

innovate

achieve

lead



BITS Pilani

Artificial Intelligence Assignment

Submitted by: Likhith Gatagat
2022MT13057

Bits, Pilani, WILP

Email: 2022MT13057@wilp.bits-pilani.ac.in

SSZC444 Artificial Intelligence

November 27, 2023



Custom Conversational AI agent (Chatbot) using OpenAI

Installations/Packages needed for this assignment:

- ☐ Node - v20.10.0
- ☐ NPM - 10.2.3
- ☐ Express - 4.18.2 - Framework we'll use to spin up a Node server
- ☐ Airtable - 0.11.6 -
- ☐ OpenAI - 3.3.0 - Node.js library for the OpenAI API

Pre-requisites for the assignment:

- ☐ OpenAI API Key
- ☐ Air Table Base Id
- ☐ Air Table API Key



Step 1 - Integrating OpenAI with Node.js/Express

- **Objective:** Communicate with OpenAI API to generate responses.
 - **Implementation:** Utilized Node.js and Express to set up an endpoint for sending requests to OpenAI.
 - **Key Points:**
 - ❑ Backend Setup: Created a Node.js/Express server (index.js) to handle API requests.
 - ❑ OpenAI Integration: Configured an endpoint to communicate with OpenAI's API.
 - ❑ Random Response Generation: Utilized the OpenAI API to generate random responses based on prompts.
 - **Impact:** Facilitated seamless communication between the backend and OpenAI's powerful language processing capabilities.
-

Step 1 – Code Snippet

A screenshot of the Visual Studio Code editor interface. The Explorer sidebar on the left shows a project named 'CHATGPTPENAI' with files: 'node_modules', '.env', 'index_embeddings.js', 'index.js' (selected), 'package-lock.json', and 'package.json'. The Search sidebar is also visible. The main editor area shows the content of 'index.js', which is a JavaScript file defining a REST API endpoint for '/ask'. The code uses 'dotenv' for environment variables, 'express' for the web framework, and the 'openai' package for API calls. It defines a 'Configuration' class and an 'OpenAIApi' class. The endpoint handler checks for a 'prompt' in the request body and uses 'createCompletion' to generate a response. The response is returned as JSON with status 200. The server is configured to listen on port 5000 or the environment variable 'PORT' if set.

```
JS index.js > app.post("/ask") callback > [0] response
1  import dotenv from 'dotenv';
2  dotenv.config();
3
4  import express from 'express';
5  import { Configuration, OpenAIApi } from 'openai';
6
7  const app = express();
8  app.use(express.json());
9
10 const configuration = new Configuration({
11   apiKey: process.env.OPENAI_API_KEY,
12 });
13 const openai = new OpenAIApi(configuration);
14
15 const port = process.env.PORT || 5000;
16
17 app.post("/ask", async (req, res) => {
18   const prompt = req.body.prompt;
19
20   try {
21     if (prompt == null) {
22       throw new Error("Uh oh, no prompt was provided");
23     }
24
25     const response = await openai.createCompletion({
26       model: "text-davinci-003",
27       prompt,
28     });
29     const completion = response.data.choices[0].text;
30
31     return res.status(200).json({
32       success: true,
33       message: completion,
34     });
35   } catch (error) {
36     console.log(error.message);
37   }
38 });
39
40 app.listen(port, () => console.log(`Server is running on port ${port}!!`));
```

Step 1 – OpenAI Integration In Action



The screenshot displays the Postman API client interface. The top navigation bar includes 'Home', 'Workspaces', 'API Network', and 'Explore'. The left sidebar shows 'My Workspace' with a list of collections, including 'WILP AI Assignment' which is currently selected. The main panel shows a 'POST' request to 'http://localhost:5000/ask'. The request body is a JSON object:

```
{  "prompt": "Who is LikhIt Gatagat",  "max_tokens": 200}
```

. The response is displayed in the 'Body' tab, showing a JSON object:

```
{  "success": true,  "message": "?\n\nLikhIt Gatagat is a self-taught photographer"}
```

. The status bar at the bottom indicates 'Status: 200 OK', 'Time: 772 ms', and 'Size: 313 B'.



Step 2 – Front-end Setup with Vite + React

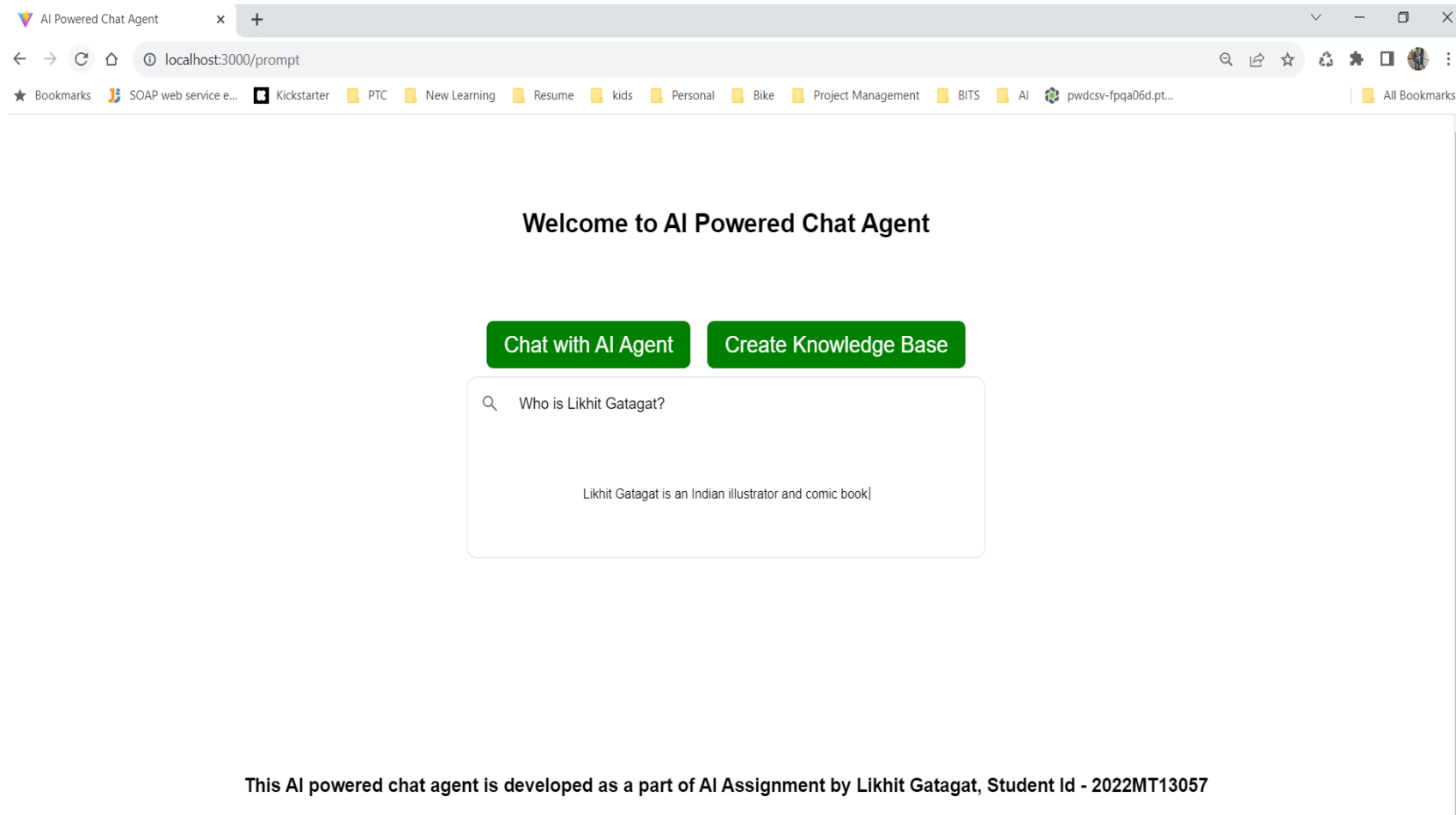
- **Objective:** Develop a front-end interface to interact with OpenAI for AI-driven responses.
- **Technologies Used:**
 - ❑ Front-end Framework: Vite + React
 - ❑ Other Technologies: HTML, CSS
- **Components and Navigation Logic:**
 - ❑ Home Page: Serves as the launch point of the application.
 - ❑ Navigation Links: "Chat with AI Agent" and "Create Knowledge Base" are two main links.
- **Functionality:**
 - ❑ Chat with AI Agent: Opens a prompt box for user queries, and fetches responses from OpenAI on enter.
 - ❑ Create Knowledge Base: To be discussed in the context of Embeddings.
- **Implementation:**
 - ❑ Vite + React: Utilized Vite as a build tool and React for UI components.
 - ❑ HomePage: Manages navigation and links to different functionalities.
 - ❑ Prompt: Handles the interaction with OpenAI for user queries.
- **Impact:** Created an intuitive front-end allowing users to engage with OpenAI for query-based responses.

Step 2 – Code Structure Overview



```
7  const Home = () => {
8    const [isLoggedIn, setIsLoggedIn] = useState(false);
9    // Function to handle login
10   const handleLogin = () => {
11     setIsLoggedIn(true);
12   };
13
14   // Function to handle logout
15   const handleLogout = () => {
16     setIsLoggedIn(false);
17   };
18
19   return (
20     <Router>
21       <div style={{ textAlign: 'center', marginTop: '100px' }}>
22         {isLoggedIn ? null : <h1>Welcome to AI Powered Chat Agent</h1>}
23       <div style={{ textAlign: 'center', marginTop: '100px' }}>
24         {/* Navigation Links */}
25         <nav style={{ textAlign: 'center', marginBottom: '20px' }}>
26           {
27             isLoggedIn ? null : (<Link to="/prompt" style={linkStyle}>Chat with AI Agent</Link>)
28           }
29
30           {
31             isLoggedIn ? (
32               null
33             ) : (<Link to="/login" style={linkStyle}>Create Knowledge Base</Link>)
34           }
35         </nav>
36
37         {/* Routes */}
38         <Routes>
39           <Route path="/prompt" element={<Prompt/>} />
40           {isLoggedIn ? (
41             <Route path="/knowledge-base" element={<KnowledgeBase handleLogout={handleLogout} />} />
42           ) : null
43           }
44           <Route path="/login" element={<LoginPage handleLogin={handleLogin}/>} />
45         </Routes>
46       </div>
47     </Router>
48   );
49 }
```

Step 2 – UI Screens





Integrated AI Agent with PDF Upload and Context Setting

- **Objective:** Integration of PDF upload, context setting, and AI agent for enriched chatbot responses.
- **Process Overview:**
 1. **Customizing Chatbot:**
 - ❑ Used PDF content to enhance the chatbot's knowledge base.
 - ❑ Utilized OpenAI's text embeddings for measuring relatedness between text strings.
 2. **Embedding Model API Usage:**
 - ❑ Generated embeddings for PDFs using OpenAI's embedding model API.
 - ❑ Stored embeddings in AirTable Base for domain-specific responses.
 3. **Cosine Similarity for Relevance:**
 - ❑ Computed cosine similarity between prompt embeddings and stored database embeddings.
 - ❑ Identified the most similar text using the highest cosine score.
 4. **Super Prompt Construction:**
 - ❑ Constructed a "super" prompt by embedding the most relevant text into a query.
 - ❑ Sent the refined prompt to OpenAI for a more contextually relevant answer.



Continued...

➤ Features Integrated:

1. PDF Upload Functionality:

- ☐ Users can browse and upload multiple PDFs within the application.
- ☐ Uploaded PDFs are utilized to enrich the chatbot's knowledge base.

2. Embedding Model API Usage:

- ☐ Users have the ability to set the context for the conversation.
- ☐ Contextual information from uploaded PDFs guides the AI agent's responses.

3. AI Agent Integration:

- ☐ The AI agent utilizes the supplied PDFs as a knowledge base for answering queries.
- ☐ Responses are generated based on the collective information from uploaded PDFs.

4. GUI Description:

- ☐ **User Interface:** A user-friendly interface allowing seamless PDF uploads and context setting.
 - ☐ **Interaction Flow:** Users navigate, upload PDFs, set context, and engage with the AI agent for responses.
-



Example Scenario

- **User Query:** "Who is Likhith Gatagat"

 - **Process Flow:**
 - ❑ Received embeddings data for the specific prompt.
 - ❑ Computed cosine similarity with database embeddings.
 - ❑ Constructed a refined prompt for OpenAI based on the most relevant text.
 - ❑ Sent the refined prompt and obtained a more contextually relevant response.

 - **Result:**
 - ❑ **Enhanced Relevance:** Obtained more relevant responses by incorporating contextually relevant prompts.
 - ❑ **Expanded Knowledge Base:** Enabled users to contribute by uploading PDFs, enriching the chatbot's knowledge.
-

Step 3 – UI Screens – User Login

A screenshot of a web browser window showing the login page for 'AI Powered Chat Agent'. The browser's address bar shows 'localhost:3000/login'. The page has a white background with a centered heading 'Welcome to AI Powered Chat Agent'. Below the heading are two green buttons: 'Chat with AI Agent' and 'Create Knowledge Base'. Further down is a 'Login' section with two input fields: one for the username 'admin' and another for a password represented by dots. Below these fields are two buttons: 'Login' and 'Cancel'. At the bottom of the page, there is a footer text: 'This AI powered chat agent is developed as a part of AI Assignment by Likhith Gatagat, Student Id - 2022MT13057'. The browser's bookmark bar is visible at the top, showing various links like 'SOAP web service e...', 'Kickstarter', 'PTC', 'New Learning', 'Resume', 'kids', 'Personal', 'Bike', 'Project Management', 'BITS', 'AI', and 'pwdcsv-fpqa06d.pt...'.

AI Powered Chat Agent

localhost:3000/login

Bookmarks SOAP web service e... Kickstarter PTC New Learning Resume kids Personal Bike Project Management BITS AI pwdcsv-fpqa06d.pt... All Bookmarks

Welcome to AI Powered Chat Agent

Chat with AI Agent Create Knowledge Base

Login

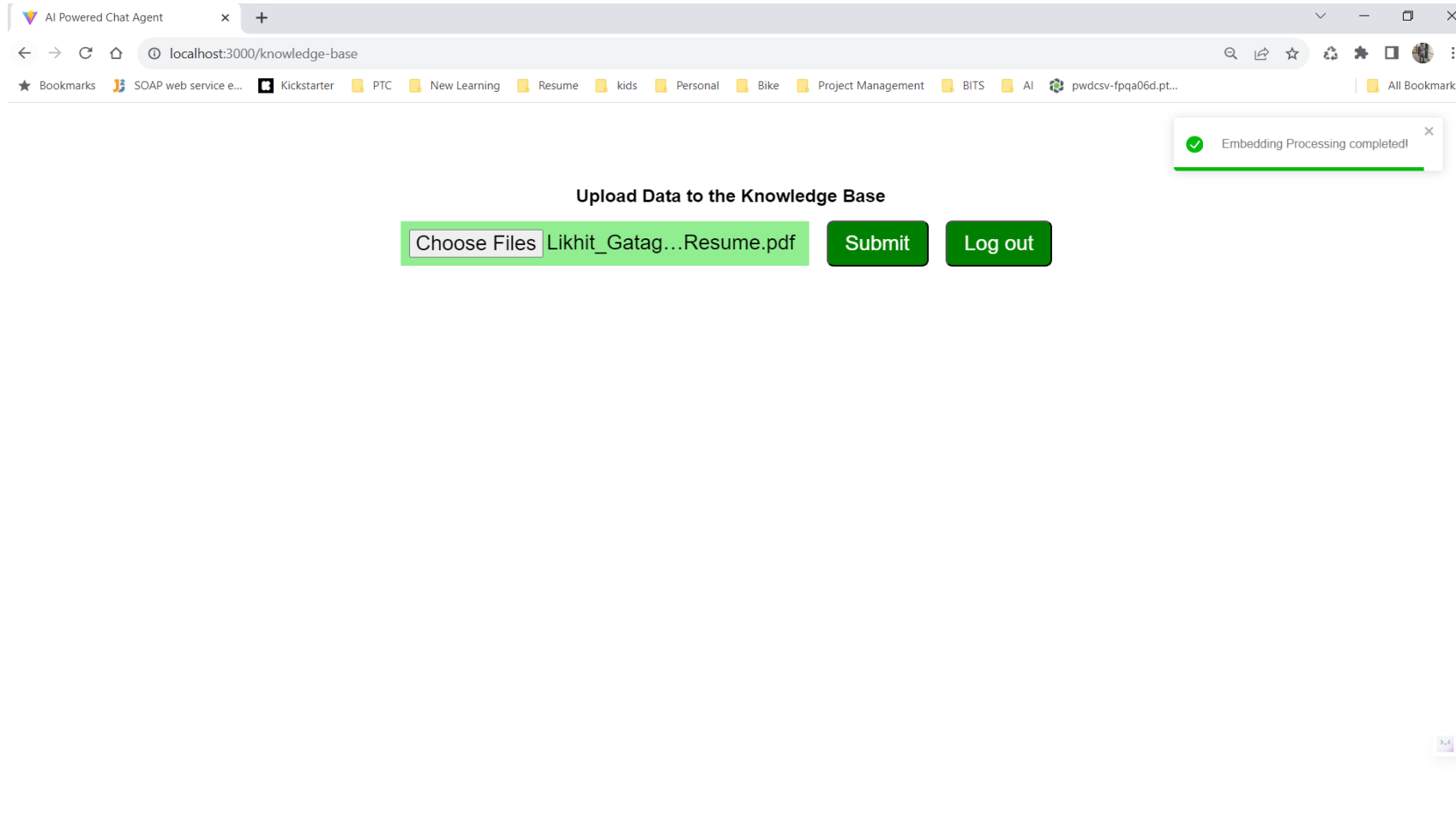
admin

Login Cancel

This AI powered chat agent is developed as a part of AI Assignment by Likhith Gatagat, Student Id - 2022MT13057



UI Screens – PDF Upload, Update Knowledge Base





UI Screens – Embeddings Database (Excel)

2022MT13057_AI_Assignment

airtable.com/appfnTH6WHqdQVTnq/tblK3nDO5tr8qRSDH/viwjJ6xJ7BbnUzYd2?blocks=hide

2022MT13057_AI_Assignment

Data Automations Interfaces

All changes saved Help Contact sales Share

Likhit Gatagat Embeddings

Views Grid view Hide fields Filter Group Sort Color Share and sync

Find a view

Grid view

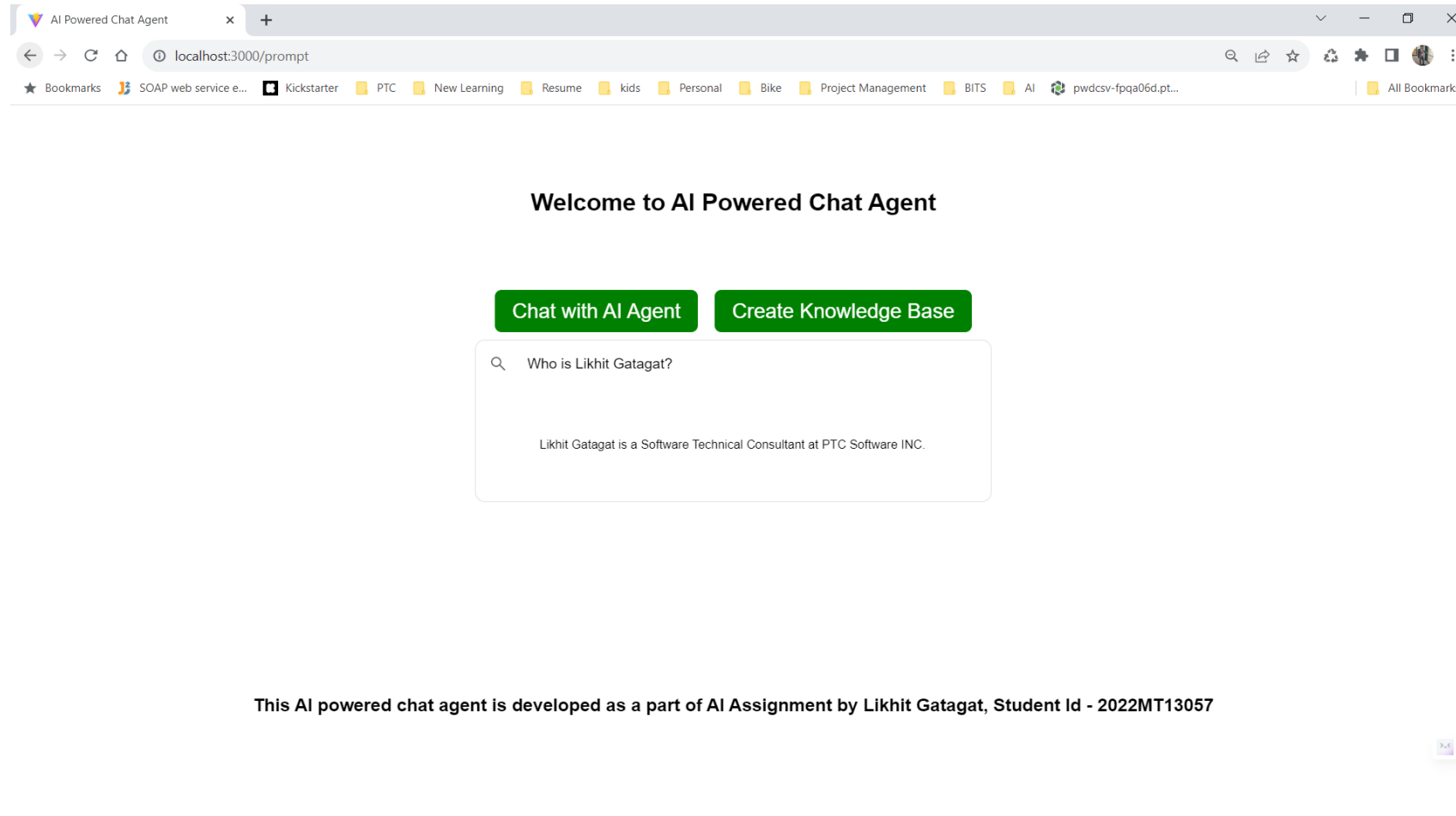
	Text	Embedding
1	Likhit Gatagat is a Software Technical Consultant, he works at PTC Software INC.	[-0.0077783535, 0.0035046826, -0.007601418, -0.01853739, 0.020088976, 0.022021655, -0.027479433, 0.01144...
2	Likhit is also pursuing M. Tech. from BITS Pilani under WILP Program	[-0.0141638415, -0.01769803, -0.003761001, -0.013053482, -0.0052674324, 0.04259985, -0.023236282, 0.0093...
3	Likhit loves travelling and is very passionate about programming	[0.009683508, 0.00862847, 0.002620797, -0.023587171, -0.0021151174, 0.0355622, -0.037148114, -0.0025720...
4	Likhit is a family man, he has two gorgeous and extremely talented children, Anaya and ...	[-0.000729351, 0.016634846, -0.0020893128, -0.01662157, -0.0017756668, 0.01891832, -0.0133423945, -0.004...
5	Anuja Gatagat Contact information 9767847420 anujagatagat@gmail.com B - 201, Shiv ...	[0.004636004, 0.019736985, -0.016184593, -0.008092296, -0.019392349, 0.034967203, -0.023647266, 0.002410...
6	RESUME Name : Tanoo Sachin Singh Mob.No : 7020260322 Date of birth : 17 th Februar...	[-0.00038198326, 0.00927252, 0.0019492622, -0.014308798, -0.006147799, 0.030637352, -0.031948883, -0.000...
7	Page 1 of 3 AJINKYA PARCHURE ajinkya.parchure@gmail.com +91 9766783823 https://...	[-0.0006565908, 0.008901242, 0.0063707056, -0.009291647, -0.02738516, 0.016893905, -0.024503259, 0.00887...
8	M GAUTAM RAJ PUNE mgautamraj9893@gmail.com 9893747097 PUNE, INDIA Hard-wo...	[-0.013359385, -0.021611173, 0.0018158053, -0.02665012, -0.027419005, 0.03674174, -0.0144440625, -0.0084...
9	. Likhit Gatagat Software Craftsman Pune, Maharashtra, 411048 09021188130 likhitgatag...	[-0.016898759, 0.0092604635, -0.0036710366, -0.029255446, -0.0057375375, 0.028465522, -0.015121426, -0.0...
+		

9 records

Getting started



UI Screens – Relevant response as per knowledge base



Demo



[Link to the Demo](#)



[Link to the Source Code](#)

A futuristic robot with a white and blue body and a glowing, complex head structure is shown in profile, reaching out with its right hand. The background is a vibrant blue and purple gradient with a city skyline at the bottom. A large, glowing digital interface is overlaid on the left side, featuring a central bar chart and various icons representing different sectors like weather, industry, healthcare, and transportation.

**THANK YOU FOR
YOUR ATTENTION**