UttarAI - Chrome Extension

1. Introduction
    a. Chrome extension that generates smart AI-based replies for Gmail and WhatsApp conversations. It captures the conversation context, sends it to a Spring Boot-powered AI backend, and writes an appropriate response directly into the input box, saving time and making communication effortless.
2. Core Working
    a. For creating an extension, the main thing is the manifest.json file. this file has all the configurations like permissions, name, version etc
        i. In our extension, we have taken permissions like storage, active tab, tabs, etc.
        ii. host_permission: This allows the extensions to access the URLs mentioned here
        iii. content_scripts: Scripts are injected into the specified URL.
        iv. web_accessible_resource: This allows your extension to inject or reference these resources securely. This has the URLs and the resource you want to inject.
    b. In content.js, we have our main code of the WhatsApp part:
        i. Observer: this tells to watch the entire page (DOM).
        ii. Track changes to the child list and also go deep into the DOM tree.
        iii. Using the observer when every time there is a change in the DOM (when a new chat is opened). The observer will check whether the new tree has the input render or not
        iv. For checking whether the input has been rendered or not, we have to identify the class name of the input field by using the DevTools of the Chrome browser and analyzing the HTML structure of WhatsApp.
        v. Then have a use condition to check if the class (identified in b.iv) is present or not, if yes, then call the injectButton function, which will inject the button beside the input field.
    c. Inject Button:
        i. In this function, we are checking if the button is already present. If yes, remove the button (so multiple buttons are not displayed).
        ii. We find the toolbar (input field) using Chrome DevTools and store this element in the toolbar variable, so further we don’t have to find it again, we can simply use the toolbar variable to perform further operations.

            iii. Call the createButton function, which creates a button similar to the UI and theme of WhatsApp.
            iv. Call the conversationContext function, which is one of the main functions that is responsible for context-aware response generation. This uses the queryselector and, using the class (identified using DevTools) gets the chat fields.
                1. The issue is that this will get all the chats that are loaded in the DOM, and we don't require that much here, we are limiting the context to the last 10 chats
                2. For this, we are using the for loop to get the latest 10 chats and formatting them, also we are annotating whether the chat is sent by the user or received (using me/them for annotation). This helps our backend to understand context accurately.
                3. After all this, we make the API call on the click of the button we injected. The conversational context is sent in the body of the API call.
                4. Once the response is received, we are injecting this text into the input field using the toolbar
                5. This helps the user to see the response or make any changes into it or just hit the send button.
    d. For Gmail, the working is similar, only it does not need to get the conversational context of the last conversation it takes the context of the current mail only.

3. Razorpay payment gateway integration:
    a. For this, we first need to create the account at Razorpay and download the secret keys.
    b. In the frontend has been created a view pricing where different plans are displayed.
        i. When clicking on the plan the API call to backend is made with the details of the plan chosen to create the order.
    c. Backend gets the plan and fetches the details about the plan from the database and creates the order using these details
        i. For creating the order, first we need to add the Razorpay dependency into the pom.xml
        ii. Then, using the razorpayClient, create the order (pass the secret key, API id, amount, receipt id into it)
        iii. Then Razorpay will create the order and send the orderId and other details.
        iv. The backend will save these details and pass the required details (user info, amount, orderId, key, etc) to frontend.
    d. As the frontend receives the order, it triggers the Razorpay function

        i. For this, we need to add the script of Razorpay in index.html
        ii. Then will create the options object that contains all the details like ordered, amount, API key, user details, etc.
        iii. Then pass this object into the Razorpay function. This will start the payment process. A pop-up will appear from Razorpay to make the payment
    e. After the payment, if the payment fails, then Razorpay itself has the callback functions, using this, we will accordingly display the message regarding it,
    f. If the payment is successful, Razorpay returns the paymentId, signature, orderId. It also has the callback, using this, we will call the verify payment function.
    g. Verify payment is used to verify the payment at the backend. For this, we will make the API call to the backend and send paymentId, signature, and orderId in body.

    h. Backend used the Utils class of Razorpay and verified the payment, which requires the paymentId, signature, and orderId. This will either return true (for success) or false (for failure).
        i. If true, then we will update the details in the database and also update the user details with the plan purchased and add API call accordingly (we recharge the API calls after payments).
        ii. If false, then we will update accordingly

4. Detailed workflow of the system:
    a. Firstly, the user will create the account by entering the details. the backend will create the user and store it in the database along with the plan details as “free tier” and API calls left as “25”.
        i. We are offering the free tier for new users and giving them 25 free API calls.
    b. The user also needs to download and add the extension to use it.
    c. For using the extension, first, it requires logging in to the extension, which can be done by clicking on the extension. A pop-up will appear, which has the basic details and the login button.
    d. On click on this button, a new tab with the login form appears. After successful login, the tab will close automatically and the WhatsApp tab will open for the user automatically.
    e. If the user didn’t have the account, then to register user needs to visit our website.
    f. After logging user can seamlessly use the extension, whenever WhatsApp is open, the ‘AI-reply’ button will appear beside the input field. On click of this button, it will make the API call and inject the context-aware response in the input field. The user can either change it or just hit the send button.

g. For authorization extension, maintains the JWT token. In case the token expires and the user clicks on the 'AI-reply' button, it will give an alert that the "token has expired. Please log in" . The user needs to log in again and can use the button.
h. We offer the free tier of 25 free calls. If these free calls are over, and if clicks on the 'AI-reply' button, the user will get an alert that "API call limit reached, please upgrade the plan".
i. After the plan upgrade, the user can again use the service without any issue.
j. For upgrading the plan user needs to visit our website and go to the view pricing section.
    i. Here, there are various plans displayed
    ii. The user can choose whatever plan he/she want
    iii. For payment, we have integrated the Razorpay Payment Gateway, using this, you can seamlessly do payments.
k. Also, in the view pricing, we are also displaying the current plan and the API calls left so that the user can use it accordingly

{ Note: ***All payments are currently in test mode. No real money is involved. You can complete the payment process using dummy/test credentials, and your API call quota will be updated according to the selected plan.*}**