

National College of Ireland

BSc Honours Computing

Software Development

2023/2024

Anton Korzhan

x20510949

x20510949@student.ncirl.ie

StatFlowAI

Technical Report

Contents

Executive Summary	2
1.0 Introduction	2
1.1. Background	2
1.2. Aims.....	2
1.3. Technology	2
1.4. Structure	2
2.0 System.....	3
2.1. Requirements.....	3
2.1.1. Functional Requirements.....	3
2.1.1.1. Use Case Diagram	3
2.1.1.2. Requirement 1	3
2.1.1.3. Description & Priority.....	3
2.1.1.4. Use Case	4
2.1.1.5. Requirement 2	5
2.1.1.6. Description & Priority.....	5
2.1.1.7. Use Case	5
2.1.1.8. Requirement 3	6
2.1.1.9. Description & Priority.....	6
2.1.1.10. Use Case	6
2.1.2. Data Requirements	7
2.1.3. User Requirements	7
2.1.4. Environmental Requirements	7
2.1.5. Usability Requirements.....	7
2.2. Design & Architecture	8
2.3. Implementation	9
2.4. Graphical User Interface (GUI)	12
2.5. Testing.....	13
2.6. Evaluation	16
3.0 Conclusions	16
4.0 Further Development or Research	16
5.0 References	16
6.0 Appendices.....	17
6.1. Project Proposal	17

6.1.	Ethics Approval Application (only if required)	17
6.2.	Reflective Journals	17
6.3.	Other materials used	17

Executive Summary

The major parts of this report will be the requirements and implementation. The purpose of this report is to give an understanding of my idea, why I picked it, what its for and what technologies I used and how I used them to achieve this idea. Just in case it is not missed, here is the public domain for my website: stat-flow-ai.vercel.app.

1.0 Introduction

1.1. Background

I undertook this project because I am myself invested into the world of games and stats and would like to incorporate AI into an idea like this as I believe it would be useful. When a new players jumps into a game like Destiny 2 that is very vast and complex, they can get lost and confused easily, and might not know where to search or look for the information they need to progress, so I believe this AI will be able to assist that way for players new and returning. I also think it would be nice to create a website that can display different stats and data for users. To meet the objectives set out in 1.0, I will do extensive research, especially on API's, how to implement them together and learn how to put together a user friendly front-end that would be comfortable to use and display the data to users clearly. I will also look back to the AI project I did in my internship and get some inspiration from there.

1.2. Aims

This project aims to achieve a new way to assist people that play video games with their gameplay, progression and statistics. AI isn't used much in the data and statistics world of gaming, and I believe it is a missed opportunity in this field. AI could be put to great use for all types of people that play video games and might inspire similar things in other fields regarding the use of AI.

1.3. Technology

I used Next.js for the back and frontend of the project, since the Next.js framework supports both back and frontend development I was able to work and integrate both aspects of my project fluidly. For testing I used manual testing and some J-unit tests. For the API's I will use postman to test them. I will have a repo on GitHub and will use VSCode as my IDE. I will also host the website on Vercel so it can be accessed anywhere with the URL.

1.4. Structure

The structure of this project is as follows: First I will talk about the requirements of the project, state each main requirement and all their use cases and flows, then I will talk

about the design and architecture of the project and how I implemented it with supplied diagrams and code snippets. Then there will be some images of the GUI and after that I will talk about my steps through testing the website.

2.0 System

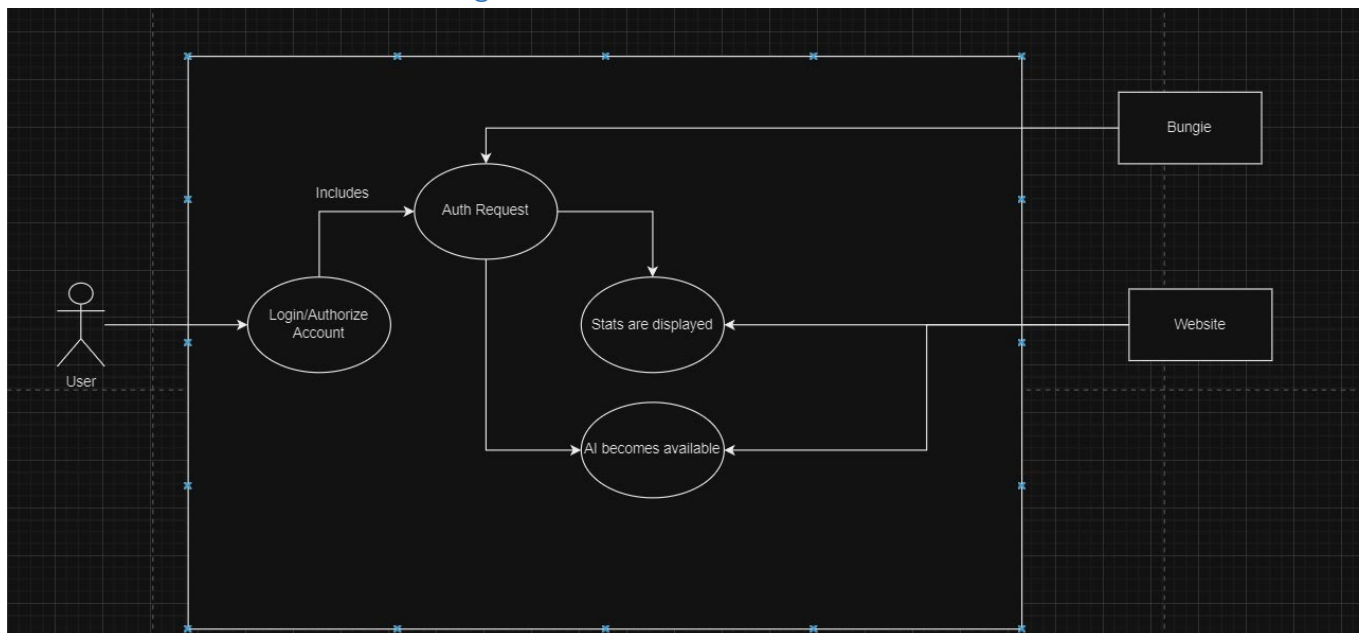
2.1. Requirements

2.1.1. Functional Requirements

Here are the functional requirements listed in order:

1. User login/authentication/authorisation
2. Data and Information Display
3. AI Bot
4. Dashboard
5. Responsive Design
6. Legal Compliance

2.1.1.1. Use Case Diagram



2.1.1.2. Requirement 1

Requirement 1: User Login/authenticate

2.1.1.3. Description & Priority

The Login/authenticate for the user so they can access the website and its features. This is the number 1 priority for a website such as this, as the safety of the user is vital.

2.1.1.4. Use Case

User clicks the authorise button and selects their platform where their account is created, then enters their email and password, and bungie will authenticate them. Then website brings user to main page after authorization.

Scope

The scope of this use case is to allow users to login/authenticate on the website with their game accounts.

Description

This use case describes the procedure that the user will take to login/authenticate their account on the website through bungie.

Use Case Diagram

Flow Description

Precondition

Must have an email address and password to login to platform related bungie account.

Activation

This use case starts when an <User> opens the website and clicks the Authorise button.

Main flow

1. The user opens the website
2. The user clicks the Authorise button
3. The user enters their email address and password to login
4. The system logs the user in to the website

Alternate flow

A1 : <User Registration>

1. The user opens the website
2. The user clicks the Authorise button
3. The user enters a valid email address and creates a password to register
4. The system registers the users account and logs them in to the website

Exceptional flow

E1 : <Invalid Details>

5. The user opens the website
6. The user enters invalid details
7. The system displays an error message

Termination

System terminates when the user closes the page or logs out.

Post condition**2.1.1.5. Requirement 2****Requirement 2: Data and Information Display****2.1.1.6. Description & Priority**

The data and information of the users account should be displayed to the user. This is priority number 2; the whole point of the website is to display information to the user of their account.

2.1.1.7. Use Case

The system will display the users data and info on the screen to them. The information/data displayed should be of the users account that they used to sign in with.

Scope

The scope of this use case is to have the system display all the necessary data of the users logged in account to them in a nice manner.

Description

This use case describes the procedure that the system will have to display the data to the user.

Use Case Diagram**Flow Description****Precondition**

User must have an existing account with said game and have logged in or registered the account on the website.

Activation

This use case starts when an <User> successfully signs/logs in.

Main flow

1. The user successfully logs/signs in with an account
2. The system displays the data of chosen game

Exceptional flow

E1 : <Account doesn't exist with the chosen game>

1. The user logs/signs in.
2. The system has no data to display to the user
3. The system displays an error message

Termination

System terminates when the user closes the page or logs out.

Post condition

[2.1.1.8. Requirement 3](#)

Requirement 3: AI Bot

[2.1.1.9. Description & Priority](#)

The AI Bot on the website that converses with the user to give valuable information, recommendations and advice on various things.

[2.1.1.10. Use Case](#)

User can type and converse with the AI bot, ask various destiny 2 related questions etc.

Scope

The scope of this use case is to create a AI bot that users can type to and ask various gaming related questions and get AI generated responses.

Description

This use case describes the procedure that the user will have to use the AI bot and converse with it.

Use Case Diagram

Flow Description

Precondition

Website must have a space and function where users can type to a AI bot and get AI generated responses.

Activation

This use case starts when an <User> types a message to the AI bot message box will generate a response.

Main flow

1. The AI bot sends an opening message to the user on the screen
2. The user types a gaming related query in the message box
3. The AI bot generates an adequate response

Exceptional flow

E1 : <Invalid Question/Message>

1. The AI bot sends a opening message to the user on the screen
2. The user types a non-gaming related question
3. The AI bot displays a message stating that it does not understand and cannot assist the user

Termination

System terminates when the user closes the page or logs out.

Post condition

The system goes into a wait state when the user closes the AI bot window.

2.1.2. Data Requirements

1. The data requirements are:
2. Users account credential data
3. Data on games used on the website
4. Users game account data
5. Open AI data

2.1.3. User Requirements

The user requirements are:

1. Log/Sign in functionality
2. Data displayed in a viewable manner
3. Data protection and account safety
4. Hardware features such as screen, keyboard etc
5. AI functionality, accurate generated responses

2.1.4. Environmental Requirements

There are no environmental requirements.

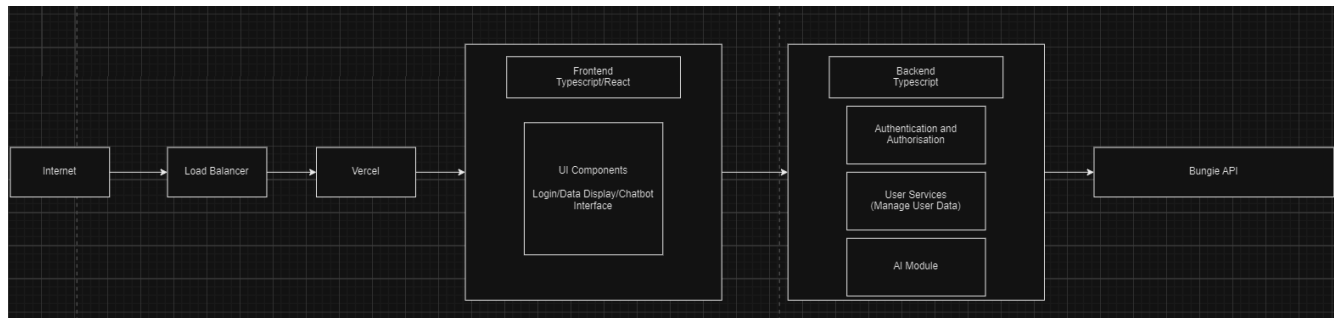
2.1.5. Usability Requirements

The usability requirements are:

1. An easy and usable interface that clearly directs the user
2. A clear and understandable presentation of data and information of the users account

3. AI uses understandable language to converse

2.2. Design & Architecture



1. **Load Balancer:** Distributes incoming traffic to multiple instances of your frontend and backend to ensure scalability and reliability.
2. **Vercel:** The website is hosted on Vercel, CI/CD process goes through here before displaying the frontend
3. **Frontend (Next.js with TypeScript/React):**
 - User Interface components for the website.
 - API Client communicates with the Backend API.
4. **Backend API (Next.js Typescript):**
 - Authentication & Authorization handle user authentication and authorization.
 - User Services manage user data.
 - Game Stats Service retrieves user game stats.
 - AI Module provides recommendations and advice.
5. **Public Game API:** The public API's of the games used that fetches the data of the user to be displayed.

2.3. Implementation

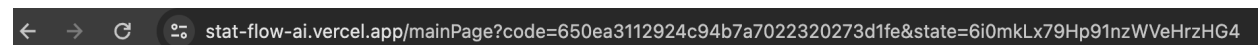
I have used my backend to store the API endpoints and my Key's are stored in a .env file. I fetch the necessary stats for the signed in account using their membership ID, the logic for fetching the stats:

```
async getAccountStats(
  membershipType: String,
  membershipId: String
): Promise<any> {
  const url = `${this.BASE_URL}/Destiny2/${membershipType}/Account/${membershipId}/Stats/`;
  const headers = { "X-API-Key": this.API_KEY };
  const response = await axios.get(url, { headers });
  const data = await response.data;

  if (response.status !== 200) {
    throw new Error("No destiny stats found");
  }

  return data;
}
```

I fetch the stats from the API, I create a certain endpoint that routes the backend to the frontend so it is able to be displayed. I also have a flexible "membershipId" which will allow me to put any membershipId in there to get not only my account but other people's accounts too. To do this I get a token by sending a POST request of a Authorisation code that appears as a param in the URL of the redirected website as shown here:



stat-flow-ai.vercel.app/mainPage?code=650ea3112924c94b7a7022320273d1fe&state=6i0mkLx79Hp91nzWVeHrzHG4

Code= is the authorisation code I need, then I send it as POST request which gets me my token, this token includes the membership ID of the current signed in user:

```
const Home = () => {
  useEffect(() => {
    const fetchData = async () => {
      setIsLoading(true);

      // Get access token if available
      const storedToken = localStorage.getItem("accessToken");
      if (storedToken) {
        setAccessToken(storedToken);
      } else {
        const urlParams = new URLSearchParams(window.location.search);
        const code = urlParams.get("code");
        if (code) {
          const response = await fetch(
            "https://www.bungie.net/platform/app/oauth/token/",
            {
              method: "POST",
              headers: {
                "Content-Type": "application/x-www-form-urlencoded",
                Authorization: `Basic ${window.btoa(
                  `${clientId}:${clientSecret}`
                )}`,
              },
              body: new URLSearchParams({
                grant_type: "authorization_code",
                code,
                redirect_uri: `${window.location.origin}/mainPage`,
              }),
            }
          );
          const data = await response.json();
          const { access_token, membership_id } = data;
          setAccessToken(access_token);
          localStorage.setItem("accessToken", access_token);
          localStorage.setItem("membershipId", membership_id);
        }
      }
    };
    fetchData();
  }, []);
}
```

Every endpoint to fetch stats requires a unique membership ID that every account has, so the point is I press the login button which allows the user to login with their account through bungie and their chosen platform, then bungie redirects them to the main page where the stats of the signed in user are loaded. The unique membership ID is acquired by the process described above.

For the ChatBot, I used OpenAI's API to be able to use their 3.5 turbo LLM. I have a secret key that I store in my .env file to access the API. For the backend side I set up the Chat messages, I set 3 different roles, System, Assistant and User. The system is where I prompt engineer the Chatbot AI to my liking, so it sends back the type of responses I want it to only.

The assistant role is where I set the welcome message that first appears to the user when the user is directed to the main page. This is the first message they see:

```
interface ContextProps {
  messages: OpenAI.Chat.ChatCompletionMessageParam[];
  addMessage: (content: string) => Promise<void>;
  isLoadingAnswer: boolean;
}

interface ChatCompletionResponse {
  data: any;
  choices: {
    message: string;
  }[];
}

const ChatsContext = createContext<Partial<ContextProps>>({});

export function MessagesProvider({ children }: { children: ReactNode }) {
  const { addToast } = useToast();
  const [messages, setMessages] = useState<
    OpenAI.Chat.ChatCompletionMessageParam[] | any
  >([]);
  const [isLoadingAnswer, setIsLoadingAnswer] = useState(false);

  useEffect(() => {
    const initializeChat = () => {
      const systemMessage: OpenAI.Chat.ChatCompletionMessageParam = {
        role: "system",
        content:
          "You are an expert on Video Games and especially the game Destiny 2, you know about all the weapons, gear and modes in Destiny 2,"
      };
      const welcomeMessage: OpenAI.Chat.ChatCompletionMessageParam = {
        role: "assistant",
        content: "Greetings Guardian, How can I help you today?",
      };
      setMessages([systemMessage, welcomeMessage]);
    };
  });
}
```

The user role is the messages that the user will write and send to the Chatbot AI to get a response.

```
const addMessage = async (content: string) => {
  setIsLoadingAnswer(true);
  try {
    const newMessage: OpenAI.Chat.ChatCompletionMessageParam = {
      role: "user",
      content,
    };
    const newMessages = [...messages, newMessage];

    setMessages(newMessages);

    const response: ChatCompletionResponse = await createMessage(newMessages);

    const reply = response.choices[0].message;
    // Add the assistant message to the state
    setMessages([...newMessages, reply]);
  } catch (error) {
    // Show error when something goes wrong
    addToast({ title: "An error occurred", type: "error" });
  } finally {
    setIsLoadingAnswer(false);
  }
};
```

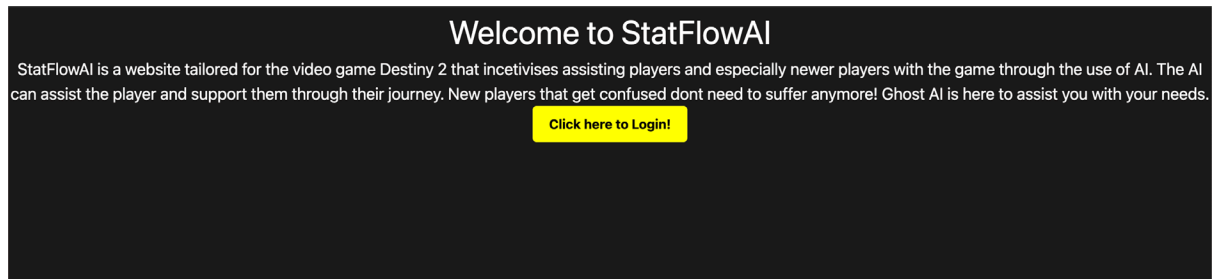
For the frontend, I created a messageList file where the messages are formed into a list with styling used from tailwind and my global.css file. This gives styling to the messages sent by the user and AI bot so they look neat and don't collide.

I also have a file for the messageBox user interface where the user will type messages and send them to the AI system for a response.

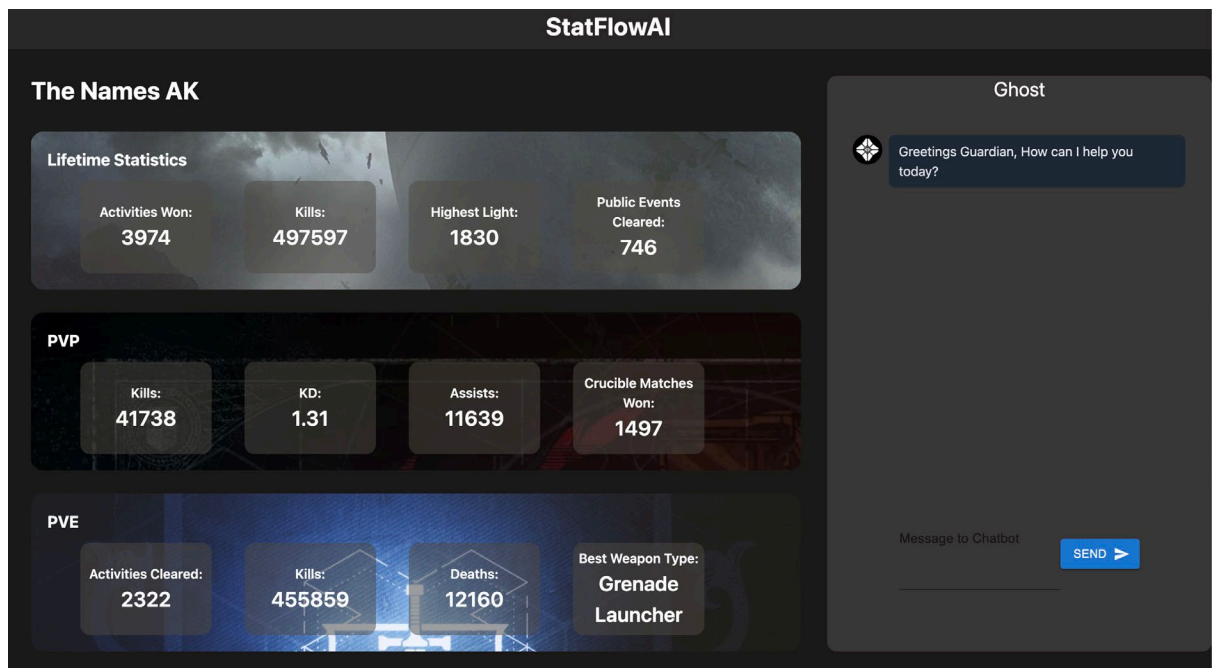
```
const MessageForm = () => {
  return (
    <form
      className="flex-none mx-auto max-w-3xl rounded-t-xl"
      onSubmit={handleSubmit}
    >
      <div>
        <TextField
          name="content"
          id="standard-multiline-static"
          label="Message to Chatbot"
          placeholder="Enter your message here..."
          multiline
          rows={3}
          defaultValue="Default Value"
          variant="standard"
          value={content}
          onChange={(e: any) => setContent(e.target.value)}
          sx={{
            "& .MuiInput-root": {
              color: "#ffffff",
              fontFamily: "Arial",
              fontWeight: "bold",
              // Bottom border
            },
          }}
        />
        <Button
          variant="contained"
          type="submit"
          className="absolute right-0 top-8"
          endIcon={SendIcon} />
      </div>
    </form>
  );
};
```

2.4. Graphical User Interface (GUI)

Here is the GUI of the login page:



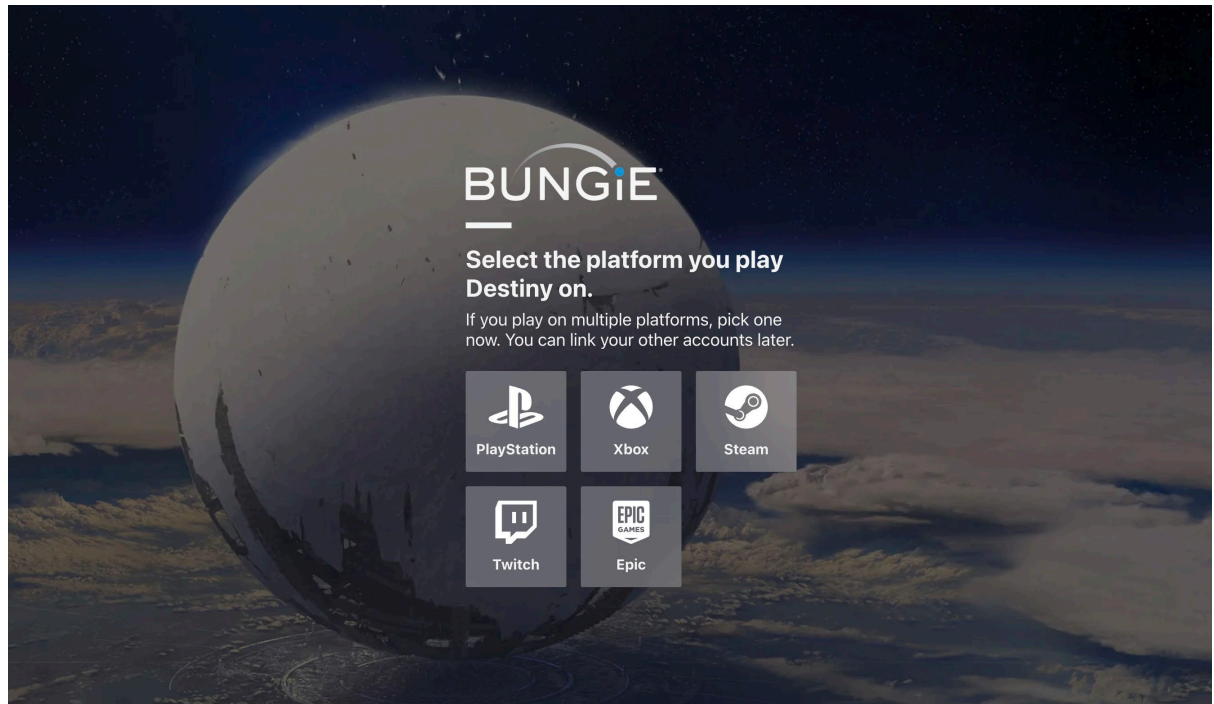
Here is the GUI of my website that displays the stats of the Authorised user and the AI bot:



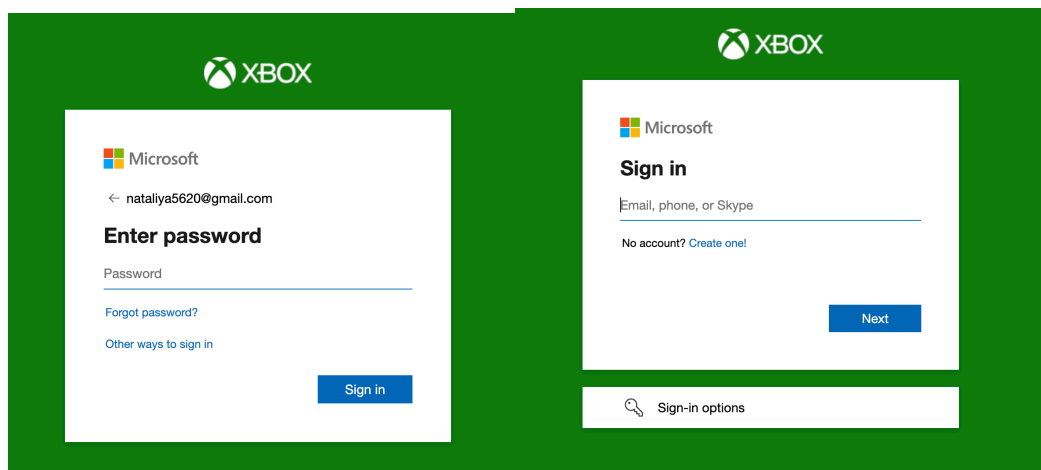
2.5. Testing

For testing, I will do manual test as well as prompt engineering test on the AI chatbot.

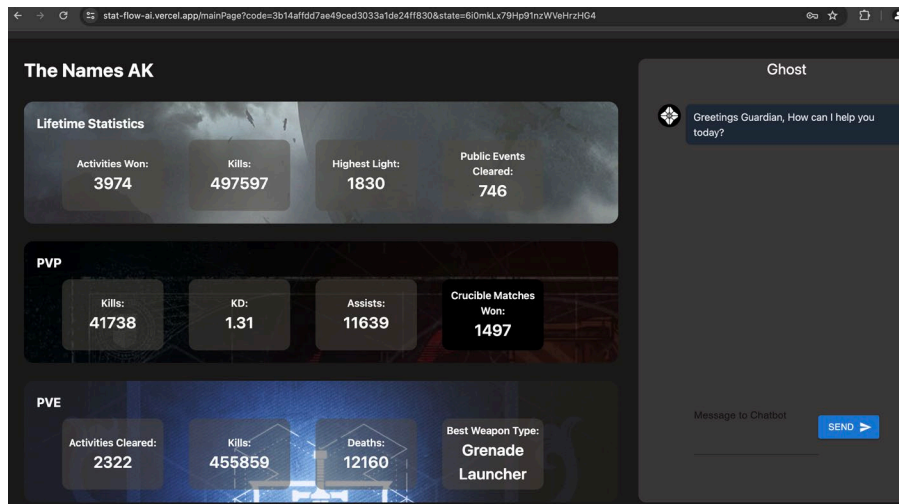
I checked to see if the login and authorisation worked, when clicking the login button I got sent to the bungie authorisation process:



I click my chosen platform which is xbox and login with my email and password:

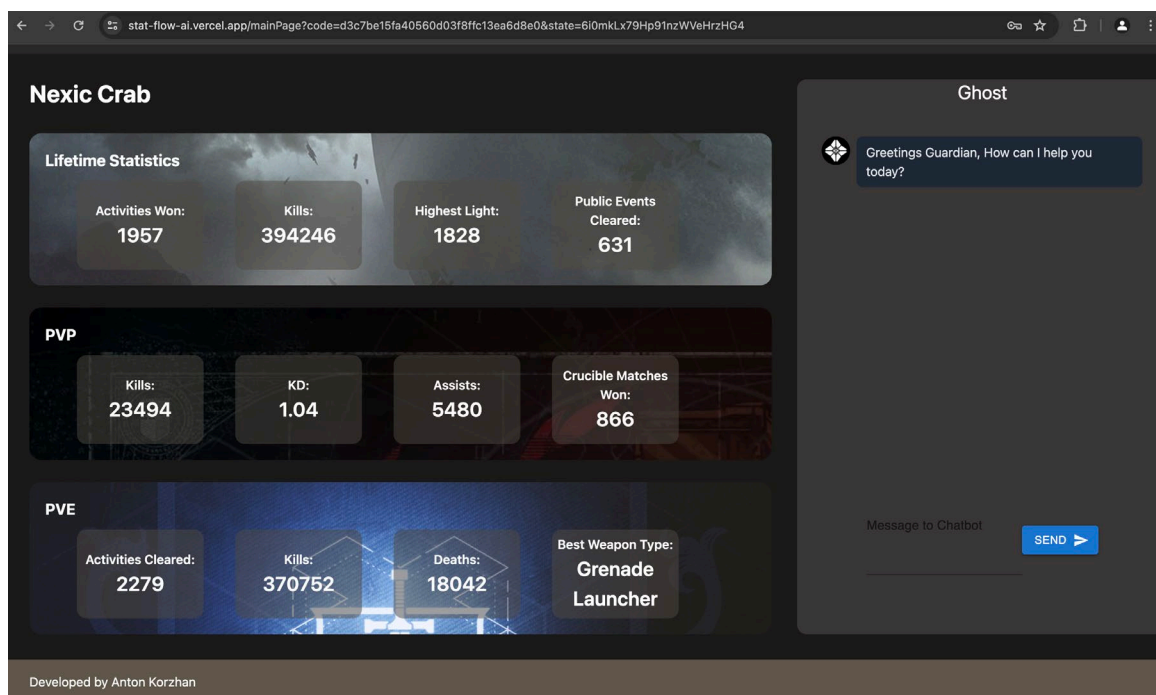


Then I got redirected to the main page of the website:



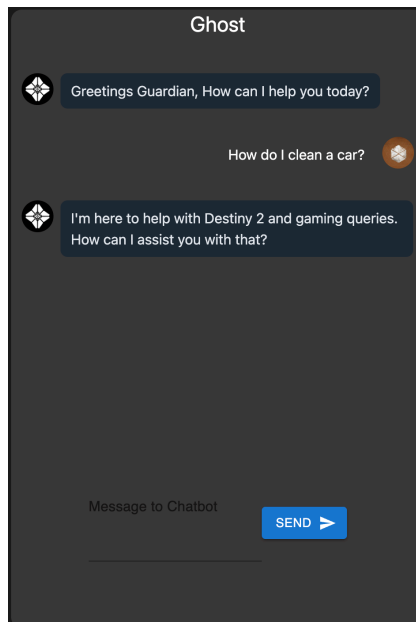
As you can see in the URL the Auth code is there, which means I successfully logged in and got the unique membership ID of the user which displayed their stats.

Here you can see I did the same process but with a different account and got that logged in accounts stats to be displayed:



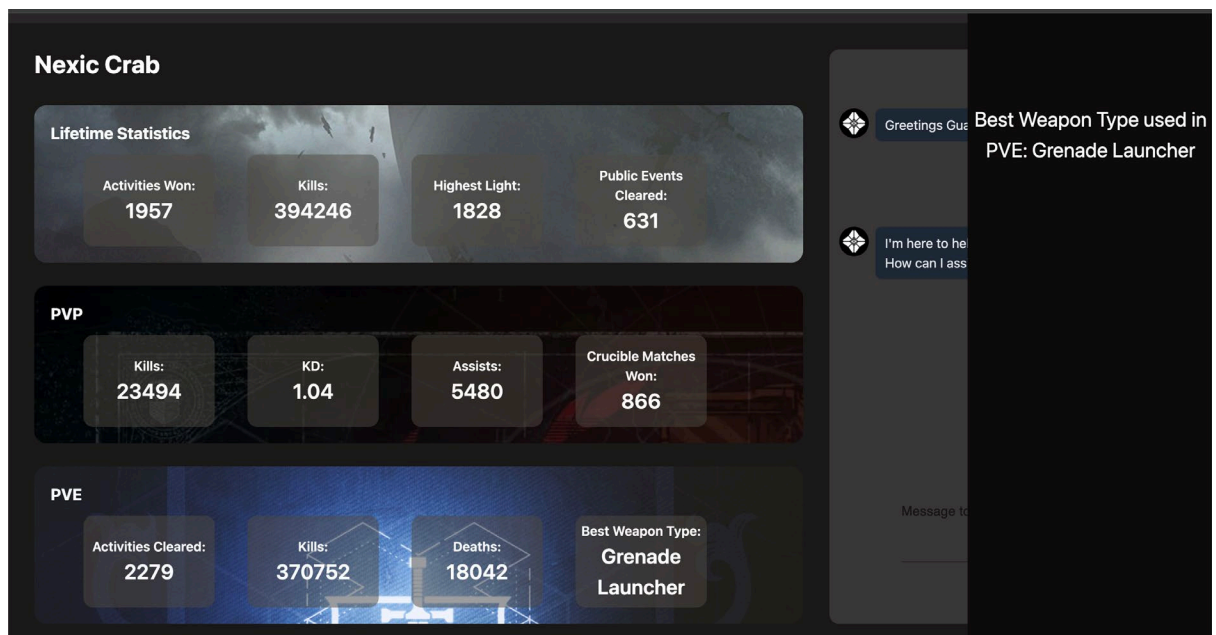
The tests were successful.

For prompt engineering, I tested different messages to be sent to the AI chatbot to see how it would react:



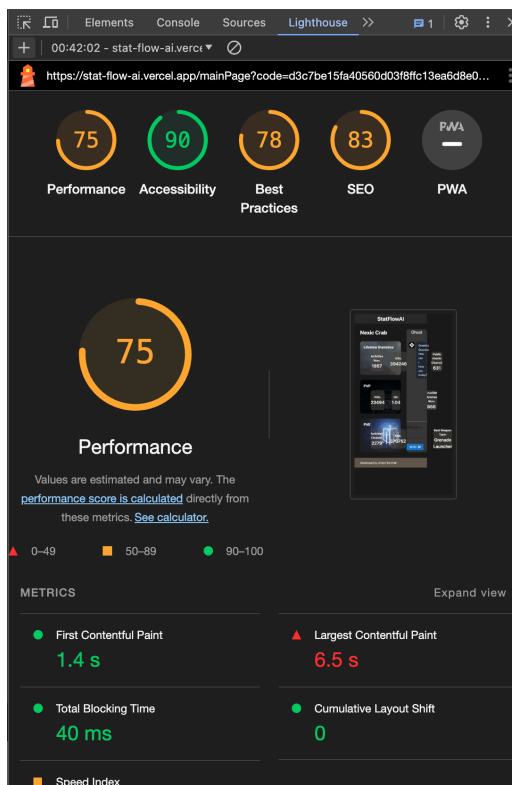
When I would give it a prompt that is unrelated to the topic of Destiny 2 and video games, the AI bot won't answer it and states it is here to help with Destiny 2 and gaming related queries.

When I click on any of the stat boxes, a side panel with additional information pops out from the right as shown here:



2.6. Evaluation

One way I evaluated the system was through lighthouse on inspecting the page, here are the results:



This shows me my performance, accessibility SEO etc of my website.

3.0 Conclusions

My conclusion with this project is that I truly believe that AI can be extremely helpful for players to interact with as it is an infinite supply of information that users can access at any time. It was challenging to get the website and logged in accounts to be dynamic but it worked in the end. Some limitations of this project were that AI as of right now isn't particularly great or able to read user info in code files. I wanted to implement that but wasn't able. Another limitation is the information that the bungie API will allow users to have, it is not always regularly updated with everything accessible so it was part of the challenge to decide what would be the most important stats to display to the user.

4.0 Further Development or Research

With additional time and resources, this project could be fleshed out much more than expected. It would include much more games, have stronger AI that would give even more - precise generated responses. Additional features such as in game item management etc could also be implemented, if the time and resources were available.

5.0 References

<https://platform.openai.com/docs/introduction>

<https://bungie-net.github.io/>

<https://github.com/Bungie-net/api/wiki/OAuth-Documentation>

<https://tailwindcss.com/docs>

<https://nextjs.org/docs>

<https://lowldev.com.au/destiny/authentication-2>

<https://www.w3schools.com/>

6.0 Appendices

6.1. Project Proposal



Project Proposal
Anton Korzhan x205

6.1. Ethics Approval Application (only if required)

6.2. Reflective Journals



Reflection Template
October.docx



Reflection Template
November.docx



Reflection Template
December.docx



Reflection Template
January.docx



Reflection Template
February.docx



Reflection Template
April.docx



Reflection Template
March.docx

6.3. Other materials used

Link to public domain: stat-flow-ai.vercel.app