PHASE 5 :
PROJECT DOCUMENTATION AND SUBMISSION

Problem Definition: The project involves creating a chatbot using IBM Cloud Watson Assistant. The goal is to develop a virtual guide that assists users on messaging platforms like Facebook Messenger and Slack. The chatbot should provide helpful information, answer frequently asked questions (FAQs), and offer a friendly conversational experience. The project includes designing the chatbot's persona, configuring responses, integrating with messaging platforms, and ensuring a seamless user experience.

DESIGN THINKING :

1. **Friendly Greeting:** Cloudy starts every conversation with a warm and welcoming greeting to make users feel comfortable.

2. **Clear and Concise:** Cloudy communicates in a clear and straightforward manner, using language that's easy for users to understand, even if they are not tech-savvy.

3. **Informative:** Cloudy provides detailed and relevant information to answer user queries or help them perform tasks within the cloud application.

4. Empathetic: When users encounter issues or express frustration, Cloudy responds empathetically, acknowledging their concerns and offering solutions or assistance.

5. Professional: While being friendly, Cloudy maintains a professional tone, especially when handling sensitive data or transactions within the cloud application.

6. Responsive: Cloudy responds promptly to user inquiries and doesn't keep users waiting unnecessarily. It acknowledges when it needs time to fetch data or perform a complex task.

7. Adaptive: Cloudy adapts its responses based on user behaviour and the context of the conversation. It remembers user preferences and previous interactions to provide a personalised experience.

8. Encouraging: Cloudy encourages users to explore features, learn more about the application, and take advantage of its capabilities.

9. Problem-Solving: When users face challenges or errors, Cloudy actively engages in problem-solving by asking clarifying questions and guiding users step-by-step through solutions.

10. Closing the Conversation: Cloudy ends conversations politely, thanking users for their time and encouraging them to return if they have more questions or need further assistance.

Response Configuration:

1. Intents:

Intents represent the user's intention or what they want to achieve. Create intents based on the types of queries your users might have. For example:

- #AccountCreation
- #DataManagement
- #BillingQuestions
- #TechnicalSupport
- #PrivacyConcerns

2. Entities:

Entities help identify and extract specific pieces of information from user input. For instance, when dealing with billing inquiries, you might create entities like @SubscriptionPlan and @BillingIssueType to understand what the user is asking about.

3. Dialog Nodes:

Dialog nodes define how the chatbot responds to user input based on intents and entities. Create dialog nodes to handle various conversation branches. For example:

Welcome Node:

- Triggered by the #Welcome intent when the user first interacts with Cloudy.
- Respond with a friendly greeting.
- Transition to the main menu.

Account-Related Node:

- Triggered by the #AccountRelated intent.
- Provide information and guide users on account-related tasks.
- Transition back to the main menu when the user's query is resolved.

Data Management Node:

- Triggered by the #DataManagement intent.
- Offer assistance with data management tasks.
- Use conditional logic to address sub-topics like file uploads or organization.

Billing and Subscriptions Node:

- Triggered by the #BillingQuestions intent.
- Provide details on subscription plans, billing history, and upgrades.
- Transition back to the main menu when the user's query is resolved.

Technical Support Node:

- Triggered by the #TechnicalSupport intent.
- Offer troubleshooting steps and technical assistance.
- Transition back to the main menu when the issue is resolved.

FAQs Node:

- Triggered by intents related to common FAQs.
- Provide concise answers to frequently asked questions.
- Encourage users to ask follow-up questions or return to the main menu.

Fallback Node:

- Triggered when Cloudy doesn't understand the user's intent.
- Request clarification or offer suggestions for common queries.
- Ensure a smooth user experience even when the intent is unclear.

4. Dialog Flow:

Define the conversation flow by linking dialog nodes with appropriate conditions. For instance, after welcoming the user in the Welcome Node, transition to the main menu where users can choose various topics. Use conditions to guide users to the relevant nodes based on their intents and entities.

5. Responses:

Craft responses within each dialog node to provide helpful information, instructions, or answers to user queries. Use system variables to personalise responses, such as addressing the user by their name when available.

6. Testing and Training:

Continuously test and train your chatbot using Watson Assistant's training features. Fine-tune intents, entities, and dialog nodes based on user interactions and feedback to improve accuracy and user satisfaction.

Platform Integration :

1. Choose a Chatbot Framework:

To integrate Cloudy with multiple messaging platforms, you can use a chatbot framework or platform that supports multi-channel integration. Microsoft Bot Framework, BotPress, and Dialogflow are examples of frameworks that provide such capabilities.

2. Set Up Cloudy for Multi-Channel Integration:

Ensure that Cloudy is designed to be channel-agnostic, meaning it can receive and process messages from different sources without any platform-specific dependencies.

3. Create Accounts on Messaging Platforms:

Create developer accounts or bots on the messaging platforms you want to integrate with, such as Facebook for Messenger and Slack for Slack.

4. Configure Webhooks or APIs:

Messaging platforms typically offer APIs or webhooks that allow external services to interact with them. You'll need to configure these to send and receive messages to and from Cloudy.

5. Implement Platform-Specific Connectors:

Depending on the chatbot framework you choose, there may be pre-built connectors or adapters available for specific messaging platforms. These connectors simplify the integration process.

6. Handle Authentication and Authorization:

Securely handle authentication and authorization for each messaging platform. Ensure that your chatbot has the necessary permissions to interact with users on these platforms.

7. Message Routing and Parsing:

Implement logic to route messages from different platforms to the appropriate parts of Cloudy's conversation flow. You'll need to parse incoming messages to understand user input and extract relevant information.

8. Formatting Responses:

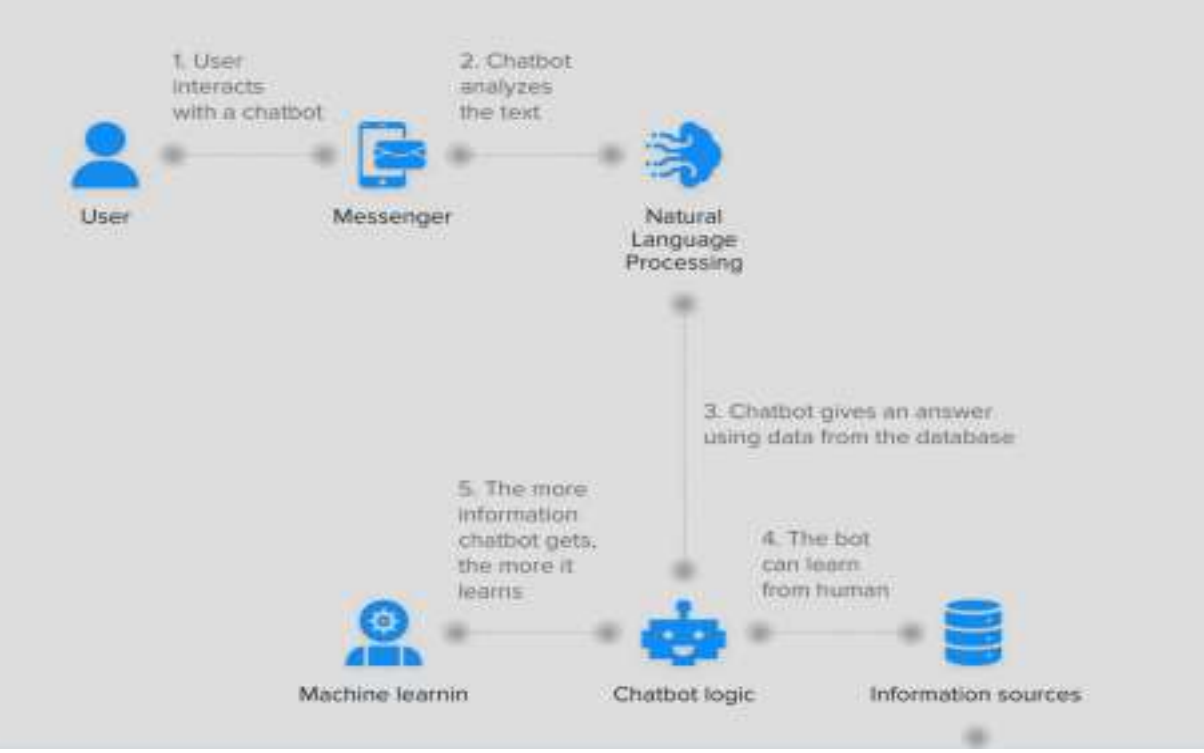
Format Cloudy's responses to match the conventions and capabilities of each messaging platform. For example, Slack messages may support rich text formatting and attachments, while Facebook Messenger might allow buttons and quick replies.

9. Testing and Debugging:

Thoroughly test the integration on each messaging platform to ensure that Cloudy behaves as expected. Debug and resolve any issues that may arise during testing.

10. Deploy and Monitor:

Once integration is complete and thoroughly tested, deploy Cloudy on the messaging platforms. Monitor its performance and user interactions to gather insights and make improvements over time.



Platform Integration:

- **Choose Integration Framework:** I'd select a multi-channel integration framework or chatbot platform capable of integrating with popular messaging platforms like Facebook Messenger and Slack.

- **Set Up Platform-Specific Connectors:** To facilitate integration, I'd configure connectors or adapters for each platform, closely following their developer documentation.

- **Handle Authentication and Authorization:** To ensure a secure integration, I'd make certain the chatbot is properly authenticated and authorized to interact with these platforms.

- **Routing and Message Parsing:** To route messages correctly and understand user inputs, I'd implement logic for message routing and parsing.

- Adapting responses to match each platform's capabilities, including text formatting and rich media, is a crucial aspect of the integration process.

Deployment:

- **Deploy the Chatbot:** After extensive testing and optimization, I'd deploy the chatbot to the selected messaging platforms or the cloud application to make it accessible to users.

- **Monitor and Maintain:** I would closely monitor the chatbot's performance and user interactions post-deployment, making regular updates and improvements to ensure it remains valuable and user-centric.

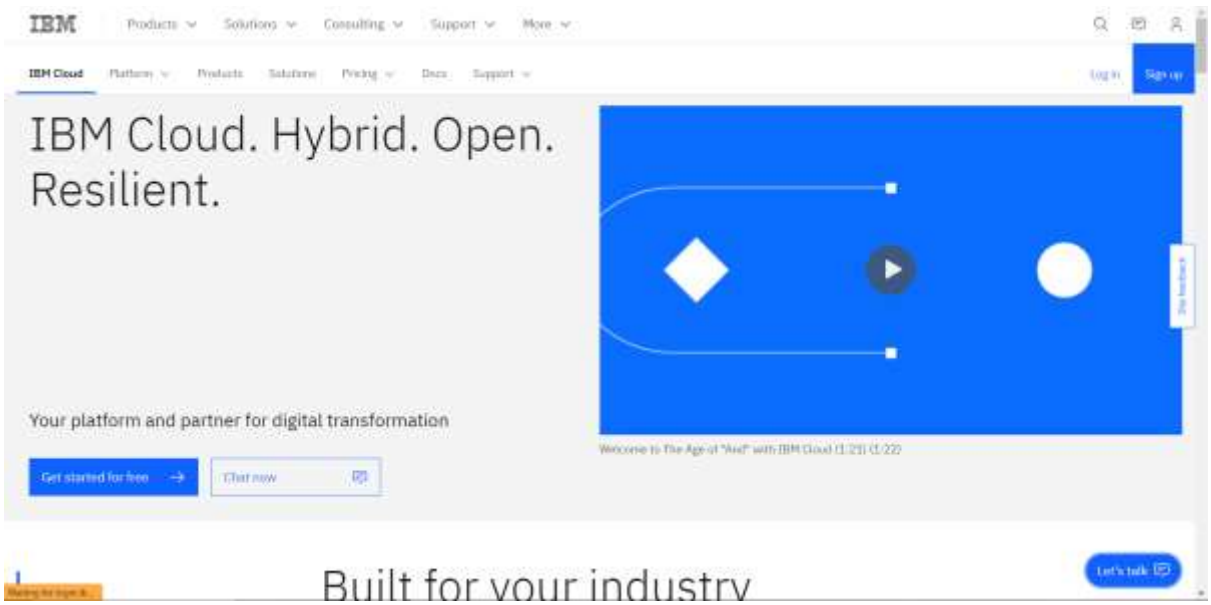
Creating a chatbot in IBM Cloud :

Sign Up for IBM Cloud:

1. Open a web browser and go to the IBM Cloud website: https://cloud.ibm.com/.

2. Click on the "Sign up" button, which is typically located in the top right corner.

3. It will be prompted to create an IBM ID. Follow the instructions and provide the required information to create your IBM Cloud account.



Create an IBM Watson Assistant Service:

1. After logging in, it will be taken to the IBM Cloud dashboard.

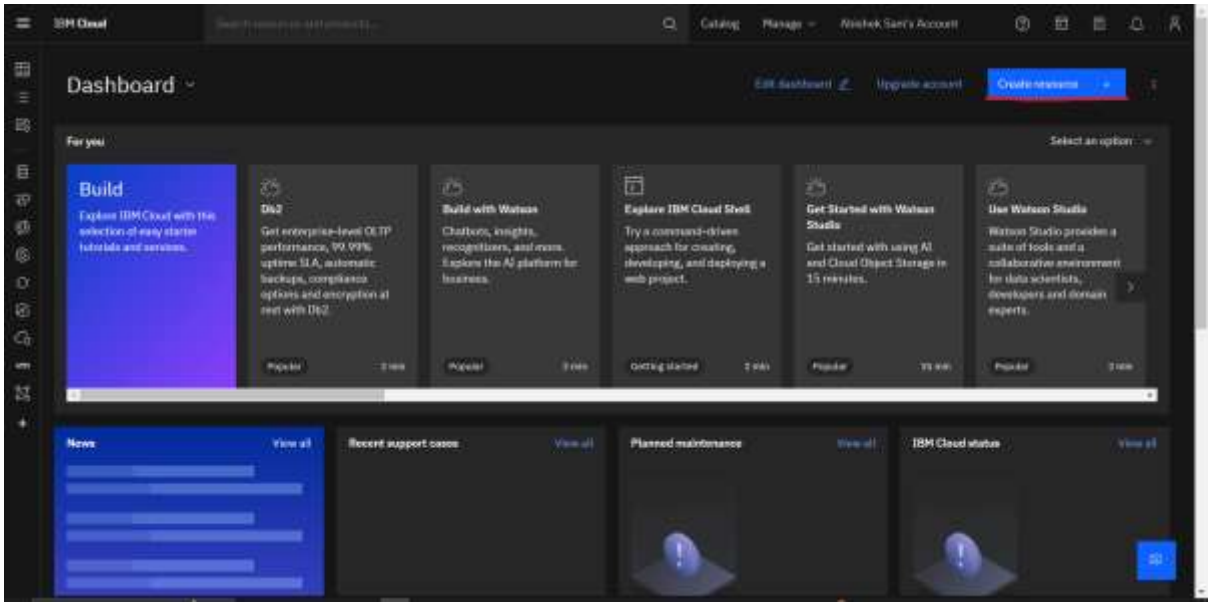
- Click on "Create Resource" or "Create a resource" from the dashboard.

- In the catalog, search for "Watson Assistant" or select the "AI" category to find it.

2. Click on the "Watson Assistant" service to configure it.

3. Fill out the required information, such as the service name and tags, and click "Create" or "Create Service."

4. Once the service is created, you'll be redirected to the Watson Assistant service dashboard.



Set Up Watson Assistant:

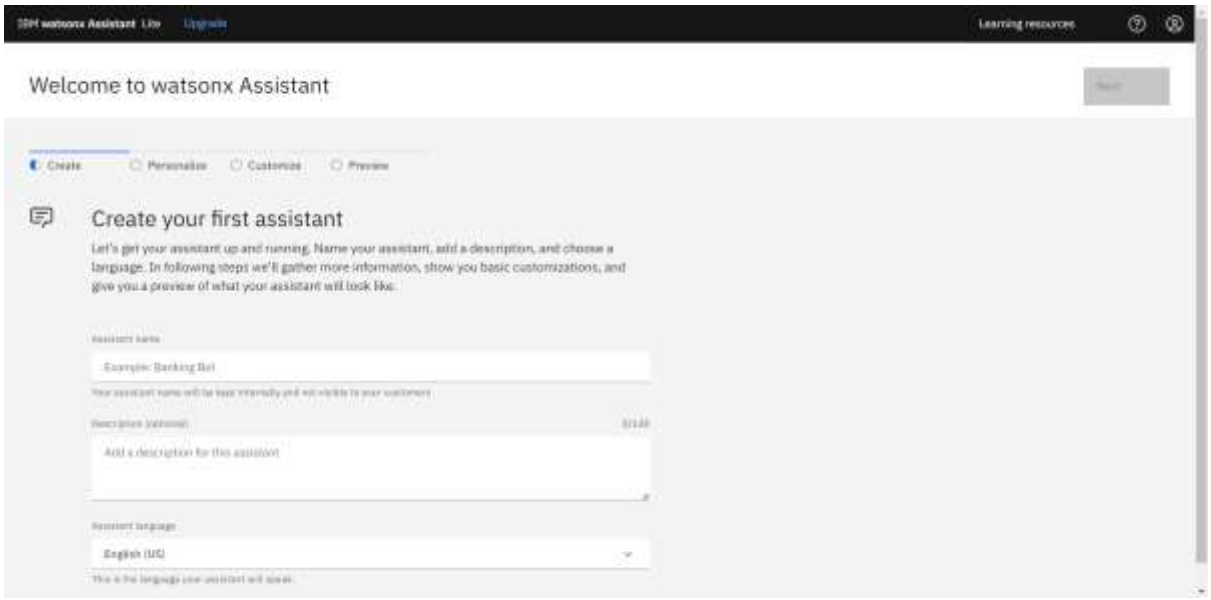
1. In the Watson Assistant dashboard, create and manage assistant instances. Click "Create assistant" to create a new chatbot instance.

2. Configure the assistant by providing a name, description, and other details as needed.

3. Design the chatbot by adding intents, entities, and dialog flows to handle user interactions. We can use the Watson Assistant interface to create and train your chatbot.

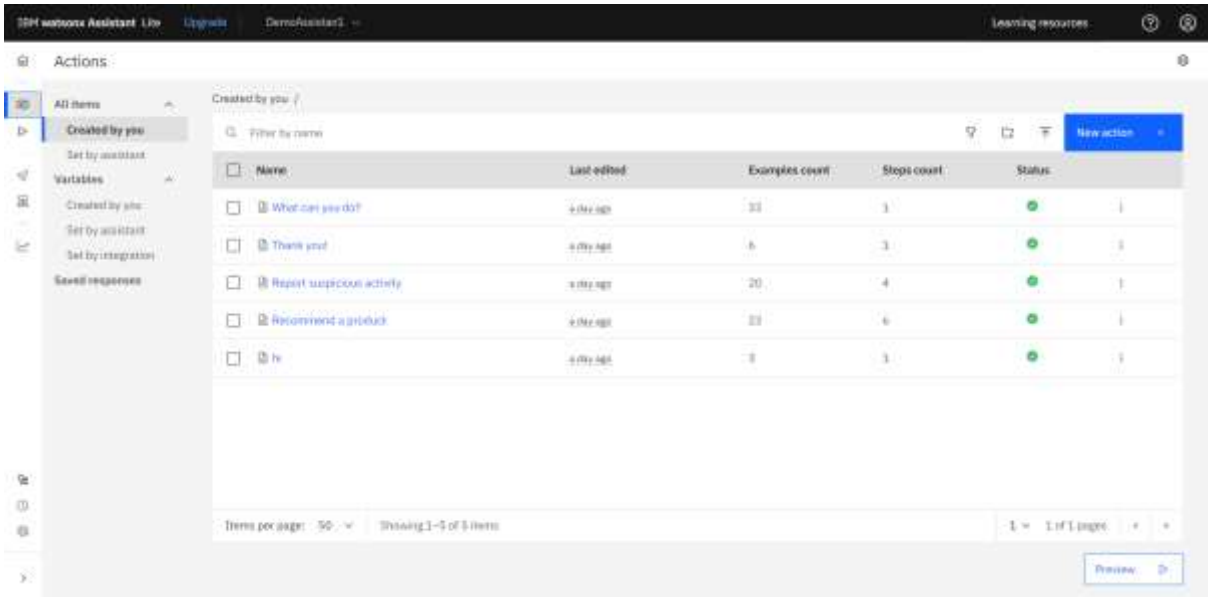
4. We can also integrate our chatbot with various channels and services, such as websites, mobile apps, or other communication platforms.

5. Once the chatbot is configured and trained, we can test it using the built-in tools in the Watson Assistant dashboard.



Connect Watson Assistant to the Application:

After configuring and testing our Watson Assistant chatbot, we can integrate it into our application or website using the provided API keys and endpoints.



INTEGRATION PART :

After creating a chatbot , we have to integrate this chatbot in Facebook Messenger.

Step 1: Create a Facebook Page:

1.Create a Facebook Page:

- If you don't have a Facebook Page for your chatbot, create one by visiting https://www.facebook.com/pages/create.

Step 2: Set Up Facebook Developer Account:

1.Create a Facebook App:

- Go to the Facebook for Developers website: https://developers.facebook.com/.

- Click on "Get Started" and create a new Facebook App.

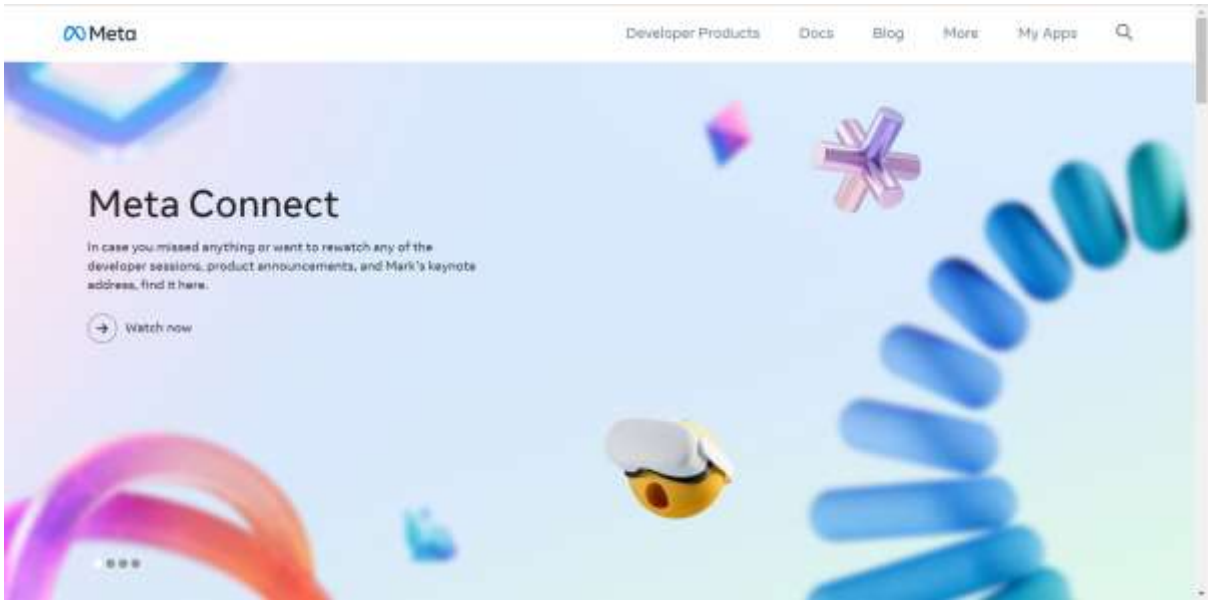
2.Configure Your App:

- In your Facebook App's dashboard, configure the settings:

  - Add a Facebook Page that you created in the previous step to your app.

  - Under "Products," add the "Messenger" product to your app.

  - Configure the "Messenger" product settings, including generating an App Secret and a Page Access Token.



Step 3: Connect Facebook and Watson Assistant:

1.Create a Webhook:

- In your Facebook App's dashboard under "Messenger," locate the "Webhooks" section.

- Click on "Setup Webhooks."

2.Configure Webhook:

- Provide your Watson Assistant webhook URL as the Callback URL. This is the endpoint where Facebook will send incoming messages.

- Enter the verification token, which you'll need to verify your webhook from your chatbot server.

3.Subscribe to Events:

- Subscribe to the necessary events (e.g., `messages`, `messaging_postbacks`) in the "Webhooks" settings.

Step 4: Develop a Chatbot Server:

1.Create a Server:

- Develop a server that listens for incoming POST requests from Facebook Messenger. You can use Node.js, Python, or any programming language of your choice.

- Use a web framework (e.g., Express.js for Node.js) to handle incoming requests.

2.Handle Webhook Events:

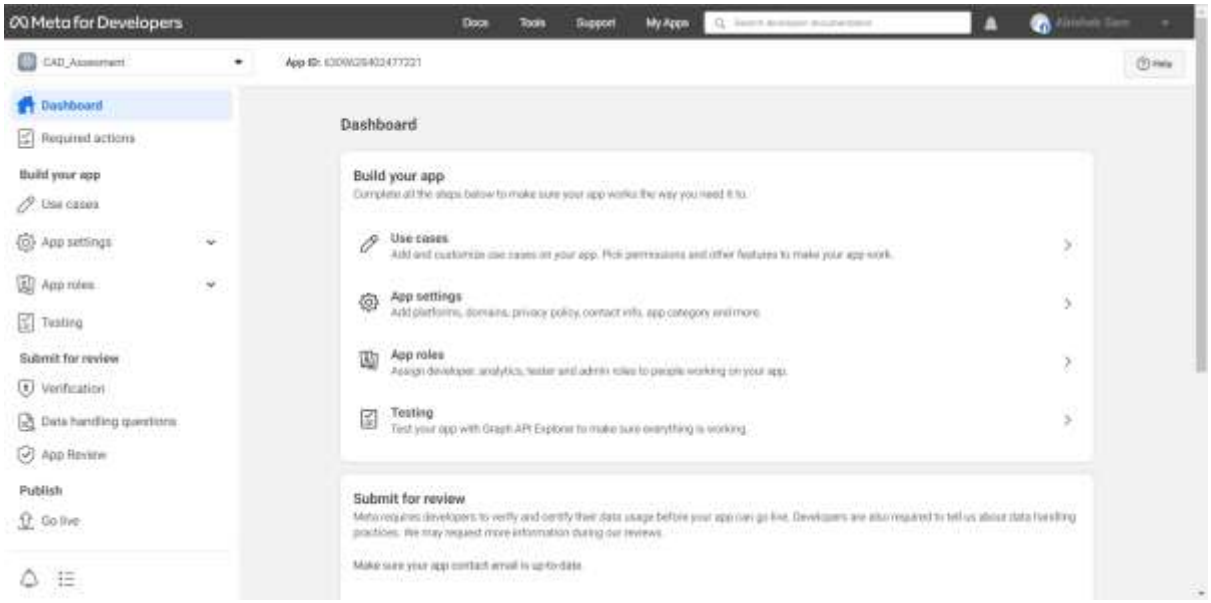
- Create logic to handle incoming webhook events from Facebook, including user messages.

3.Integrate with Watson Assistant:

- Use the Watson Assistant SDK to send user messages to your Watson Assistant service and receive responses.

4.Send Responses to Facebook:

- Send chatbot responses back to Facebook Messenger using the Page Access Token.



Step 5: Test and Deploy:

1.Test Your Chatbot:

- Test your chatbot by sending messages to the associated Facebook Page to ensure it works as expected.

2.Deploy Your Server:

- Deploy your server to a production environment that is publicly accessible over HTTPS.

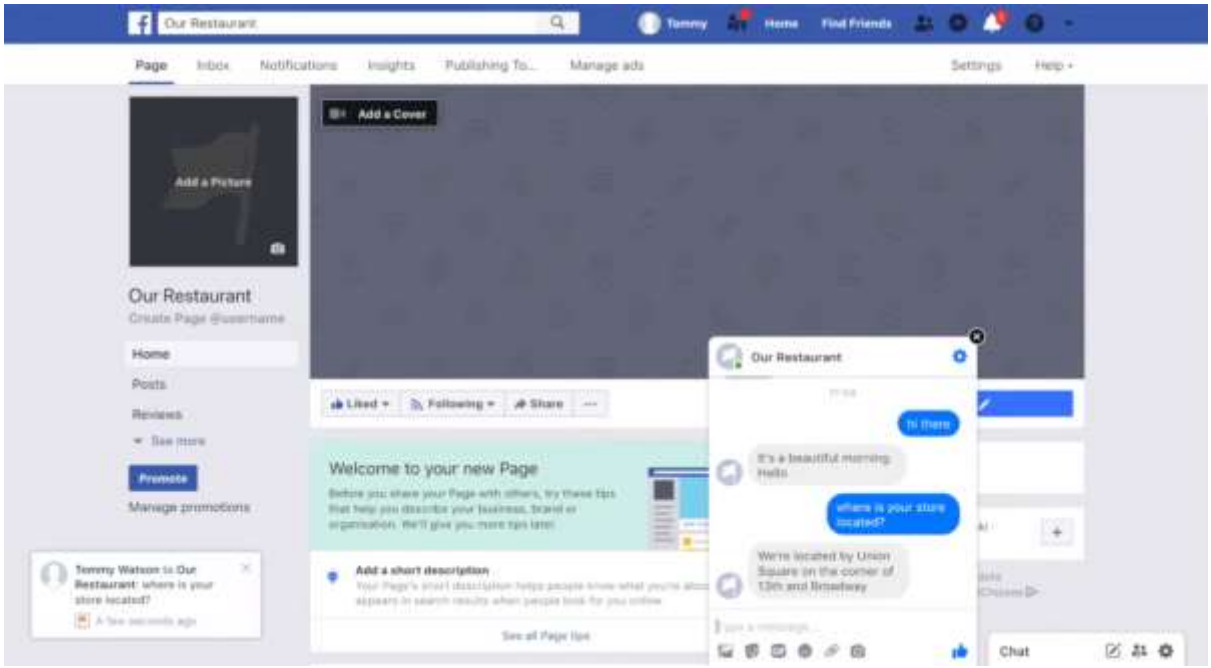
Step 6: Go Live:

1.Submit for Review:

- If you want to make your chatbot publicly available to a wider audience, you may need to submit your Facebook App and chatbot for review by Facebook.

2.Approval and Go Live:

- Once Facebook approves your chatbot, it can go live, and users can start interacting with it on Facebook Messenger.



****