

## AI BASED DISCOURSE FOR BANKING INDUSTRY

**Team ID:** PNT2022TMID33460

### **TEAM**

#### **MEMBERS:**

VISWAS M [TL] - 922519104182

SIVAPRAKASH C [TM1] - 922519104149

KAVIYARASU E [TM2] - 922519104076

MUHAMAD UMAR S [TM3] - 922519104097

### **INTRODUCTION:**

#### **Overview:**

- Industries are forced to evolve and update their practices due to technological advances and the contemporary market. The banking sector is one of the most developed sectors and is always looking for the latest technological solutions that improve its efficiency.
- Net banking websites are complex and involve navigating through a lot of pages to find the information you need. Bank staff undergoes a lot of stressful situations when communicating with clients directly. Such situations can be avoided gracefully by using chatbots.
- Only 32% of companies in the finance industry currently use AI chatbots, and 37% are planning to start using them within 18 months said a report from Salesforce. This results in a potential growth rate of 118% which indicates the demand in the industry.
- A smart chatbot takes a query from the user in natural language and gives the appropriate response for the same. This paper aims to discuss the relevance of

chatbots in the banking sector and explore how chatbots can be implemented using natural language processing techniques that can be used in the banking industry.

## **LITERATURE SURVEY:**

### **Existing Problem:**

- This paper [1] presents the use of the RASA framework for building smart context-remembering chatbots, it also describes how Rasa NLU works and how its performance is elevated by using intent recognition and entity extraction. It also compares the accuracies of entity extraction using Rasa NLU and a NN, results show Rasa NLU performs better to extract entities when whole sentences are provided as compared to neural networks which require segmented inputs. This paper discusses Rasa by implementing a chatbot related to the finance domain, using which the users can inquire about stock-related information.
- RASA NLU can introduce a vital component in intelligent chatbot systems. We can compose the system to extract the entity after intent recognition. This can be further improved for complicated sentences and more entities.
- This paper [2] briefly discusses advancements in the field of AI and how this has led to major shifts in some organizations about how they operate. It further mentions how the banking industry has moved to use chatbots for providing an interface to customers so that they can have an assistant throughout the day for service. This paper also gauges the ability of current chatbots to provide all the services that a user needs.
- It includes several strategies for managing dialogue in the banking and finance industry based on ontology. Although further use of AI can make the chatbot not only respond to questions but also self-learning to improve itself in more stages, improving user service quality and also reducing human load.

### **Proposed solution:**

- The solution to the problem is Artificial intelligence in the banking sector makes banks efficient, trustworthy, helpful, and more understanding. It is strengthening the competitive edge of modern banks in this digital era. The growing impact of AI in banking sector minimizes operational costs improves customer support and process automation.
- Nearly 40% to 50% of financial and banking service providers are using AI in their processes to harness the power of next-generation AI capabilities. The companies believe that AI is the future of banking sector which can perform a

range of banking operations in faster, easier, and more secure ways.

- AI banking Chatbots help customers in many ways. AI-based chatbot service for financial industry is one of the significant use cases of AI in banking sector. AI chatbots in banking are modernizing the way how businesses provide services to their customers.
- AI chatbots in the banking industry can assist customers 24\*7 and give accurate responses to their queries. These chatbots provide a personalized experience to users.
- AI chatbots in banking is providing a better customer experience.
- Hence, AI chatbots for banking and finance operations let banks attract customer attention, optimize service quality, and expand the brand mark in the market.

## THEORETICAL ANALYSIS:

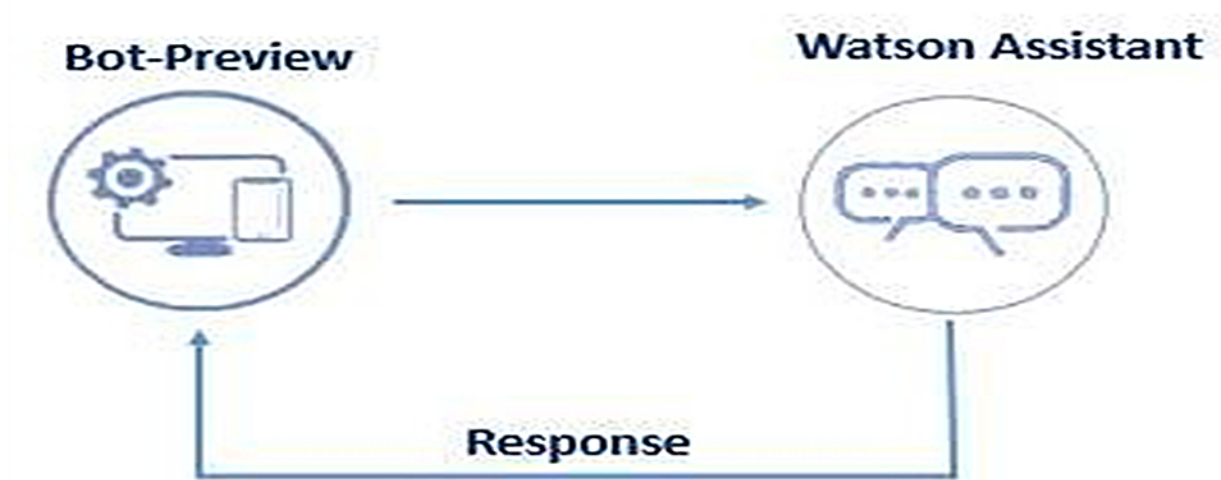
### Services Used:

- IBM Watson Assistant

# Watson Assistant



Block diagram:



## Hardware / Software designing:

To complete this project, you should have the following software and packages.

### Softwares:

- Visual studio code
- IBM Watson studio

### Packages:

- Flask

## FLOWCHART:

To accomplish the above task, you must complete the below activities and tasks:

- Create IBM Services.
- Creating skills & Assistant for Chatbot.
- Creating Savings account action.
- Creating Current account action.

Creating Loan account action.

Creating a general query action.

Creating a Net banking action.

Create an HTML web page.

Integrate the Watson Chatbot with a web page.

## **ADVANTAGES & DISADVANTAGES:**

### **Advantages:**

1. Round-the-clock service.
2. Brand Consistency.
3. Increased Productivity.
4. Reduced Staffing Needs.
5. Consistent Response Rate and Availability.
6. Helps with Fraud Prevention.
7. Chats can be saved.
8. Lower costs.

### **Disadvantages:**

9. Questions must be programmed beforehand.
10. Impersonal
11. Must keep information up-to-date.

12. Technology issues.

13. Needs additional measures to protect identities.

## APPLICATIONS:

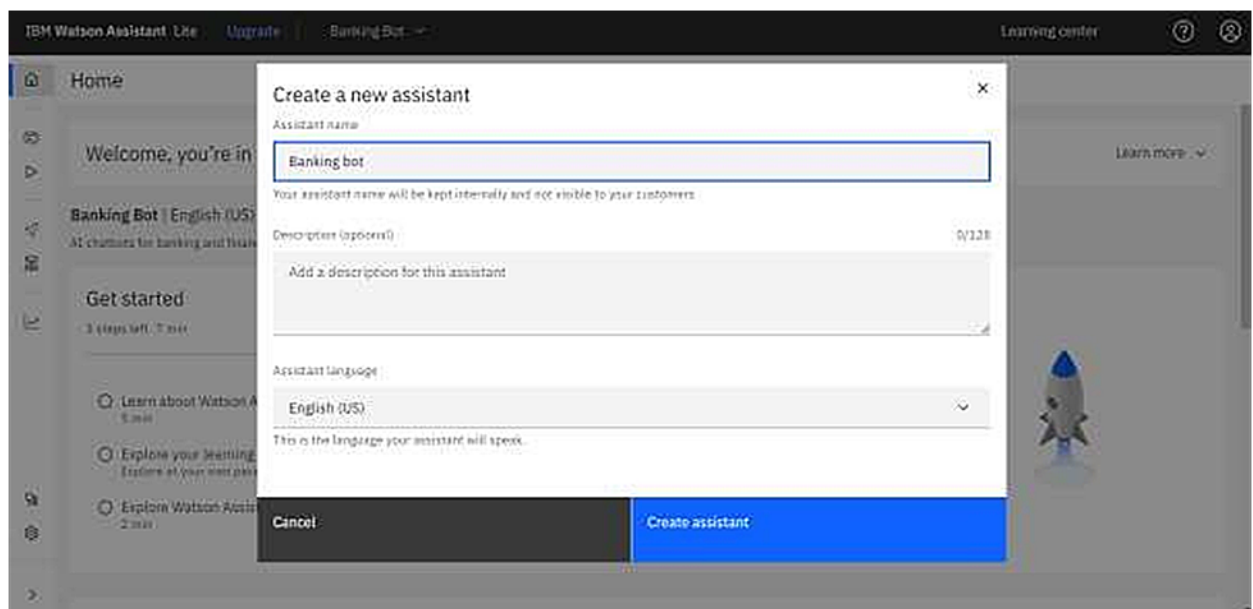
14. Banking chatbots have all the data to predict the spending habits of customers and help them keep their finances on track.

## APPENDIX:

### Create IBM Service

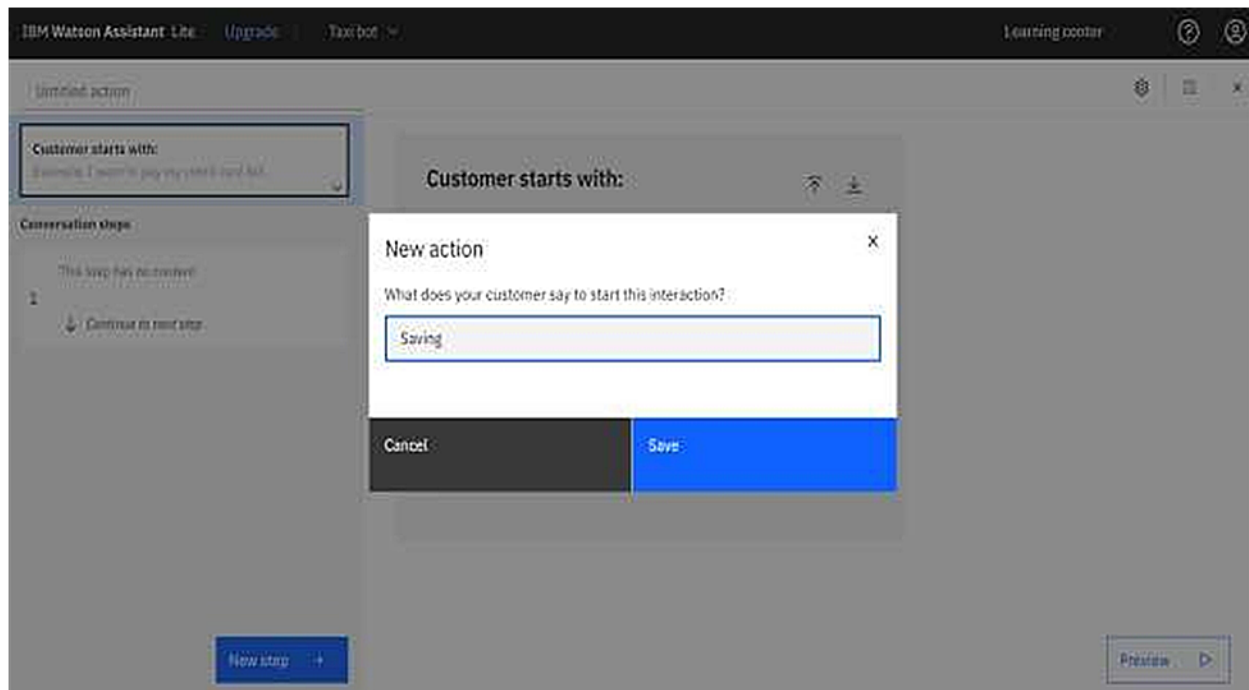
In this activity, you will be creating the Necessary IBM service. The following are the service that you have to create.

- Watson Assistant

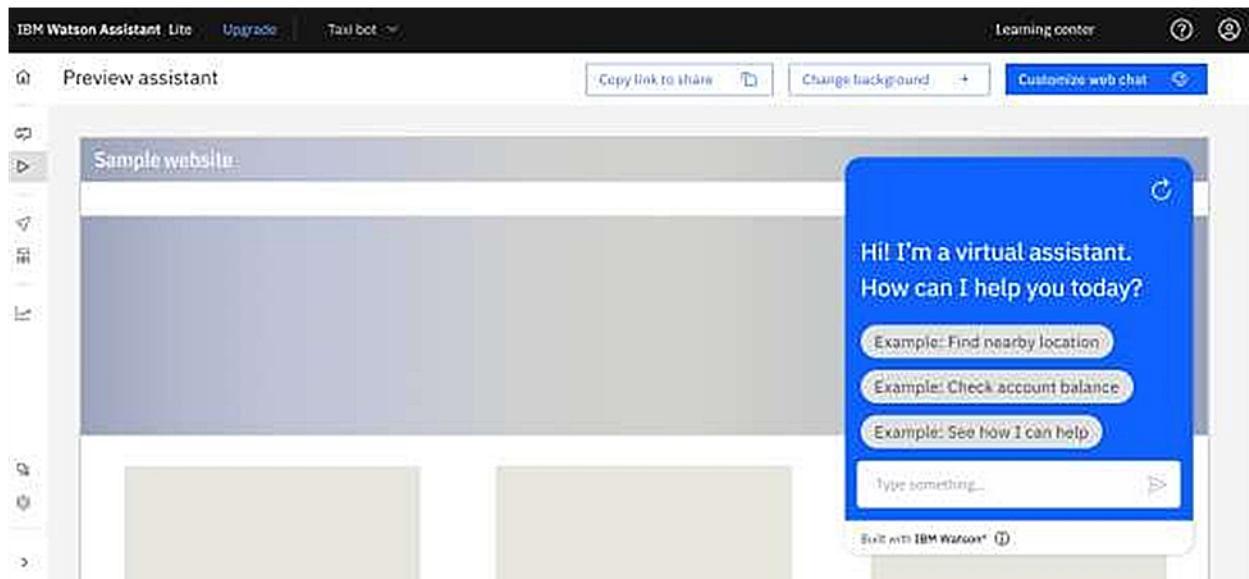
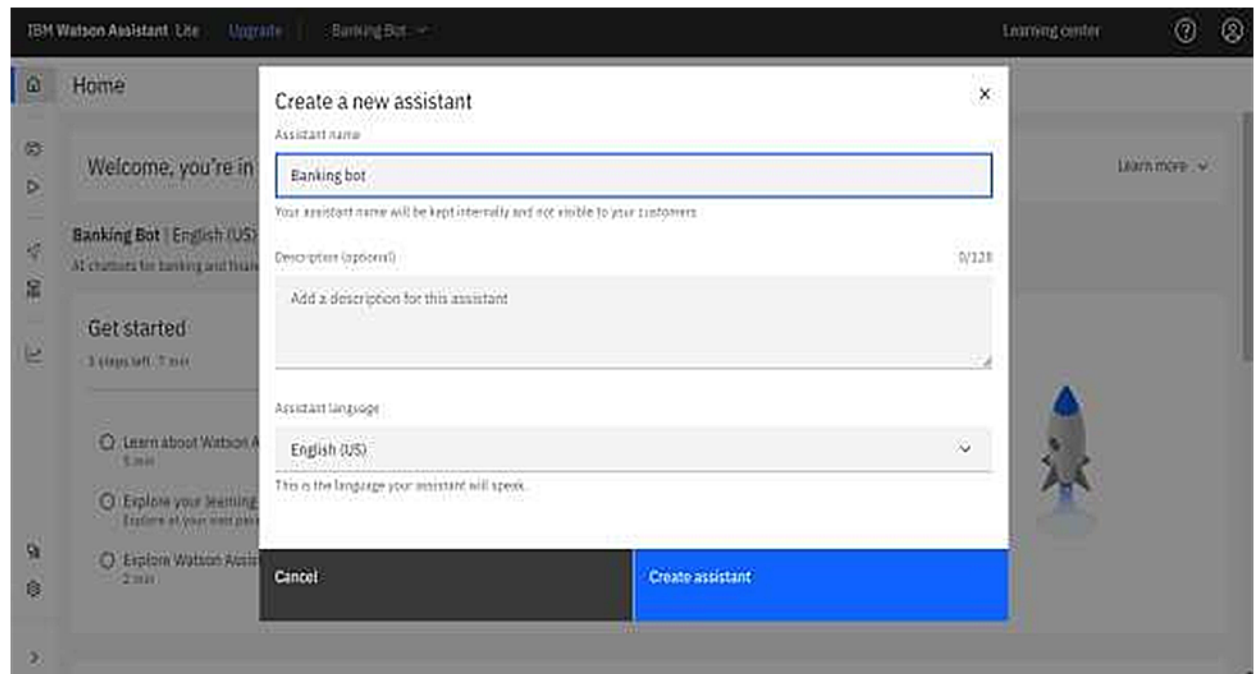


## Creating Skills & Assistant For Chatbot

Skills are nothing but actions and steps. Steps are the subset of actions where conversations are built and Assistant is used to integrate skills.



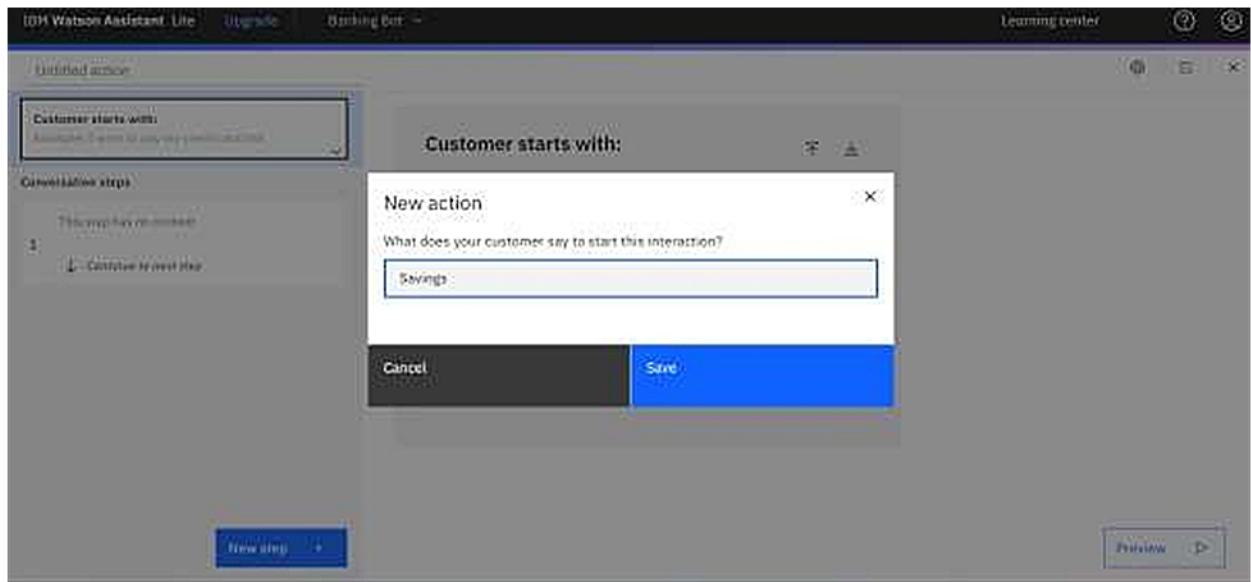
A default template chatbot is created. Need to add actions.



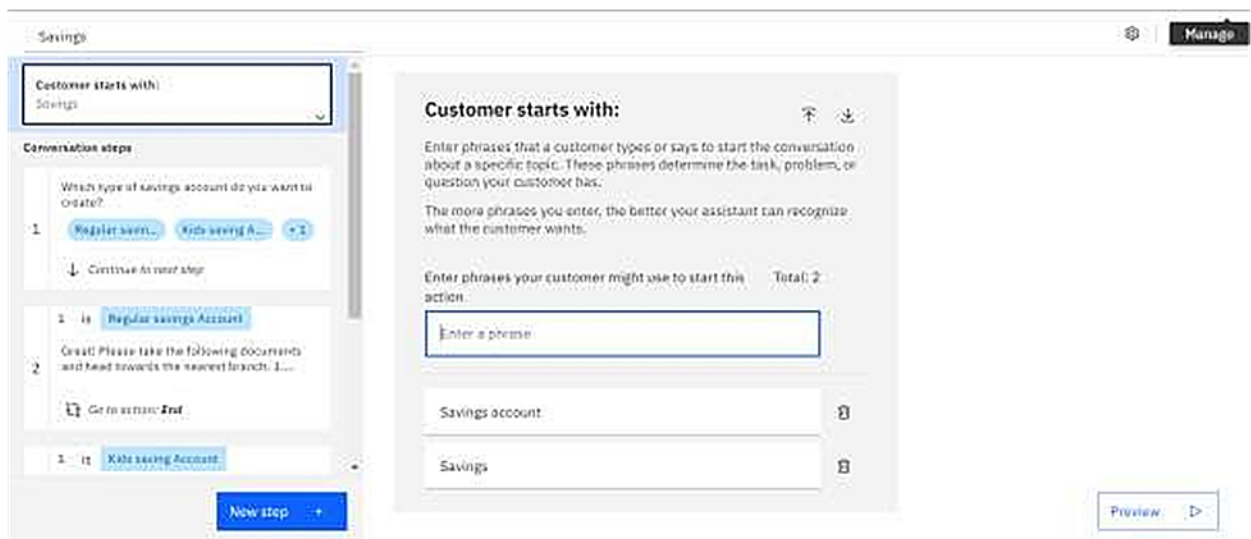
## Creating Saving Account Action



Create a saving account in IBM Watson. Create new **Action** Saving.

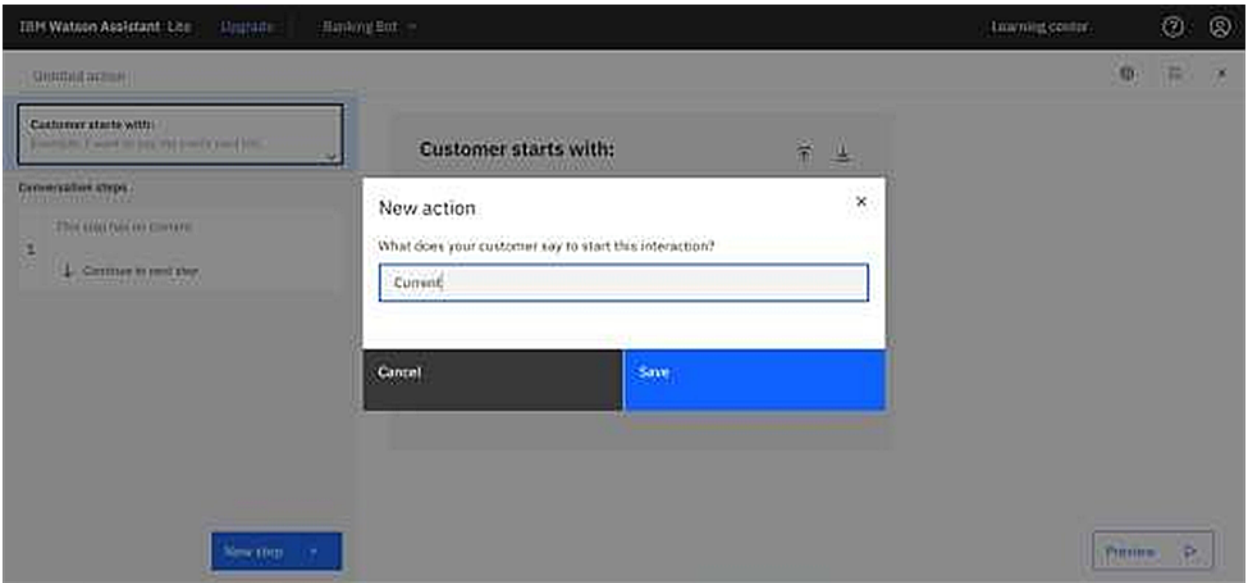


Add steps in savings action.

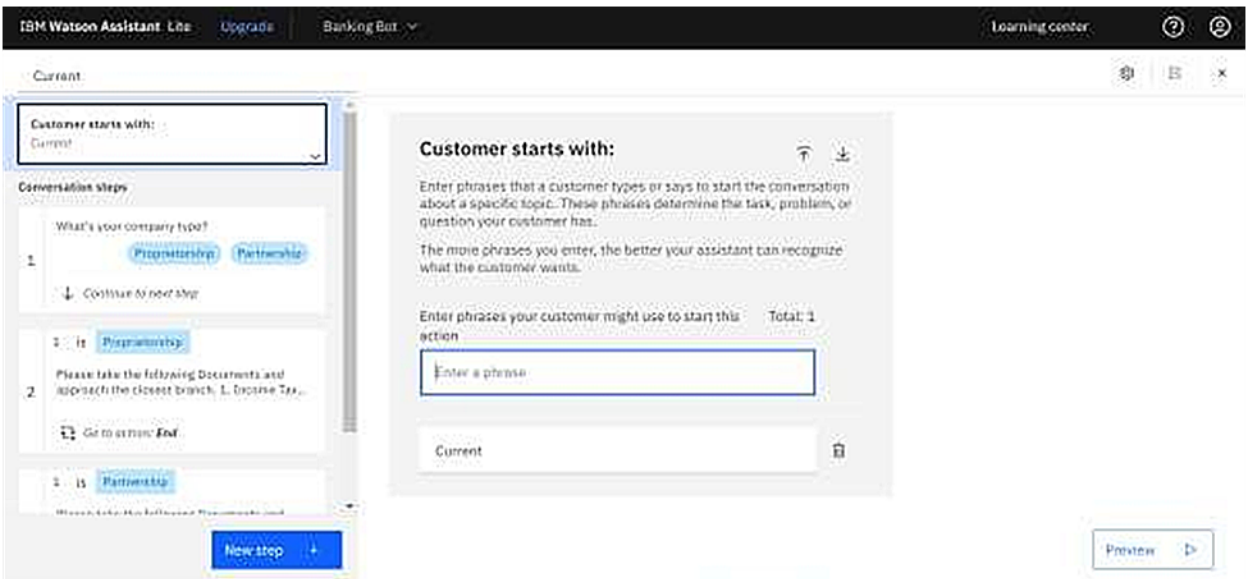


## Creating Current Account Action

Create a new **Action** Current for the current account action.



Add steps in savings action.



## Creating Loan Account Action

Loan action is created with the necessary steps.

The screenshot shows the IBM Watson Assistant interface for creating a 'Loan' action. The top bar includes 'IBM Watson Assistant', 'Life', 'Upgrade', 'Banking Bot', and 'Learning center'. The left sidebar shows the 'Loan' action being edited. The main area is divided into two panels. The left panel, titled 'Conversation steps', shows a sequence of steps: 1. 'What type of loan are you looking at?' with buttons for 'Vehicle loan', 'Business loan', and '+3'. 2. 'To be eligible for a house loan, please contact our bank service providers with all existing...'. 3. 'Go to action: End'. The right panel, titled 'Customer starts with:', provides instructions on how to start the conversation and a list of phrases that can trigger the action. The phrases listed are 'Loan' and 'How to apply loan?'. A 'Preview' button is located at the bottom right.

## Creating General Query Action

General query action is created with the necessary steps.

The screenshot shows the IBM Watson Assistant interface for creating a 'Query' action. The top bar includes 'IBM Watson Assistant', 'Life', 'Upgrade', 'Banking Bot', and 'Learning center'. The left sidebar shows the 'Query' action being edited. The main area is divided into two panels. The left panel, titled 'Conversation steps', shows a sequence of steps: 1. 'Select the general queries listed below' with buttons for 'CIBD', 'Bank Working Days', and '+3'. 2. 'The bank is open all days from Monday to Saturday from 9 am to 3 pm, with exceptions...'. 3. 'List of branches'. The right panel, titled 'Customer starts with:', provides instructions on how to start the conversation and a list of phrases that can trigger the action. The phrases listed are 'Query in general' and 'general'. A 'Preview' button is located at the bottom right.

## Creating Net Banking Action

Net banking action is created with the necessary steps.

The screenshot shows the IBM Watson Assistant console interface. At the top, there's a header with 'IBM Watson Assistant Lite', 'Upgrade', and 'Banking Bot'. On the right, there's a 'Learning center' link and user icons. The main area is titled 'Net Banking'. On the left, under 'Conversation steps', there's a list of steps: 1. 'What issues do you have regarding net banking?' with sub-phrases 'What are the...' and 'What is Net B...'. 2. 'What is Net Banking?' with a description 'The facility offered by the bank allows customers to use banking services over the...'. 3. 'How do I register for Net Banking?'. A 'New step' button is at the bottom left. On the right, the 'Customer starts with:' section explains that phrases determine the task and provides a text input field labeled 'Enter a phrase' with the example 'Net Banking' entered. A 'Preview' button is at the bottom right.

In addition to this greeting, end greeting, index, and end actions are also created.

New action +			
Name	Last edited	Status	
Current	2 days ago	✓	⋮
Index	2 days ago	✓	⋮
Register	3 days ago	✓	⋮
Greeting	2 days ago	✓	⋮
End Greeting	2 days ago	✓	⋮

			Q	↑	New action +
Name	Last edited	Status			
Net Banking	3 minutes ago	✓	...		
End	2 days ago	✓	...		
Loan	2 days ago	✓	...		
Query	a few seconds ago	✓	...		
Savings	16 minutes ago	✓	...		
Current	2 days ago	✓	...		
Items per page: 50 ▾			Showing 1–10 of 10 actions		
			1 ▾	1 of 1 pages	◀ ▶

## 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**

### **1: Importing Libraries**

The first step is usually importing the libraries that will be needed in the program.

```
from flask import Flask, render_template
```

Importing the flask module into the project is mandatory. An object of the Flask class is ourWSGI application. Flask constructor takes the name of the current module (name).

### **2: Creating our flask application and loading**

```
app = Flask(__name__)
```

### **3: Routing to the Html Page**

Here, the declared constructor is used to route to the HTML page created earlier.

The '/' route is bound with the bot function. Hence,when the home page of a web server is opened in the browser, the HTML page will be rendered.

```
@app.route('/')
def bot():
    return render_template('chatbot.html')
```

### Main Function

This is used to run the application in localhost.

### 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 Assistant is copied and pasted inside the body tag

### Run The Application

5. Open the anaconda prompt from the start menu.
6. Navigate to the folder where your app.py resides.
7. Now type the "python app.py" command.

8. It will show the local host where your app is running on `http://127.0.0.1.5000/`
9. Copy that localhost URL and open that URL in the browser. It does navigate to where you can view your web page.

## Source Code:

### index.html

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8" />
    <meta name="theme-color" content="#000000" />
    <meta
      name="description"
      content="Web site created using create-react-app"
    />

    <title>BankingAi</title>
  </head>
  <body>
    <noscript>You need to enable JavaScript to run this app.</noscript>
    <div id="root"></div>

  </body>
</html>
```

### App.js :

```
import React, {useState} from 'react';
import {Chat} from './Chat.jsx';
import './App.css';
```



```

function App() {

  const [showChat, setShowChat] = useState(false);

  const openChat = () => {
    setShowChat(true);
  };

  return (
    <div className="App">
      <header className="App-header">
        AI Based Discourse For Banking Industry
      </header>
      <div className='App-content'>
        Team Id : PNT2022TMID33460
        <br />
        Chat bot description :
        <ul>
          <li>The bot should be able to guid a customer</li>
          <li>The bot should be able to answer the loan queries</li>
          <li>The bot should be able to answer general loan queries</li>
          <li>The bot should be able to answer queries regarding net banking</li>
        </ul>
        {showChat ? <Chat setShowChat={setShowChat} /> : <span className='chatButton'
onClick={openChat}>Chat</span>}

      </div>
    </div>
  );
}

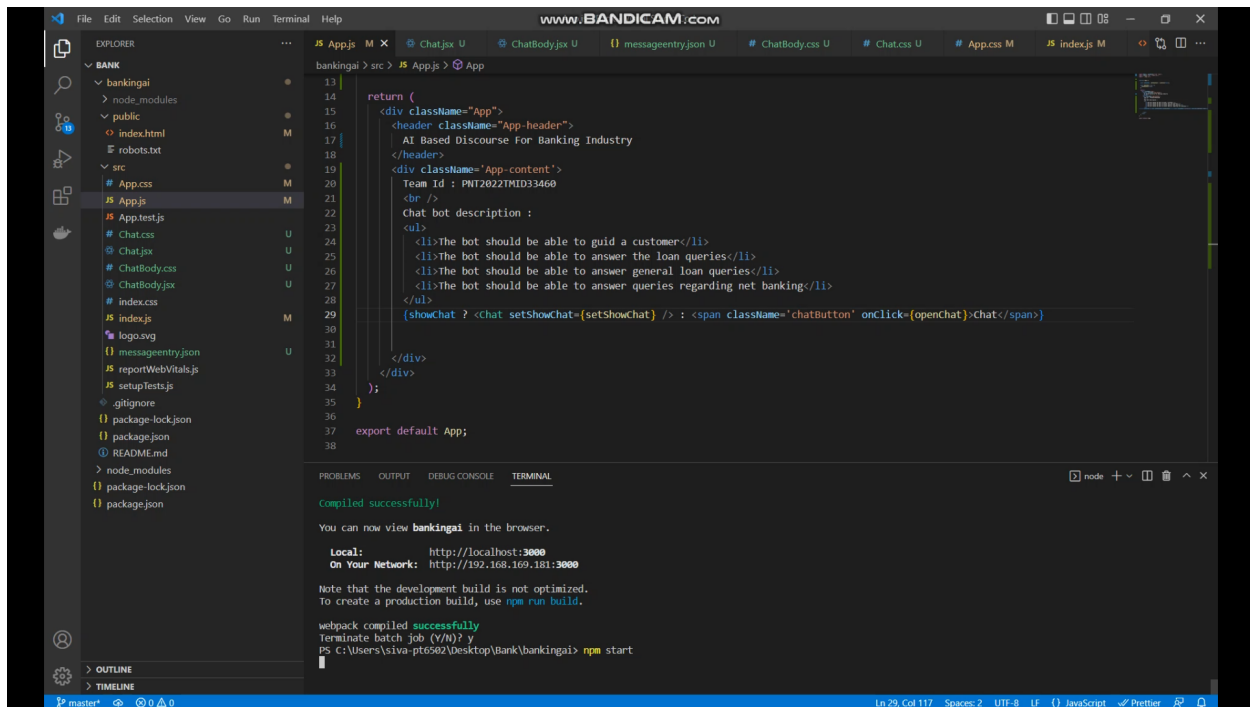
export default App;

```

## app.py

```
from flask import Flask,render_template
app=Flask(__name__)
@app.route('/')
def bank():
    return
render_template('Chatbot.html') if
__name__ == '__main__':
    app.run(debug = True)
```

## OUTPUT:



The screenshot shows a VS Code editor with the following components:

- Explorer:** A file tree on the left showing the project structure. The 'src' folder is expanded, showing files like App.css, App.js, App.test.js, Chat.css, Chat.js, ChatBody.css, ChatBody.js, index.css, index.js, logo.svg, messageentry.json, reportWebVitals.js, and setupTests.js.
- Editor:** The main workspace shows the 'App.js' file. It contains a React component definition for 'App' with a header and content section. The content section includes a chat bot description and a list of capabilities.
- Terminal:** The bottom panel shows the output of the 'npm start' command. It indicates that the application was compiled successfully and is now running on 'http://localhost:3000'.

```
13 |
14 |
15 | return (
16 |   <div className="App">
17 |     <header className="App-header">
18 |       AI Based Discourse For Banking Industry
19 |     </header>
20 |     <div className="App-content">
21 |       Team Id : PH172022TMD33460
22 |       <br />
23 |       Chat bot description :
24 |       <ul>
25 |         <li>The bot should be able to guid a customer</li>
26 |         <li>The bot should be able to answer the loan queries</li>
27 |         <li>The bot should be able to answer general loan queries</li>
28 |         <li>The bot should be able to answer queries regarding net banking</li>
29 |       </ul>
30 |       {showChat ? <chat setShowChat={setShowChat} /> : <span className='chatButton' onClick={openChat}>Chat</span>}
31 |     </div>
32 |   </div>
33 | );
34 |
35 | export default App;
36 |
37 |
38 |
```

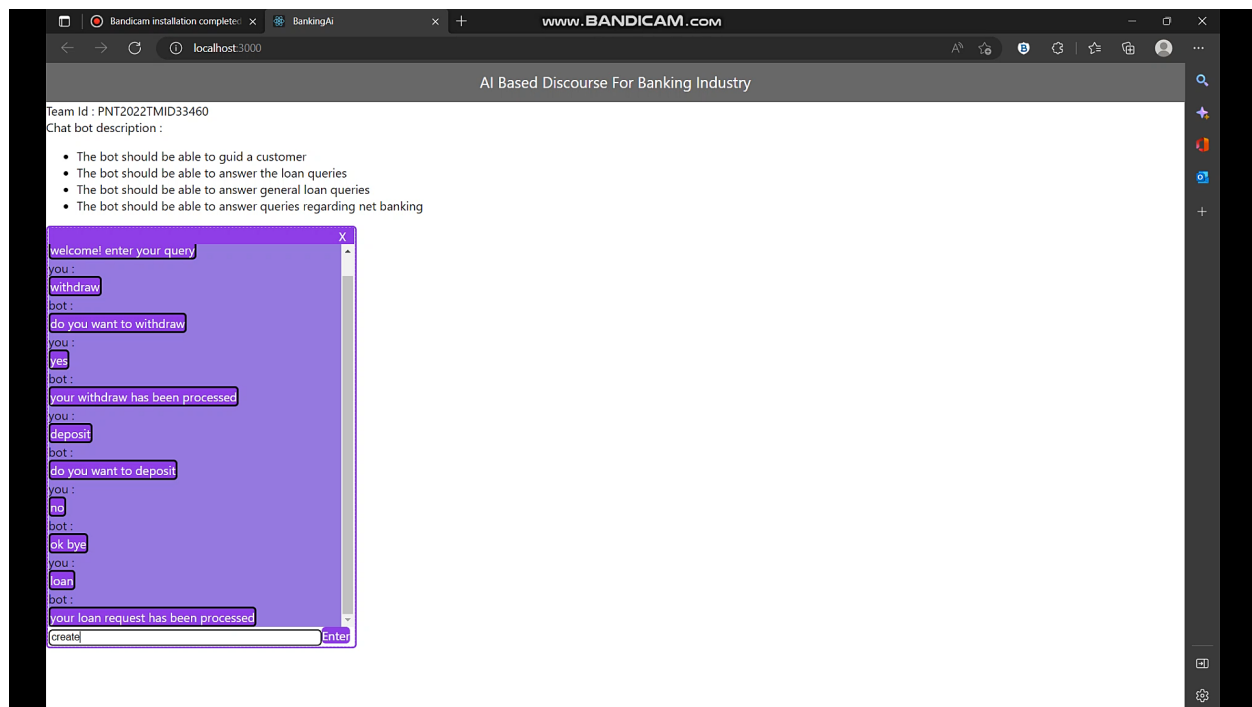
compiled successfully!

You can now view **bankingai** in the browser.

Local: http://localhost:3000  
On Your Network: http://192.168.169.181:3000

Note that the development build is not optimized.  
To create a production build, use `npm run build`.

webpack compiled successfully  
Terminate batch job (Y/N)? y  
PS C:\Users\siva-pt6502\Desktop\bank\bankingai> npm start



LINKS:

Youtube: [youtube.com/watch?v=C-y9YsQXQIU](https://www.youtube.com/watch?v=C-y9YsQXQIU)

GIT REPO : <https://github.com/IBM-EPBL/IBM-Project-39976-1660574006.git>

