and refresh token and get the user information by calling the appropriate endpoint.

On error in fetching user information log the error and handle response using the handle exception and send response functions.

If user information was retrieved successfully then validate the user information to check if any field is not empty if not then store the results in the database (mongodb).

For storing the data In the collections in mongodb first check if a successful connection via the mongodb instance is established or not, if not then retry and if you are unable to establish the connection then prompt the user with message unable to establish connection due to server side issues.

If the connection is established successfully and then store data:

1. If data storing was unsuccessfull then prompt the user with the message unable to sign up due to server side issues- pleas try again if again the same issue persists and redirect to landing page then ask the user to login via their email and password instead of login with google.
2. If data storing was successful then return the access token and refresh tokens to the frontend along with basic user information like users full name, profile pic and redirect them to home page.
3. Now store the access tokens and the refresh tokens in the local storage in the frontend and use them later for all the api calls to the backend to validate if they have access to that particular service.

Home page flow:

1. Existing workspaces option on the topic - List the existing workspaces by making a call to the /list workspace endpoint and send user name and access token in the request.
2. Change model for responses option besides the existing workspaces option.

Sidebar:

1. Create workspace option- make an api call to the /create-workspace option along with username.
2. Create New chat option – make a call to the /create-chat endpoint with username and workspace name.
3. Previous chats in the Sidebar.
4. Previously uploaded media files on the right side of the page.
5. Chat interface in the middle with options to send message, upload media files, provide voice input and send button option.

Backend:

1. Existing workspaces:
   * Scenario 1: Workspaces exist then first validate the access token.
     1. If the access token is invalid then:
        + Check for refresh tokens validity:
          1. If the refresh token is expired then ask the user to login in with the message that session has expired message and remove the access token and refresh token form the local storage.
          2. If the refresh token is not expired then use it to generate a new access token by making a api call to the appropriate google endpoint store it in the database and send to the frontend.
     2. If the access token is valid then:
        + Check for users existing workspaces using the user id in the workspace mapping collection and send a list of all the existing collections.
        + By default select the very first workspace and use the workspace id to get existing files uploaded in the file uploads collection as well as the previous chats for the workspace id and return chat id along with the last 5 message for each chat id in markdown format.

* + Scenario 2: Workspaces don’t exist then ask the user to create a workspace first.

1. Make an api call for the selected model name to the backend and initialize the selected model for the same answering the subsequent queries by the user.
2. For the /create-workspace endpoint first validate the request to have user name and workspace name.
   * Generate a workspace id for the new workspace name and create a collection in Qdrant with the workspace id and update the workspace mappings to have the workspace id and workspace name for the correct user.
   * On successful workspace creation prompt the user about successful workspace creation and choose it to get automatically to upload media files and create new chats.
3. For /create-chat endpoint based on the username and access token fetch the user id and create a new chat by generating a new chat id and answering the user query, based on the user query also ask the LLM to give a Chat name and store it in the respective user workspace collection with the user query and response, chat id and chat name and return the result to the user and rename the chat.