

1. Ai Use Case: concept of the sports Results Chatbot

Title: GoalBuddy – Your personalized football assistant

Subtitle: Practical work for artificial intelligence use Case course

GoalBuddy is an app we developed to make life easier for football fans by allowing them to easily track competition scores and standings. This project was carried out as part of our studies in artificial intelligence. The app is built with modern technologies like React, Node.js and Wit.ai, which helps create a simple and enjoyable user experience. You can discover and test goalbuddy at the following link:<https://goalbuddy-ai.web.app/>

here are the key features of our application:

- **Next matches:** View upcoming match information for each team.
- **Match details:** Get accurate information about matches between two specific teams.
- **League standings:** View the current rankings of the main football competitions.
- **Live score tracking:** Stay informed about the latest scores of your favorite teams.
- **Todays match:** view the current match play during the day



2. GoalBuddy Features

GoalBuddy is designed to provide users with an intuitive experience for tracking soccer scores, getting match information, and viewing league standings. Here is an overview of the main features:

1. Latest match score:

- Description: Get the latest match scores for a specific competition.
- Example command: "What is the latest Bundesliga score?"

2. Next match:

- Description: Get information about a team's next match.
- Example of command: "When is the next France match?"

3. Get the scorers of a specific competition:

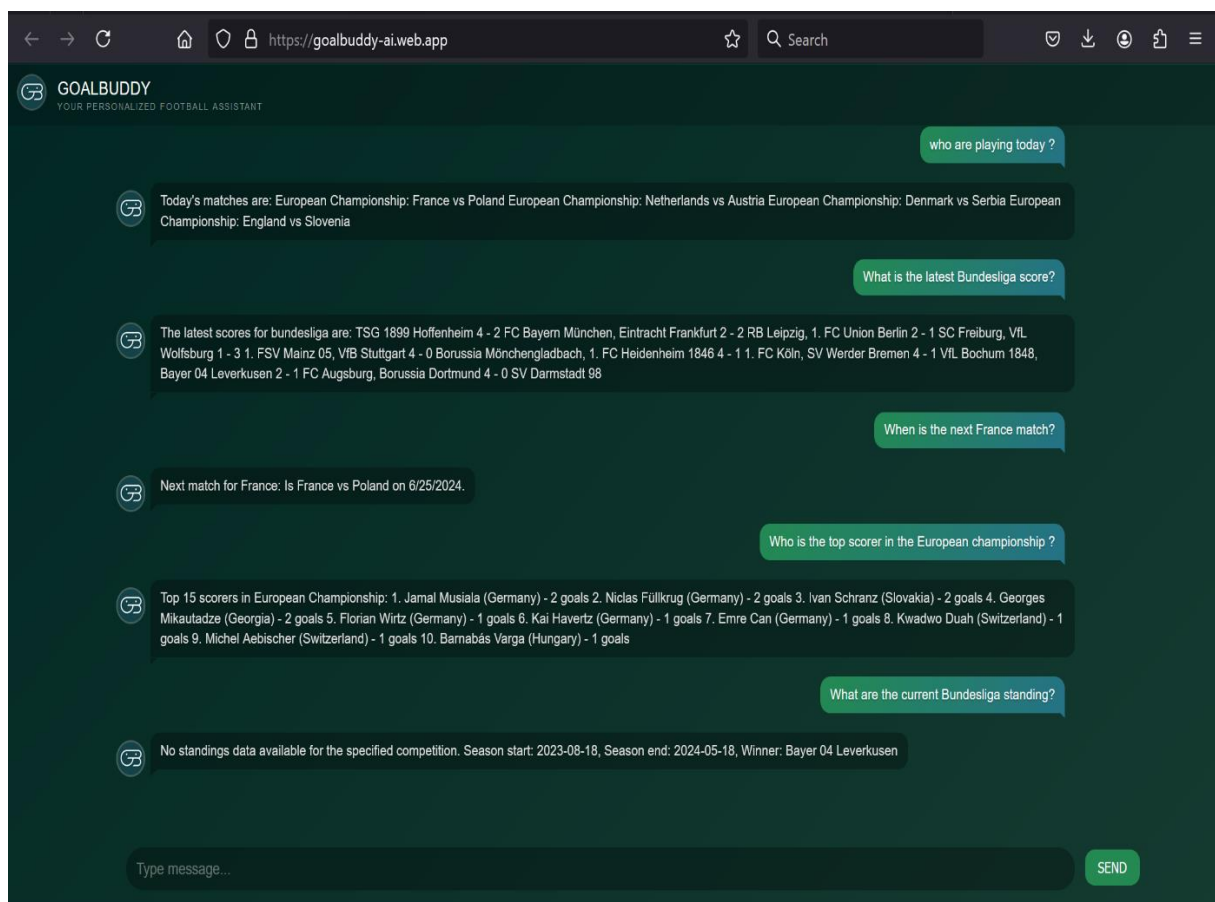
- Description: Obtain information on the top 15 scorers in a specific competition.
- Example command: "Who is the top scorer in the European Cup?"

4. League standings:

- Description: Get current rankings for a specific competition.
- Example command: "What is the current Premier League standings?"

5. Matches of the day:

- Description: Get current rankings for a specific competition.
- Example command: "Who are playing today?"



We would like to point out that the above functionalities are those available with the free version of the football-data.org API. Indeed the API does not allow us, for example, to have the statistics of a match for example: scorers, assists ... for more information consult this link:<https://www.football-data.org/pricing>

3. Technologies Used

List of frameworks and tools:

a) React <https://reactjs.org/> :

- **Role:** React allowed us to build the app user interface. It allows us to create reusable components and efficiently manage application state.
- **Description:** Thanks to React, the application offers a responsive and dynamic interface where users can easily view the scores, standings and upcoming matches of their favorite teams.

b) Node.js <https://nodejs.org/en> :

- **Role:** Node.js is the server technology used for our backend. It allows JavaScript to be executed on the server side, making it easier to manage requests and responses between the frontend and the databases.
- **Description:** Node.js handles user requests, interacts with the soccer API to retrieve the necessary data, and returns that data to React for display.

c) Wit.ai <https://wit.ai/> :

- **Role:** We used Wit.ai for intent recognition and natural language understanding (NLP). It allows the application to understand and interpret user queries in natural language.
- **Description:** Thanks to Wit.ai, our web app can understand questions like "What is France's next match?" and provide appropriate responses by querying the Football API.

d) Football-data.org <https://www.football-data.org/> :

- **Role:** Football-data.org provides necessary football data, such as match scores, league standings and match schedules.
- **Description:** The football-data.org API is essential for obtaining the up-to-date information that our app displays to users. It allows you to retrieve precise and reliable data on football competitions.

e) Firebase <https://firebase.google.com/> :

- **Role:** We used Firebase to host the frontend application. It offers a reliable and scalable hosting solution for web applications.
- **Description:** With Firebase, we deployed and hosted our application with ease, ensuring optimal availability and performance for all users.

f) Heroku <https://dashboard.heroku.com/apps> :

- **Role:** We used Heroku to host the backend server of our application. It makes it easier to deploy and manage Node.js applications.
- **Description:** Heroku allows us to deploy our backend server quickly and reliably. It also manages application updates and scalability.

g) Visual studio code <https://code.visualstudio.com/> :

- **Role:** Visual studio code is a simple, flexible code editor with an easy-to-use interface, it allowed us to implement and compile the code of our application

- **Description:** it was with visual studio code that we were able, from our terminal, to launch the “create-react-app goalbuddy” command for the base of our application.

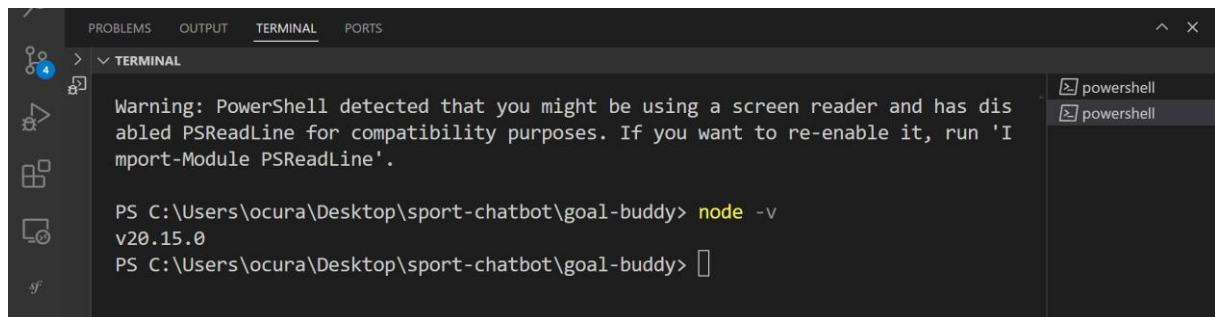
4. Environment Configuration

Preparing the development environment for building our web app

Description: To start working on a web app, it is essential to configure our development environment correctly. Here are the detailed steps to prepare our environment and start the project on our local machine.

a) Installing Node.js:

In our case it is already done we will show you the version that we had installed on our PC, but we have created a “sport-chatbot” folder first in the desktop of our PC and in this folder we have a subfolder “goalbuddy” (which is the name of our app):



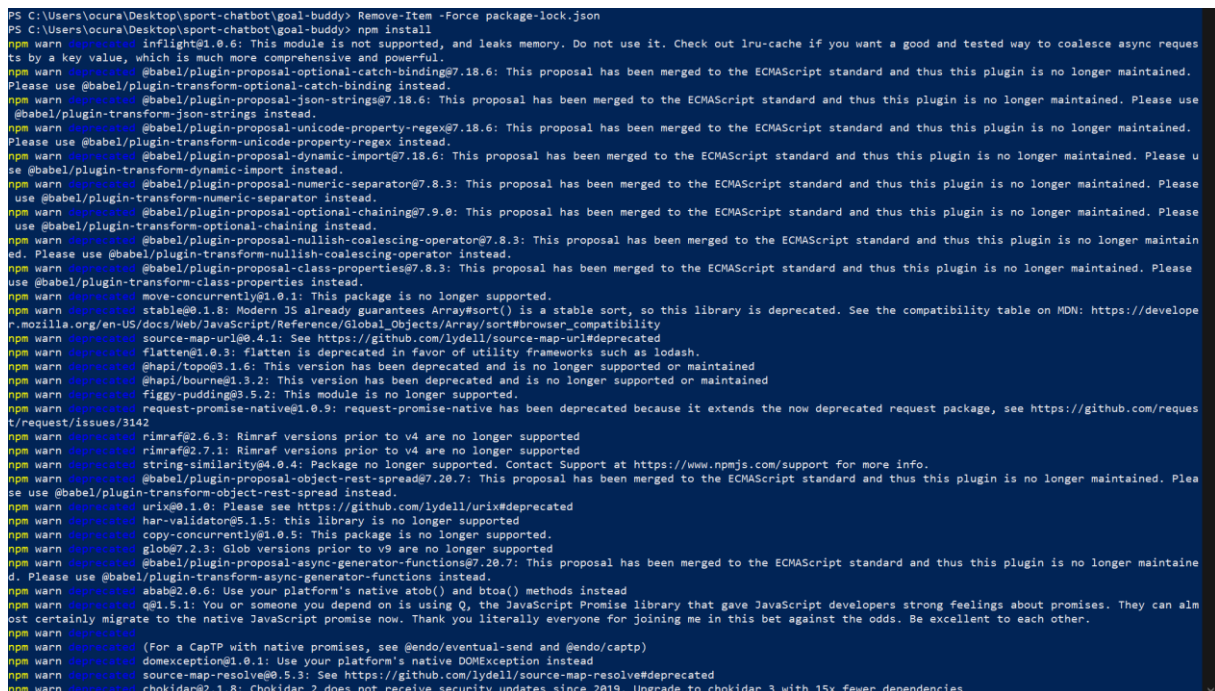
```

Warning: PowerShell detected that you might be using a screen reader and has disabled PSReadLine for compatibility purposes. If you want to re-enable it, run 'Import-Module PSReadLine'.

PS C:\Users\ocura\Desktop\sport-chatbot\goal-buddy> node -v
v20.15.0
PS C:\Users\ocura\Desktop\sport-chatbot\goal-buddy>
  
```

b) Installing dependencies :

we would like to point out that the dependencies were not installed all at once, however as we evolved we needed for example **axios** to allow us to manage **HTTP** requests, then we needed **body-parser** which is a module which allows us to interpret the **JSON** cors of an **HTTP** request. And all these dependencies are stored in a **package.json** file at the root of our project which allows us to install them in one click during deployment in a remote server:



```

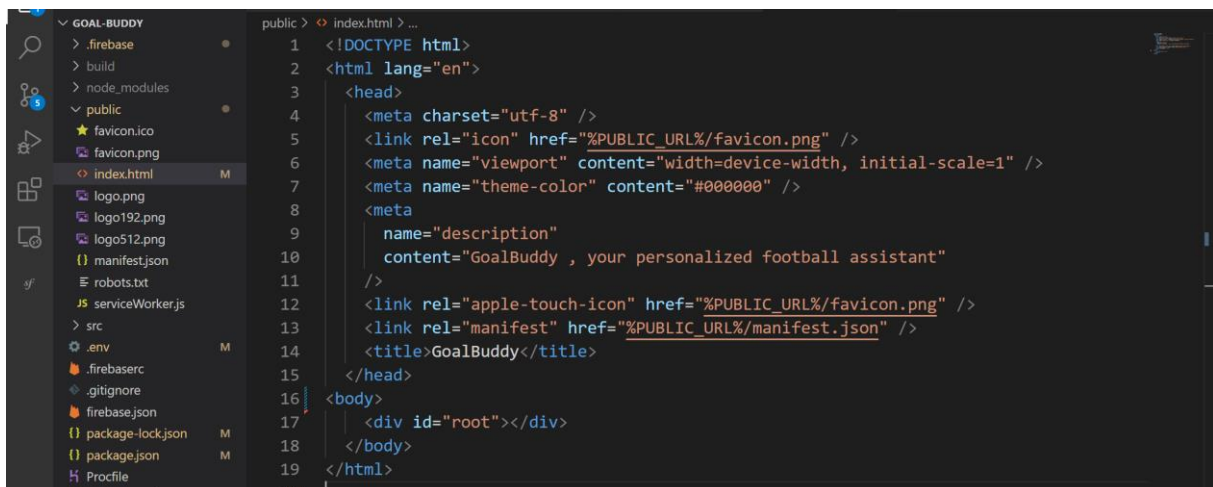
PS C:\Users\ocura\Desktop\sport-chatbot\goal-buddy> Remove-Item -Force package-lock.json
PS C:\Users\ocura\Desktop\sport-chatbot\goal-buddy> npm install
npm warn deprecated inflight@1.0.6: This module is not supported, and leaks memory. Do not use it. Check out lru-cache if you want a good and tested way to coalesce async requests by a key value, which is much more comprehensive and powerful.
npm warn deprecated @babel/plugin-proposal-optional-catch-binding@7.18.6: This proposal has been merged to the ECMAScript standard and thus this plugin is no longer maintained. Please use @babel/plugin-transform-optional-catch-binding instead.
npm warn deprecated @babel/plugin-proposal-json-strings@7.18.6: This proposal has been merged to the ECMAScript standard and thus this plugin is no longer maintained. Please use @babel/plugin-transform-json-strings instead.
npm warn deprecated @babel/plugin-proposal-unicode-property-regex@7.18.6: This proposal has been merged to the ECMAScript standard and thus this plugin is no longer maintained. Please use @babel/plugin-transform-unicode-property-regex instead.
npm warn deprecated @babel/plugin-proposal-dynamic-import@7.18.6: This proposal has been merged to the ECMAScript standard and thus this plugin is no longer maintained. Please use @babel/plugin-transform-dynamic-import instead.
npm warn deprecated @babel/plugin-proposal-numeric-separator@7.8.3: This proposal has been merged to the ECMAScript standard and thus this plugin is no longer maintained. Please use @babel/plugin-transform-numeric-separator instead.
npm warn deprecated @babel/plugin-proposal-optional-chaining@7.9.0: This proposal has been merged to the ECMAScript standard and thus this plugin is no longer maintained. Please use @babel/plugin-transform-optional-chaining instead.
npm warn deprecated @babel/plugin-proposal-nullish-coalescing-operator@7.8.3: This proposal has been merged to the ECMAScript standard and thus this plugin is no longer maintained. Please use @babel/plugin-transform-nullish-coalescing-operator instead.
npm warn deprecated @babel/plugin-proposal-class-properties@7.8.3: This proposal has been merged to the ECMAScript standard and thus this plugin is no longer maintained. Please use @babel/plugin-transform-class-properties instead.
npm warn deprecated move-concurrently@1.0.1: This package is no longer supported.
npm warn deprecated stable@0.1.8: Modern JS already guarantees Array#sort() is a stable sort, so this library is deprecated. See the compatibility table on MDN: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/sort#browser_compatibility
npm warn deprecated source-map-url@0.4.1: See https://github.com/lydell/source-map-url#deprecated
npm warn deprecated flatten@0.3.1: flatten is deprecated in favor of utility frameworks such as lodash.
npm warn deprecated @hapi/topo@3.1.6: This version has been deprecated and is no longer supported or maintained
npm warn deprecated @hapi/bourne@1.3.2: This version has been deprecated and is no longer supported or maintained
npm warn deprecated figgy-pudding@3.5.2: This module is no longer supported.
npm warn deprecated request-promise-native@1.0.9: request-promise-native has been deprecated because it extends the now deprecated request package, see https://github.com/request/request/issues/3142
npm warn deprecated rimraf@2.6.3: Rimraf versions prior to v4 are no longer supported
npm warn deprecated rimraf@2.7.1: Rimraf versions prior to v4 are no longer supported
npm warn deprecated string-similarity@4.0.4: Package no longer supported. Contact Support at https://www.npmjs.com/support for more info.
npm warn deprecated @babel/plugin-proposal-object-rest-spread@7.20.7: This proposal has been merged to the ECMAScript standard and thus this plugin is no longer maintained. Please use @babel/plugin-transform-object-rest-spread instead.
npm warn deprecated @babel/plugin-proposal-async-generator-functions@7.20.7: This proposal has been merged to the ECMAScript standard and thus this plugin is no longer maintained. Please use @babel/plugin-transform-async-generator-functions instead.
npm warn deprecated abab@2.0.6: Use your platform's native atob() and btoa() methods instead
npm warn deprecated q@1.5.1: You or someone you depend on is using Q, the JavaScript Promise library that gave JavaScript developers strong feelings about promises. They can almost certainly migrate to the native JavaScript promise now. Thank you literally everyone for joining me in this bet against the odds. Be excellent to each other.
npm warn deprecated (For a CapTP with native promises, see @endo/eventual-send and @endo/captp)
npm warn deprecated domexception@1.0.1: Use your platform's native DOMException instead
npm warn deprecated source-map-resolve@0.5.3: See https://github.com/lydell/source-map-resolve#deprecated
npm warn deprecated chokidar@2.1.8: Chokidar 2 does not receive security updates since 2019. Upgrade to chokidar 3 with 15x fewer dependencies
  
```

5. Description of the main directories and main files

Please note that we will not be able to illustrate the entire tree of our project... We will illustrate here the main directories and the main files of each directory:

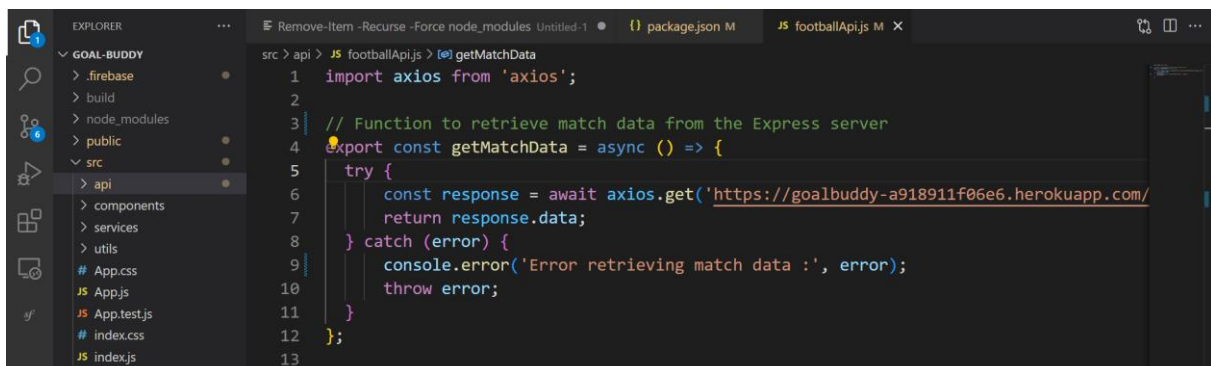
- a) **public/**: Contains static public files that will be served directly by the server.
- b) **src/**: Contains all the source code of the application.
- c) **components/**: Contained in the **src/** folder, contains the React components used in the application.
- d) **services/**: Contained in the **src/** folder, contains service files for API calls and data management.
- e) **utils/**: Contained in the **src/** folder, Contains utility files for various auxiliary functions.
- f) **App.js**: Content in the **src/** folder, main component of the application.
- g) **index.js**: Content in the **src/** folder, entry point of the React application.
- h) **server.js**: Located at the root of our project, this is the Express server file, handling API requests and serving static files for the application.

In the **public/** folder, the main file is **index.html**, which is the main **HTML** page of the application.



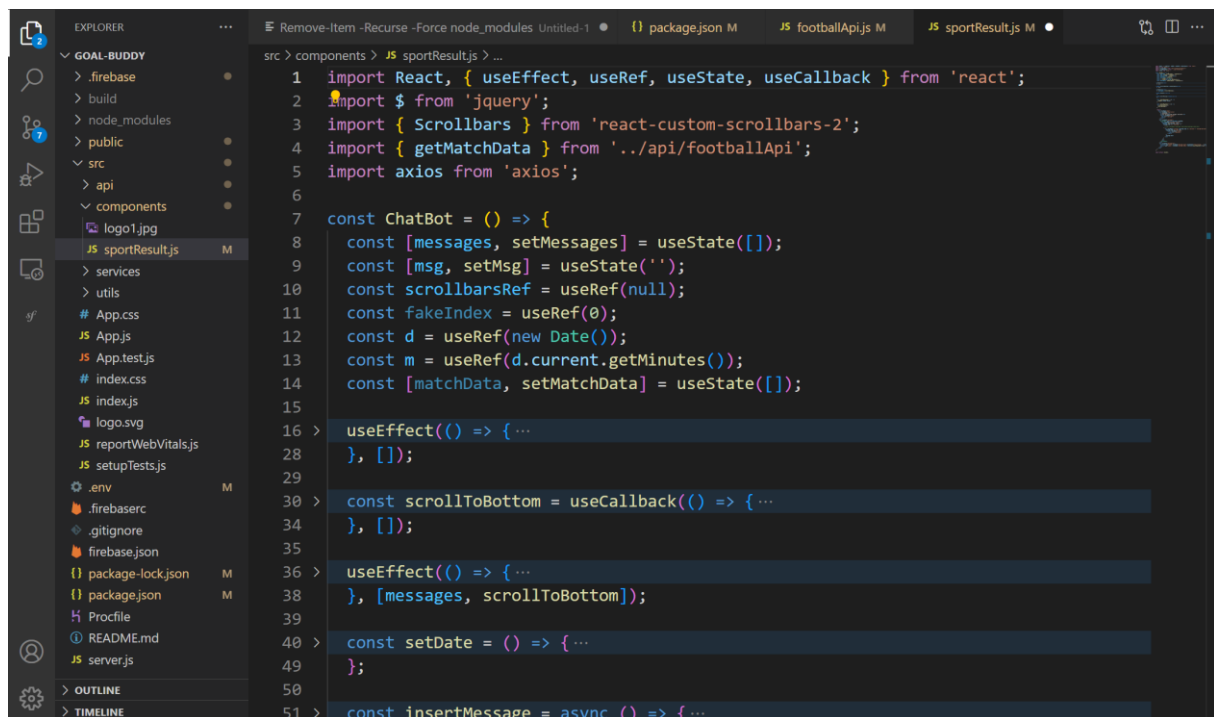
```
1 <!DOCTYPE html>
2 <html lang="en">
3   <head>
4     <meta charset="utf-8" />
5     <link rel="icon" href="%PUBLIC_URL%/favicon.png" />
6     <meta name="viewport" content="width=device-width, initial-scale=1" />
7     <meta name="theme-color" content="#000000" />
8     <meta
9       name="description"
10      content="GoalBuddy , your personalized football assistant"
11    />
12    <link rel="apple-touch-icon" href="%PUBLIC_URL%/favicon.png" />
13    <link rel="manifest" href="%PUBLIC_URL%/manifest.json" />
14    <title>GoalBuddy</title>
15  </head>
16  <body>
17    <div id="root"></div>
18  </body>
19 </html>
```

In the **/src** folder, we have the **api/** subfolder which contains the **footballApi.js** file, which contains code to retrieve match data from the Express server.



```
1 import axios from 'axios';
2
3 // Function to retrieve match data from the Express server
4 export const getMatchData = async () => {
5   try {
6     const response = await axios.get('https://goalbuddy-a918911f06e6.herokuapp.com/');
7     return response.data;
8   } catch (error) {
9     console.error('Error retrieving match data :', error);
10    throw error;
11  }
12 };
13
```

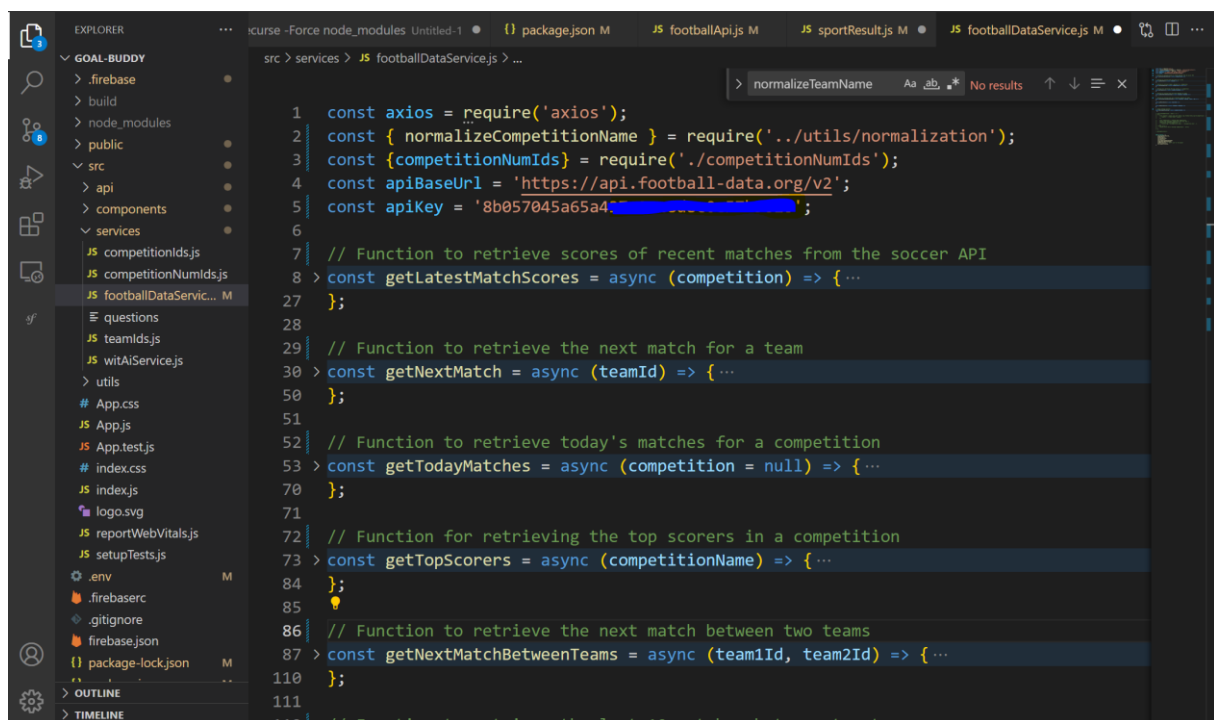
In the **component/** folder we have the **sportResult.js** file, which plays a central role in the application by providing an interactive user interface to interact with the chatbot. It integrates API calls to retrieve match data and manages the state and rendering of messages reactively to user interface.



The screenshot shows the VS Code editor with the file explorer on the left displaying the project structure. The file `src/components/sportResult.js` is selected. The main editor area shows the code for the `ChatBot` component, which uses `useState` and `useEffect` from `react`, `jQuery`, `react-custom-scrollbar-2`, and `axios`. The code includes state for messages, a scroll bar reference, a fake index, a date, and match data. It also features several `useEffect` hooks for initializing the chat bot and scrolling to the bottom.

```
1 import React, { useEffect, useRef, useState, useCallback } from 'react';
2 import $ from 'jquery';
3 import { Scrollbars } from 'react-custom-scrollbar-2';
4 import { getMatchData } from '../api/footballApi';
5 import axios from 'axios';
6
7 const ChatBot = () => {
8   const [messages, setMessages] = useState([]);
9   const [msg, setMsg] = useState('');
10  const scrollbarsRef = useRef(null);
11  const fakeIndex = useRef(0);
12  const d = useRef(new Date());
13  const m = useRef(d.current.getMinutes());
14  const [matchData, setMatchData] = useState([]);
15
16  > useEffect(() => { ...
28  }, []);
29
30  > const scrollToBottom = useCallback(() => { ...
34  }, []);
35
36  > useEffect(() => { ...
38  }, [messages, scrollToBottom]);
39
40  > const setDate = () => { ...
49  };
50
51  > const insertMessage = async () => { ...
```

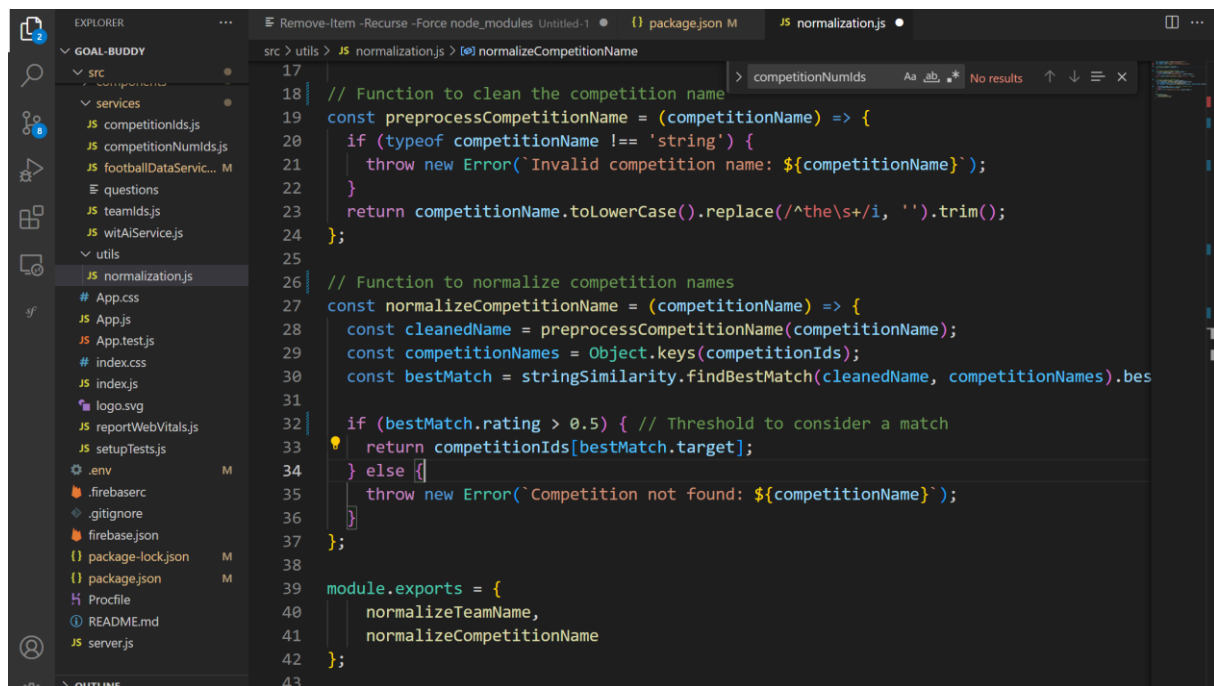
In the **service/** folder we have the file **footballDataService.js**, which centralizes the interaction logic with the **football-data.org** API. It provides functions to retrieve various football information, such as match scores, upcoming matches, top scorers, and league standings.



The screenshot shows the VS Code editor with the file explorer on the left displaying the project structure. The file `src/services/footballDataService.js` is selected. The main editor area shows the code for the `footballDataService` module, which uses `axios` and `normalizeTeamName` from `../utils/normalization`. The code includes several functions for retrieving match scores, upcoming matches, today's matches, top scorers, and the next match between two teams.

```
1 const axios = require('axios');
2 const { normalizeTeamName } = require('../utils/normalization');
3 const { competitionNumIds } = require('../competitionNumIds');
4 const apiBaseUrl = 'https://api.football-data.org/v2';
5 const apiKey = '8b057045a65a4...';
6
7 // Function to retrieve scores of recent matches from the soccer API
8 > const getLatestMatchScores = async (competition) => { ...
27  };
28
29 // Function to retrieve the next match for a team
30 > const getNextMatch = async (teamId) => { ...
50  };
51
52 // Function to retrieve today's matches for a competition
53 > const getTodayMatches = async (competition = null) => { ...
70  };
71
72 // Function for retrieving the top scorers in a competition
73 > const getTopScorers = async (competitionName) => { ...
84  };
85
86 // Function to retrieve the next match between two teams
87 > const getNextMatchBetweenTeams = async (team1Id, team2Id) => { ...
110  };
111
112 // Function to retrieve the last 10 matches between two teams
```

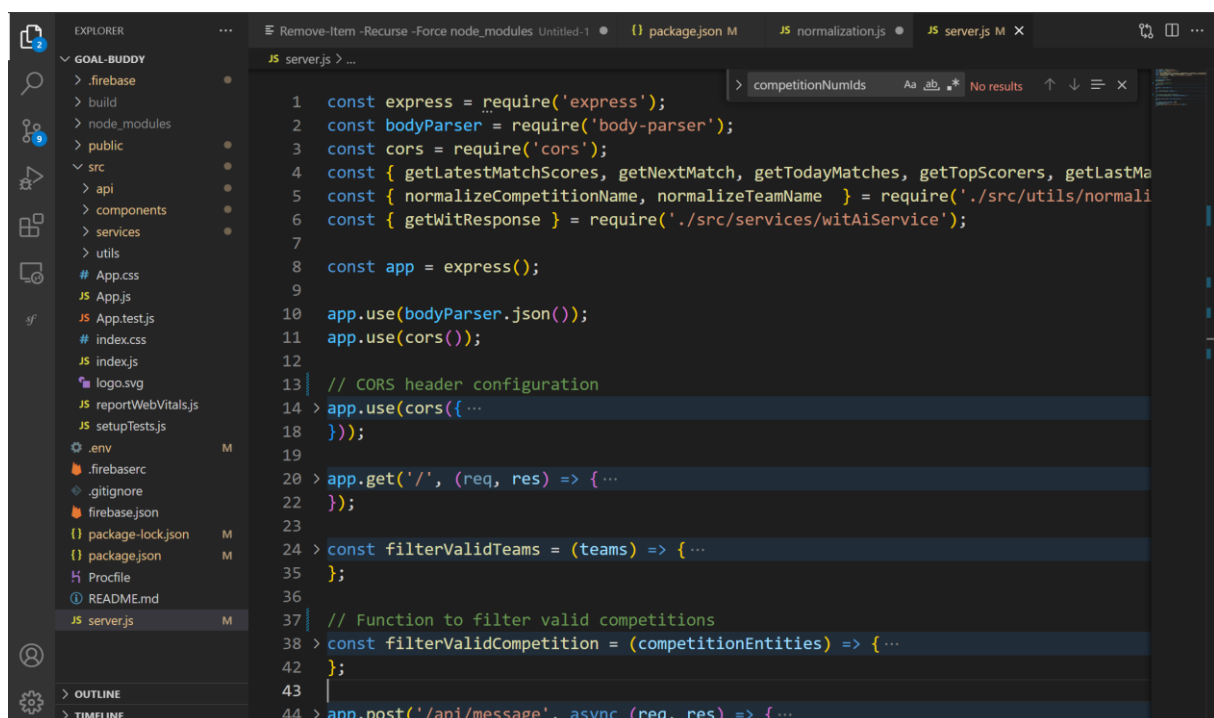
In the **utils/** folder we have the **normalization.js** file, which ensures that team and competition names are standardized before being used to interact with the **football-data.org** API. This helps minimize errors due to variations in names and ensures API requests are accurate.



```
17
18 // Function to clean the competition name
19 const preprocessCompetitionName = (competitionName) => {
20   if (typeof competitionName !== 'string') {
21     throw new Error(`Invalid competition name: ${competitionName}`);
22   }
23   return competitionName.toLowerCase().replace(/the\s+/i, '').trim();
24 };
25
26 // Function to normalize competition names
27 const normalizeCompetitionName = (competitionName) => {
28   const cleanedName = preprocessCompetitionName(competitionName);
29   const competitionNames = Object.keys(competitionIds);
30   const bestMatch = stringSimilarity.findBestMatch(cleanedName, competitionNames).bestMatch;
31
32   if (bestMatch.rating > 0.5) { // Threshold to consider a match
33     return competitionIds[bestMatch.target];
34   } else {
35     throw new Error(`Competition not found: ${competitionName}`);
36   }
37 };
38
39 module.exports = {
40   normalizeTeamName,
41   normalizeCompetitionName
42 };
43
```

Regarding the **App.js** and **index.js** files, there were no major changes in the basic configuration generated by the compilation of the “**create-react-app**” code, which allowed us to initialize our react project . However since they play an essential role in the execution of our program we have mentioned them.

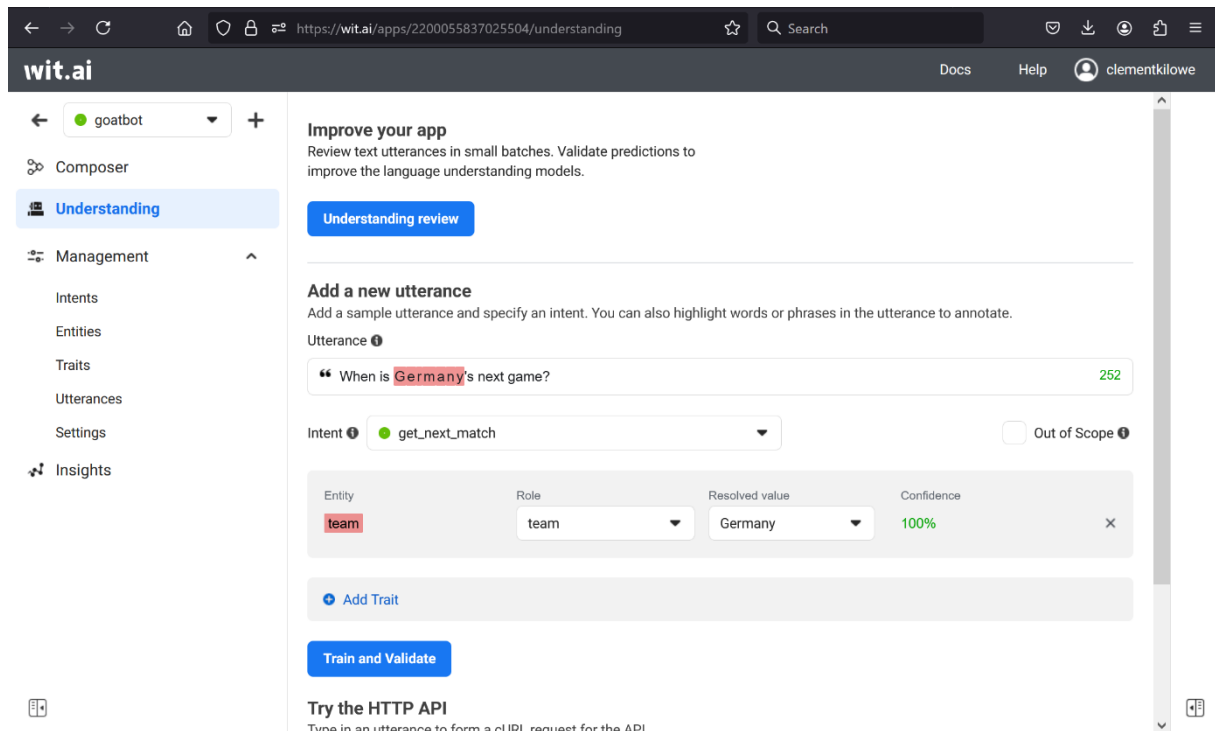
The file **server.js** is essential for bridging the web app frontend and backend services, particularly for processing user requests, interacting with the Wit.ai API to understand user intent, and retrieving relevant data from the **football-data.org** API.



```
1 const express = require('express');
2 const bodyParser = require('body-parser');
3 const cors = require('cors');
4 const { getLatestMatchScores, getNextMatch, getTodayMatches, getTopScorers, getLastMatch } = require('./src/services/footballDataService');
5 const { normalizeCompetitionName, normalizeTeamName } = require('./src/utls/normalizeCompetitionName');
6 const { getWitResponse } = require('./src/services/witAiService');
7
8 const app = express();
9
10 app.use(bodyParser.json());
11 app.use(cors());
12
13 // CORS header configuration
14 app.use(cors({
15   origin: '*',
16   methods: 'GET,HEAD,PUT,PATCH,POST,DELETE',
17   credentials: true,
18 }));
19
20 app.get('/', (req, res) => {
21   res.json({ message: 'Welcome to the API' });
22 });
23
24 const filterValidTeams = (teams) => {
25   return teams.filter(team => team.name !== ' ');
26 };
27
28 // Function to filter valid competitions
29 const filterValidCompetition = (competitionEntities) => {
30   return competitionEntities.filter(entity => entity.name !== ' ');
31 };
32
33 app.post('/api/message', async (req, res) => {
34   const { message } = req.body;
35   if (!message) {
36     res.status(400).json({ message: 'Message is required' });
37     return;
38   }
39   const witResponse = await getWitResponse(message);
40   const intent = witResponse.intent.name;
41   const entities = witResponse.entities;
42
43   switch (intent) {
44     case 'get_latest_matches':
45       const latestMatches = await getLatestMatchScores();
46       res.json(latestMatches);
47       break;
48     case 'get_next_match':
49       const nextMatch = await getNextMatch();
50       res.json(nextMatch);
51       break;
52     case 'get_today_matches':
53       const todayMatches = await getTodayMatches();
54       res.json(todayMatches);
55       break;
56     case 'get_top_scorers':
57       const topScorers = await getTopScorers();
58       res.json(topScorers);
59       break;
60     case 'get_last_match':
61       const lastMatch = await getLastMatch();
62       res.json(lastMatch);
63       break;
64     case 'normalize_team_name':
65       const teamName = entities.team_name.value;
66       const normalizedTeamName = normalizeTeamName(teamName);
67       res.json({ normalizedTeamName });
68       break;
69     case 'normalize_competition_name':
70       const competitionName = entities.competition_name.value;
71       const normalizedCompetitionName = normalizeCompetitionName(competitionName);
72       res.json({ normalizedCompetitionName });
73       break;
74     default:
75       res.status(400).json({ message: 'Invalid intent' });
76       break;
77   }
78 });
79
80 const port = process.env.PORT || 3000;
81 app.listen(port, () => {
82   console.log(`Server is running on port ${port}`);
83 });
```

6. Wit.ai API

Wit.ai has been a valuable asset in the development of our app, adding a layer of artificial intelligence that significantly improves the user experience. Thus transforming our football score tracking application into a real intelligent assistant. Using **Wit.ai**, users can ask questions in natural language, and our app understands these questions and answers them in a meaningful way.

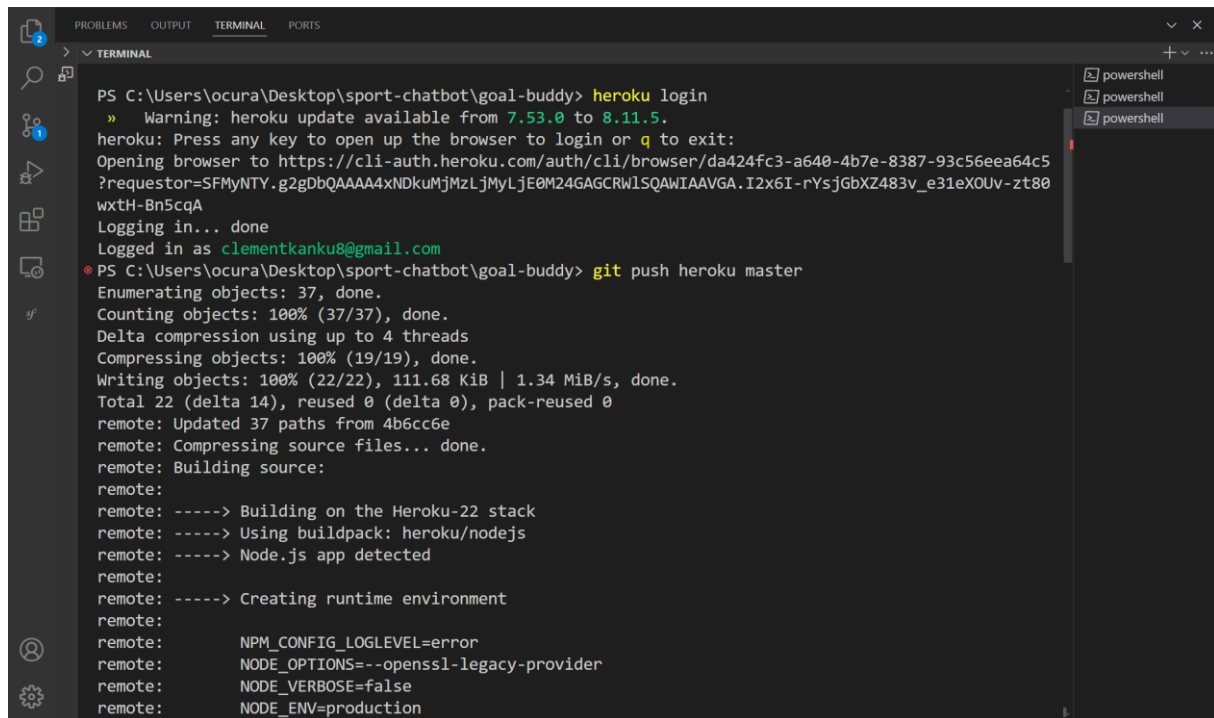


Wit.ai works as follows; it analyzes user messages and extracts **intent** (intention) and **entities** (elements that we had preconfigured on the Wit.ai interface). For example, when the user asks "When is **Germany's** next match?", Wit.ai detects the **get_next_match** intent and the **team** entity with the value "Germany". Our Node.js server then processes this **intent** and returns the appropriate information.

7. Deployment on Heroku and Google firestore

To make our application accessible to a wide audience, it is essential to host it on reliable and efficient platforms. We chose to use **Heroku** to host our backend server and **Firebase** to deploy our frontend application.

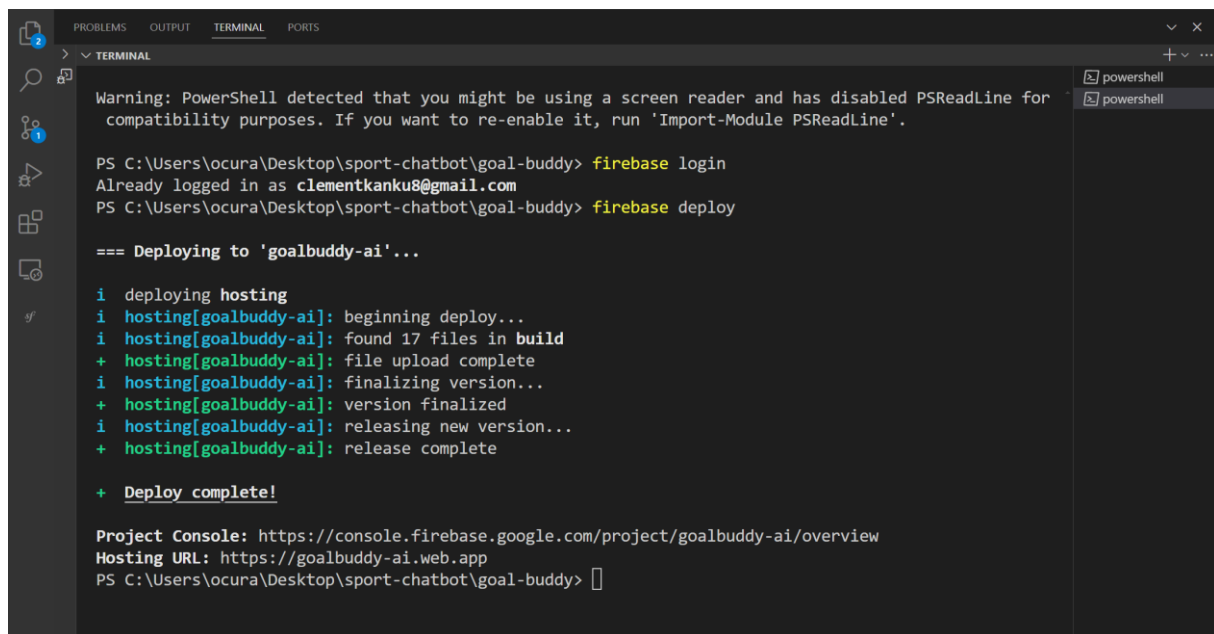
Heroku makes it quick and easy to deploy applications with a simple Git command. This greatly simplifies the process of bringing our **Nodejs** server online. To do this we went to the root of our project via command prompt, as illustrated:



```
PS C:\Users\ocura\Desktop\sport-chatbot\goal-buddy> heroku login
» Warning: heroku update available from 7.53.0 to 8.11.5.
heroku: Press any key to open up the browser to login or q to exit:
Opening browser to https://cli-auth.heroku.com/auth/cli/browser/da424fc3-a640-4b7e-8387-93c56eea64c5
?requestor=SFMyNTY.g2gDbQAAAA4xNDkuMjMzLjMyLjE0M24GAGRWL1SQAWIAAVGA.I2x6I-rYsjGbXZ483v_e31eXOUv-zt80
wxtH-Bn5cQA
Logging in... done
Logged in as clementkanku8@gmail.com
PS C:\Users\ocura\Desktop\sport-chatbot\goal-buddy> git push heroku master
Enumerating objects: 37, done.
Counting objects: 100% (37/37), done.
Delta compression using up to 4 threads
Compressing objects: 100% (19/19), done.
Writing objects: 100% (22/22), 111.68 KiB | 1.34 MiB/s, done.
Total 22 (delta 14), reused 0 (delta 0), pack-reused 0
remote: Updated 37 paths from 4b6cc6e
remote: Compressing source files... done.
remote: Building source:
remote:
remote: -----> Building on the Heroku-22 stack
remote: -----> Using buildpack: heroku/nodejs
remote: -----> Node.js app detected
remote:
remote: -----> Creating runtime environment
remote:
remote:     NPM_CONFIG_LOGLEVEL=error
remote:     NODE_OPTIONS=--openssl-legacy-provider
remote:     NODE_VERBOSE=false
remote:     NODE_ENV=production
```

heroku server link:<https://goalbuddy-a918911f06e6.herokuapp.com/>

Firebase offers fast and secure hosting for static web applications, with automatic **SSL** configuration. To host the front-end part of our application on **firestore**, we once again went to the root of our project as the illustration shows:



```
Warning: PowerShell detected that you might be using a screen reader and has disabled PSReadLine for compatibility purposes. If you want to re-enable it, run 'Import-Module PSReadLine'.

PS C:\Users\ocura\Desktop\sport-chatbot\goal-buddy> firebase login
Already logged in as clementkanku8@gmail.com
PS C:\Users\ocura\Desktop\sport-chatbot\goal-buddy> firebase deploy

=== Deploying to 'goalbuddy-ai'...

i  deploying hosting
i  hosting[goalbuddy-ai]: beginning deploy...
i  hosting[goalbuddy-ai]: found 17 files in build
+  hosting[goalbuddy-ai]: file upload complete
i  hosting[goalbuddy-ai]: finalizing version...
+  hosting[goalbuddy-ai]: version finalized
i  hosting[goalbuddy-ai]: releasing new version...
+  hosting[goalbuddy-ai]: release complete

+  Deploy complete!

Project Console: https://console.firebase.google.com/project/goalbuddy-ai/overview
Hosting URL: https://goalbuddy-ai.web.app
PS C:\Users\ocura\Desktop\sport-chatbot\goal-buddy>
```

Link to our chatbot:<https://goalbuddy-ai.web.app/>

Link to logo

creation:https://www.canva.com/design/DAGlvw_xpdl/YllbxArHiFpwwxNE5a47HQ/edit?utm_content=DAGlvw_xpdl&utm_campaign=designshare&utm_medium=link2&utm_source=sharebutton

Conclusion

This practical work was an enriching adventure full of learning. We started with the choice of a simple idea (among the ones proposed to us): create an application capable of providing real-time information on football matches. Through this project, we explored and mastered different technologies, each bringing its own benefit.

Indeed, this project not only allowed us to apply the theoretical concepts of artificial intelligence and web development, but also to discover the practical challenges and real solutions in the development of a modern application. We gained a deep understanding of the technologies used and learned to overcome technical obstacles, thereby strengthening our software development skills.

Finally, this project reminds us of the importance of collaboration and innovation. By combining powerful tools and creative ideas, we were able to turn a simple idea into a useful and powerful application. **GoalBuddy** is the result of this synergy between technology and imagination, and we are proud of what we have accomplished.

Thank you for your attention and support throughout this journey. We hope you find **GoalBuddy** as exciting and useful as we enjoyed developing it.