

Project Development Phase

Delivery of Sprint - 4

Date	04 November 2022
Team ID	PNT2022TMID03463
Project Name	AI-based discourse for Banking Industry

Integrating it with web chat:

The top screenshot shows the 'Web chat' configuration page. It includes a header with 'IBM Watson Assistant Lite', 'Upgrade', 'Banking assista...', 'Learning center', and a user icon. The main content area is titled 'Web chat' with a 'Draft' button. There are 'Close' and 'Save and exit' buttons. The configuration fields include: Assistant's name as known by customers (Watson Assistant), Primary color (#FFFFFF), Secondary color (#8e97c4), Chat header, User message bubble, Accent color (#8e97c4), and Significant and interactive objects. A placeholder for an avatar image is also visible.

The bottom screenshot shows the 'Web chat' configuration page with a 'Greeting message' field containing 'Hi! I'm a virtual assistant. How can I help'. Below this is a 'Conversation starters' section with a description: 'Help your customers get going. When selected, the text of these starters is sent as a message. Each should be something your assistant is trained and tested to answer.' A sidebar on the right shows a list of categories: 'Banking', 'Check account balance', and 'Loan query'. The 'Banking' category is selected. At the bottom, there is a 'Type something...' input field with a send button.

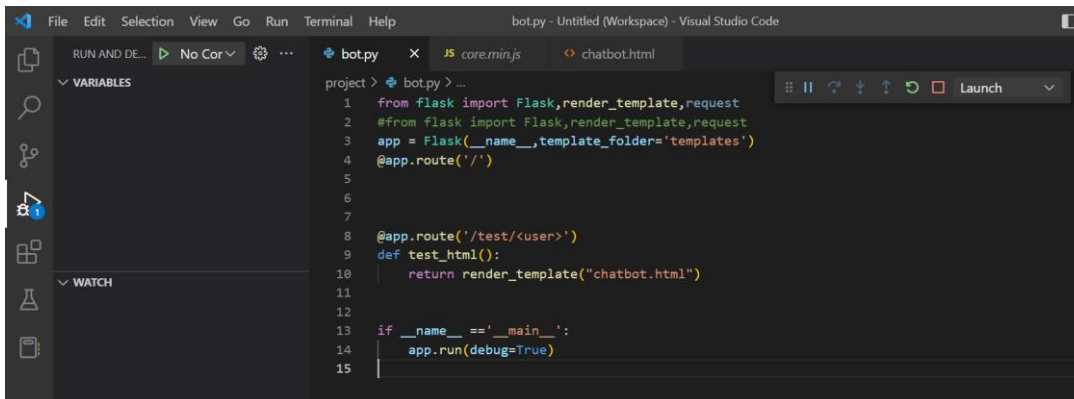
Creating Assistant & Integrate With Flask Web Page

You will be creating a banking bot in this activity that has the following capabilities

1. The Bot should be able to guide a customer to create a bank account.
2. The Bot should be able to answer loan queries.
3. The Bot should be able to answer general banking queries.
4. The Bot should be able to answer queries regarding net banking.
5. With the help of this bot, you can get all the required details related to banking.

Let us build our flask application which will be running in our local browser with a user interface. In the flask application, users will interact with the chatbot, and based on the user queries they will get the outcomes.

Build Python Code

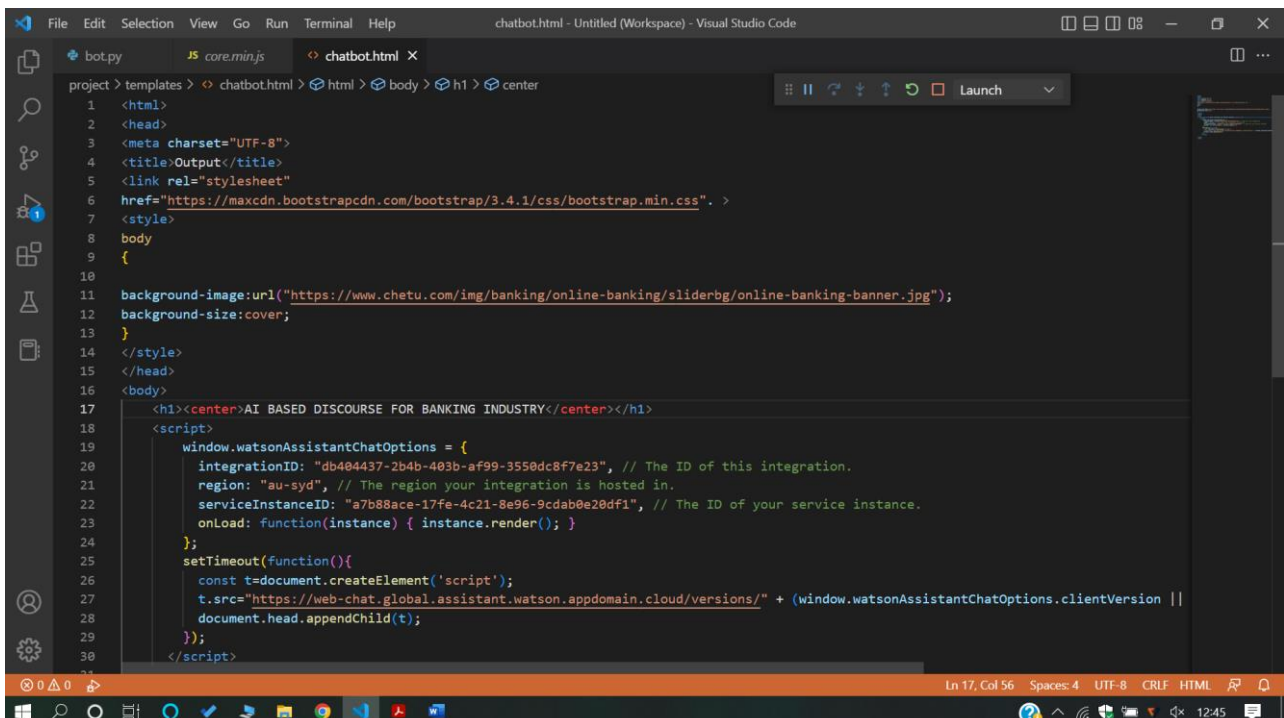


The screenshot shows the Visual Studio Code editor with a Python file named `bot.py` open. The code is a Flask application that serves a template named `chatbot.html`. The code is as follows:

```
1 from flask import Flask,render_template,request
2 #from flask import Flask,render_template,request
3 app = Flask(__name__,template_folder='templates')
4 @app.route('/')
5
6
7
8 @app.route('/test/<user>')
9 def test_html():
10     return render_template("chatbot.html")
11
12
13 if __name__ == '__main__':
14     app.run(debug=True)
15
```

Build HTML Code

1. We use HTML to create the front-end part of the web page..
2. Here, we have created 1 HTML page-Chatbot.html.
3. Chatbot.html displays the home page which integrates with Watson Assistant..
4. A simple HTML page is created. Auto-generated source code from IBM Watson Assistants is copied and pasted inside the body tag.

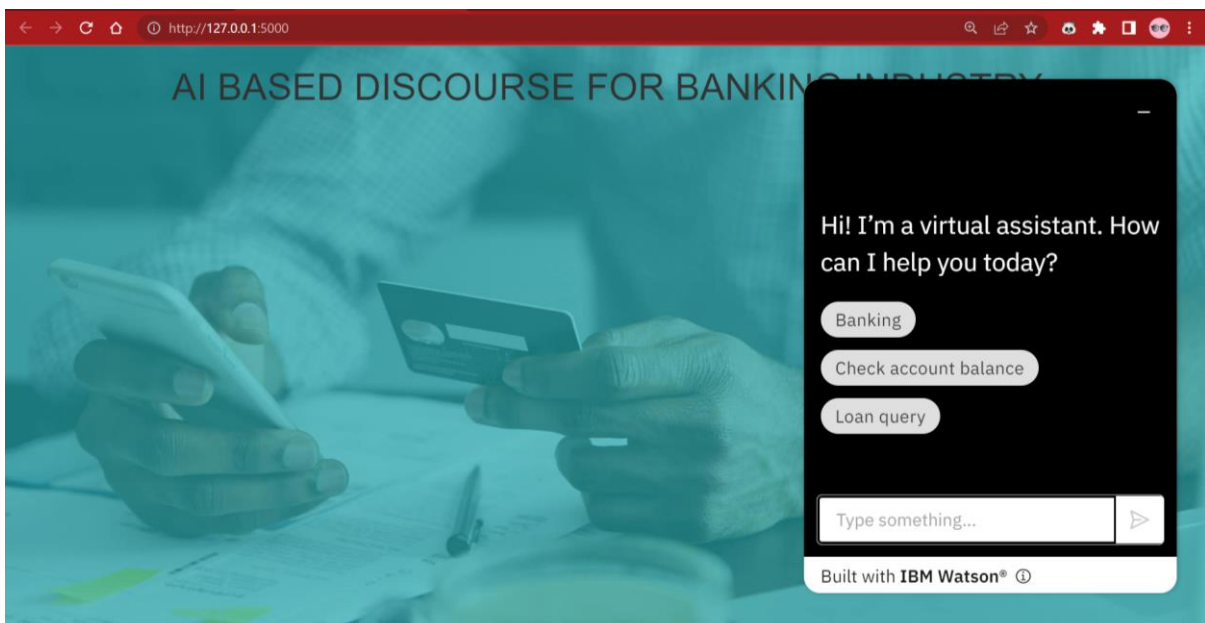


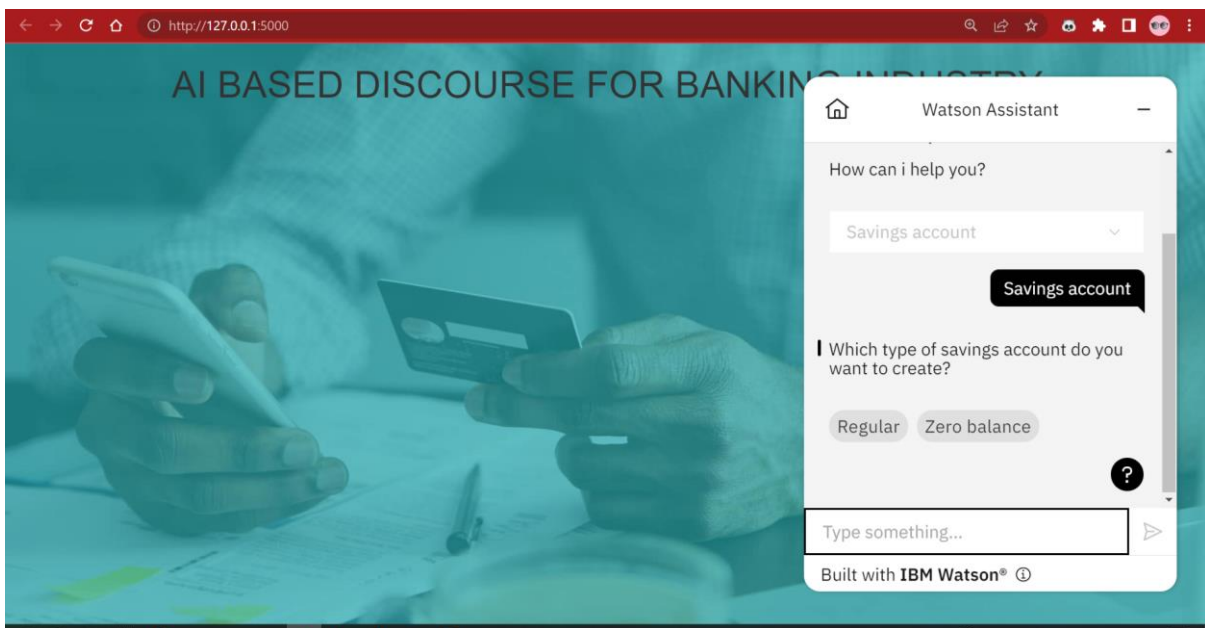
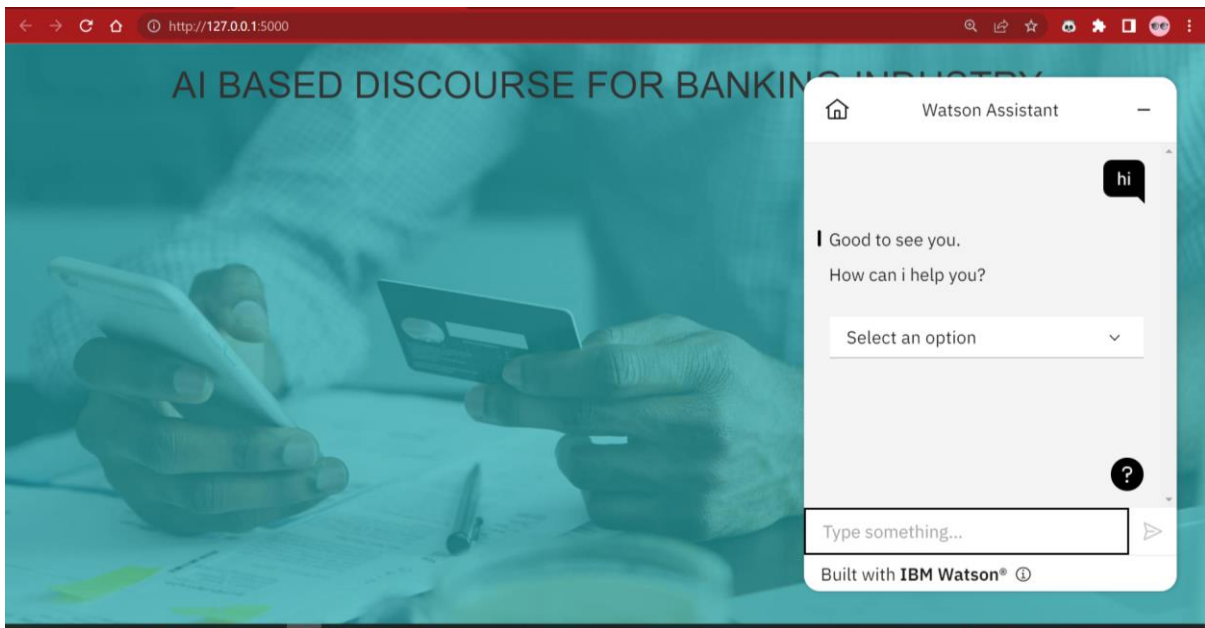
The screenshot shows the Visual Studio Code editor with an HTML file named `chatbot.html` open. The code is an HTML document that includes a Bootstrap CSS link and a script to integrate with Watson Assistant. The code is as follows:

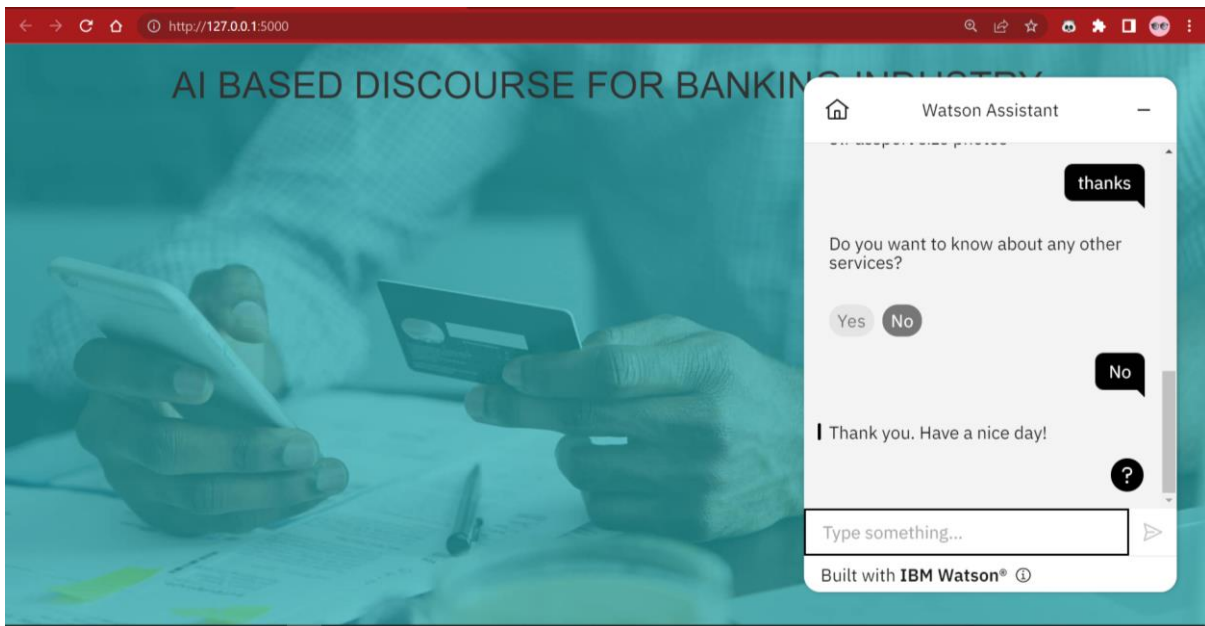
```
1 <html>
2 <head>
3 <meta charset="UTF-8">
4 <title>Output</title>
5 <link rel="stylesheet"
6 href="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/css/bootstrap.min.css">
7 <style>
8 body
9 {
10
11 background-image:url("https://www.chetu.com/img/banking/online-banking/sliderbg/online-banking-banner.jpg");
12 background-size:cover;
13 }
14 </style>
15 </head>
16 <body>
17 <h1><center>AI BASED DISCOURSE FOR BANKING INDUSTRY</center></h1>
18 <script>
19     window.watsonAssistantChatOptions = {
20         integrationID: "db404437-2b4b-403b-af99-3550dc8f7e23", // The ID of this integration.
21         region: "au-syd", // The region your integration is hosted in.
22         serviceInstanceID: "a7b88ace-17fe-4c21-8e96-9cdab0e20df1", // The ID of your service instance.
23         onload: function(instance) { instance.render(); }
24     };
25     setTimeout(function(){
26         const t=document.createElement('script');
27         t.src="https://web-chat.global.assistant.watson.appdomain.cloud/versions/" + (window.watsonAssistantChatOptions.clientVersion ||
28 document.head.appendChild(t);
29     });
30 </script>
```

Run the application:

It will run on localhost:5000







Preview of chatbot:

<https://web-chat.global.assistant.watson.appdomain.cloud/preview.html?backgroundImageURL=https%3A%2F%2Fau-syd.assistant.watson.cloud.ibm.com%2Fpublic%2Fimages%2Fupx-a7b88ace-17fe-4c21-8e96-9cdab0e20df1%3A%3A530eedf1-9c42-4bb9-9aa8-f4cc5c1c5d6b&integrationID=db404437-2b4b-403b-af99-3550dc8f7e23®ion=au-syd&serviceInstanceID=a7b88ace-17fe-4c21-8e96-9cdab0e20df1>