

IT 5007 final report

Team Member

- Ma Qianli (A0250950)
- Song Tianqi (A0250964Y)
- He Can (A0255986H)

1. Introduction

Pokemon GAI is an AI-native, innovative, web-based application that utilizes generative artificial intelligence technology to create an engaging and unique gaming experience for Pokemon fans. This cutting-edge platform offers a range of exciting features, such as:

- 1. Unique Pokemon Generation:** By harnessing the power of AI, Pokemon GAI creates one-of-a-kind Pokemon characters complete with attributes, names, and values. These custom creatures are represented by NFT tokens, ensuring their uniqueness and collectible value.
- 2. Secure Login and Registration:** To access all the features of Pokemon GAI, users must log in or register. This process ensures user data protection and a personalized experience.
- 3. Personal Cloud Storage:** Pokemon GAI offers users their own personal cloud space, where they can store their Pokemon, personal profile, level, and other related information. This storage can be set to public or private, depending on the user's preference.
- 4. Social Interaction:** Pokemon GAI encourages interaction between players through its friends and contacts feature, allowing users to battle with their friends and save their contacts for easy communication.
- 5. Pokemon Chat Function:** Pokemon GAI introduces an interactive chat feature that allows users to communicate with their Pokemon, fostering a deeper bond and more immersive experience with their virtual companions.
- 6. Sharing Capabilities:** To further enhance the gaming experience, Pokemon GAI offers multiple sharing methods, including integration with popular platforms like Twitch, Telegram, Facebook, Instagram, and WeChat.

In summary, Pokemon GAI is a dynamic and engaging web-based application that offers a unique and immersive experience for Pokemon enthusiasts. By combining AI technology with a range of exciting features, Pokemon GAI provides users with a fresh and exciting way to explore

the world of Pokemon, battle with friends, chat with their Pokemon, and share their experiences with others.

2. Problem Statement

1. **Novelty of the problem:** The novelty of the Pokemon GAI project lies in the combination of generative artificial intelligence technology, NFTs, and social gaming features to create a unique and engaging experience for Pokemon fans. By leveraging AI to generate custom Pokemon and incorporating blockchain technology to ensure their uniqueness, Pokemon GAI provides users with a one-of-a-kind gaming platform.
2. **Challenges:** Several challenges need to be addressed in developing Pokemon GAI, including:
 - Efficiently implementing AI algorithms for generating unique Pokemon with balanced attributes, names, and values.
 - Integrating blockchain technology to create NFTs for each Pokemon, ensuring their uniqueness and collectible value.
 - Developing a user-friendly interface for personal cloud storage and maintaining data privacy and security.
 - Creating a seamless social interaction experience, allowing users to connect with friends, battle, and chat with their Pokemon.
 - Ensuring compatibility and integration with popular social media platforms for sharing features.
3. **Relevance in 2/5/10 years:** The problem of providing engaging and novel gaming experiences will remain relevant in the coming years. With the increasing popularity of AI, NFTs, and social gaming, Pokemon GAI is well-positioned to stay relevant and potentially adapt to new technologies and trends.
4. **The complexity of the solution:** The complexity of the Pokemon GAI solution involves the integration of various technologies and features, such as AI, NFTs, cloud storage, social interaction, and sharing capabilities. Developing a seamless and user-friendly experience while addressing challenges such as data privacy and security adds to the complexity of the solution.
5. **Understanding of the problem domain:** A comprehensive understanding of the Pokemon franchise, AI algorithms, NFTs, and social gaming mechanics is necessary to develop the Pokemon GAI platform effectively. By addressing the unique challenges and combining these technologies, the platform can provide an engaging and immersive experience for Pokemon fans, ensuring its success in the market.

3. Solution Architecture

3.1 Overall Architecture

Here is our overall architecture, including four-layer: frontend and user interface, application, server, infa.



- Frontend and user interface
 - React
 - User interface API
 - Authentification
 - Login
 - Register
 - Profile
 - Imagen
 - Gallery

- Friend
- Application
 - Image generation
 - 2D
 - 3D
 - Image get
 - Image save
 - Chatbot
 - Chat
 - Prompt template
 - langchain
 - Battle
 - Router
 - Schema
 - Core
 - Crud
- Sever
 - Huggingface space
 - Stable diffusion v2
 - Docker
 - Dreambooth
 - Pokemon GAI space
 - Gradio local servering
 - Gradio
 - OpenAI API
 - FastAPI
 - Router
 - Schema
 - Core
 - crud
- Infa

- ML infra
 - Transformers
 - Diffusers
 - Pytorch
 - Huggingface

3.2 Backend

3.2.1 Framework: FastAPI

[FastAPI](#) is a modern, high-performance web framework for building APIs with Python 3.7 and above, based on standard Python-type hints. It was developed by Sebastián Ramírez and was first released in 2018. The main goal of FastAPI is to enable developers to create robust and scalable applications with minimal coding effort and high efficiency.

Why FastAPI for our application?

1. **AI Native(most important!!!):** AI Native is a key feature of FastAPI to enhance its ability to handle artificial intelligence (AI) and machine learning (ML) models. FastAPI works well with deep learning frameworks such as TensorFlow, PyTorch, Scikit-learn, and MLflow frameworks like Gradio, and JinaAI. FastAPI will provide developers with an easy-to-use interface to deploy and serve AI/ML models in their applications and leverage FastAPI's high performance and scalability.
2. **Fast performance:** FastAPI is built on top of Starlette for the web parts and Pydantic for the data handling, which makes it one of the fastest Python frameworks available. Its performance is comparable to NodeJS and Go.
3. **Easy to learn and use:** FastAPI's syntax is straightforward and easy to understand. It helps developers to quickly create API endpoints with minimal code, using modern Python features like type hints and decorators.
4. **Automatic validation and documentation:** FastAPI uses the type hints provided in the function signatures to perform automatic data validation and generate interactive API documentation using OpenAPI and JSON Schema standards. This greatly reduces the time spent on manual validation and documentation tasks.
5. **Dependency injection:** FastAPI offers a built-in dependency injection system that simplifies managing dependencies in your application, making it easier to test, maintain, and scale.

6. **Asynchronous support:** FastAPI supports both synchronous and asynchronous code, enabling developers to write highly concurrent applications that can handle a large number of requests efficiently.
7. **Extensibility:** FastAPI is highly extensible, allowing developers to integrate it with various third-party libraries and plugins to fulfill specific requirements.

Overall, FastAPI is an excellent choice for building APIs, especially when speed, simplicity, and scalability are essential. Its features make it a popular choice among developers looking to build AI native applications using Python.

3.2.2 Database: SQLite

[SQLite](#) is a popular open-source relational database management system that provides a lightweight, file-based SQL database engine. It is widely used in embedded systems, mobile applications, and small-scale web applications due to its simplicity, ease of use, and low resource requirements.

Why SQLite for our application?

1. **Easy to set up:** SQLite is a file-based database, so it does not require a separate database server to be installed and configured. This makes it easy to set up and use with FastAPI.
2. **Lightweight:** SQLite is a lightweight database engine that requires very few system resources. This makes it ideal for small-scale applications where performance and resource usage are critical.
3. **Fast:** SQLite is optimized for read-heavy workloads and provides fast read access to data. This makes it suitable for applications that need to retrieve data quickly.
4. **Easy to use:** SQLite provides a simple SQL interface that is easy to learn and use. It supports a large subset of SQL commands, making it easy to work with existing SQL code or migrate data from other databases.
5. **Portable:** SQLite stores data in a single file, which makes it easy to move or copy between systems. This makes it a great choice for applications that need to be deployed across multiple platforms or environments.

Overall, SQLite is a great choice for small-scale applications that require a lightweight, easy-to-use database engine. FastAPI's built-in support for SQLite makes it easy to use with minimal configuration, allowing developers to focus on building their applications rather than managing their databases.

In Our Scenario:

- Once the app is being served on your local machine, **sql_app.db** will be created in our project root directory, including four tables `users`, `images`, `profiles`, `friends`.
- All binary images generated by our Generative AI model will be uploaded to **upload.io(Cloud Platform)** and the corresponding URL will be returned and stored in our database created before.

Tables (4)	<code>id</code>	<code>prompt</code>	<code>pokeType</code>	<code>pokeName</code>	<code>description</code>	<code>property</code>
<code>users</code>	Search column...	Search column...	Search column...	Search column...	Search column...	Search column...
<code>images</code>	1	1 fire greninja, wings, ...	2D	pokemon	NULL	BUG
<code>profiles</code>	2	2 fire greninja, wings, ...	2D	pokemon	NULL	GROUND
<code>friends</code>	3	3 fire greninja, wings, ...	2D	pokemon	NULL	DRAGON

3.2.3 Key API

Once the app is being served on your local machine, you can click the following links below.

- Backend Root: <http://127.0.0.1:8000>

You will see the JSON response as in your browser:

```
1 {"message": "Welcome to PokemanGAI!"}
```

- Interactive API docs: <http://127.0.0.1:8000/docs>

You will see the automatic interactive API documentation like this:

The screenshot shows the FastAPI documentation interface. At the top right is a green "Authorize" button with a lock icon. Below it, the "users" section is expanded, showing five endpoints: GET /users/ (Read Users), GET /users/me (Read Users Me), GET /users/{user_id} (Read User), POST /users/register (Create User) which is highlighted in green, and PUT /users/{username} (Update User). Below the users section is the "login" section, which is collapsed. It contains one endpoint: POST /login/ (Login For Access Token).

In the following part, we will abstract the main API and its functionalities, which are currently used in the back-end. More importantly, how to interact with them from the view of the users in the front-end. For more details, please use the API docs to have a rough look at them or deep down into our codebase. This part is located in the path **course-project-felix-felicitis/backend/app**.

3.2.3.1 Authentication

Authorize

Authorization Form:

1 username
2 password

In this section we build up a password flow to handle security and authentication. We use **OAuth2** to build that with FastAPI. Moreover, in our scenario, we use **JWT tokens** to codify JSON objects and implement password hashing using the algorithm **HS256**.

Here is the simplified point of view of password flow.

- The user types the `username` and `password` in the frontend, and hits `Enter`. This way works both on login section and registration section.
- The frontend sends that `username` and `password` to a specific URL in our API (<http://127.0.0.1:8000/login>).

- The API checks that `username` and `password`, and responds with a "token" (we haven't implemented any of this yet).
- The API checks that `username` and `password`, and responds with a "token".
- The frontend stores that token temporarily somewhere.
- The user clicks in the frontend to go to another section of the frontend web app.
- The frontend needs to fetch some more data from the API. It sends a header `Authorization` with a value of `Bearer` plus the token to authenticate with our API.

3.2.3.2 Login

(1) Operation: Login

HTTP Methods: POST

```
1 Path  
2 http://127.0.0.1:8000/login
```

Request

```
1 Query Parameters:  
2     null  
3  
4 Request Body:  
5     email: string  
6     password: string
```

Response

```
1 Response Body:  
2     message: string  
3     user_id: string  
4     access_token: string  
5     token_type: string
```

3.2.3.3 Register

(1) Operation: Register

HTTP Methods: POST

```
1 Path  
2 http://127.0.0.1:8000/register
```

Request

```
1 Query Parameters:  
2     email: string  
3  
4 Request Body:  
5     username: string  
6     password: string
```

Response

```
1 Response Body:  
2     message: string  
3     user_id: string  
4     access_token: string  
5     token_type: string
```

3.2.3.4 Profile

(1) Operation: Get Profile

HTTP Methods: GET

```
1 Path  
2 http://127.0.0.1:8000/profile
```

Request

```
1 Query Parameters:  
2     user_id: string  
3
```

```
4 Request Body:
```

```
5     null
```

Response

```
1 Response Body:  
2     id: string  
3     age: number  
4     role: string  
5     like: string  
6     motto: string  
7     contact: string  
8     avatar: string // image url  
9     owner_id: string
```

(2) Operation: Update Profile

HTTP Methods: POST

```
1 Path
```

```
2 http://127.0.0.1:8000/profile/info
```

Request

```
1 Query Parameters:
```

```
2     user_id: string
```

```
3
```

```
4 Request Body:
```

```
5     age?: number,  
6     role?: string,  
7     like?: string,  
8     motto?: string,  
9     contact?: string,
```

Response

```
1 Response Body:
```

```
2     message: string
```

(3) Operation: Upload Avatar

HTTP Methods: POST

```
1 Path  
2 http://127.0.0.1:8000/profile/avatar
```

Request

```
1 Query Parameters:  
2     user_id: string  
3  
4 Request Body:  
5     avatar: string
```

Response

```
1 Response Body:  
2     message: string
```

3.2.3.5 Imagen

(1) Operation: Generate Pokemon Image

HTTP Methods: GET

```
1 Path  
2 http://127.0.0.1:8000/imagen/generate
```

Request

```
1 Query Parameters:  
2     user_id: string  
3  
4 Request Body:  
5     prompt: string,  
6     poke_type: string,  
7     poke_name: string
```

Response

```
1 Response Body:  
2     pokemon_id: string  
3     pokemon_image: binary or string ?  
4     property: string  
5     hp: number  
6     attack: number  
7     star: number  
8     IQ: number  
9     MBTI: string
```

3.2.3.6 Gallery

(1) Operation: Get Gallery

HTTP Methods: GET

```
1 Path  
2 http://127.0.0.1:8000/imagen/gallery
```

Request

```
1 Query Parameters:  
2     user_id: string  
3  
4 Request Body:  
5     null
```

Response

```
1 Response Body:  
2     {  
3         pokemon_id: string  
4         pokemon_image: binary or string ?  
5         property: string  
6         hp: number
```

```
7     attack: number
8     IQ: number
9     MBTI: string
10    }[]
```

3.2.3.7 Friend

(1) Operation: Add Friend

HTTP Methods: POST

```
1 Path
2 http://127.0.0.1:8000/friend/add
```

Request

```
1 Query Parameters:
2     user_id: string
3
4 Request Body:
5     friend_id: string
```

Response

```
1 Response Body:
2     message: string
3     friend_name: string
```

(2) Operation: Get a Friend List

HTTP Methods: GET

```
1 Path
2 http://127.0.0.1:8000/friend/list
```

Request

```
1 Query Parameters:  
2     user_id: string  
3  
4 Request Body:  
5     null
```

Response

```
1 Response Body:  
2     {  
3         user_id: string  
4         user_name: string  
5         avatar: string  
6     }[]
```

(3) Operation: Get Friend's Gallery

HTTP Methods: GET

```
1 Path  
2 http://127.0.0.1:8000/friend/gallery
```

Request

```
1 Query Parameters:  
2     friend_id: string  
3  
4 Request Body:  
5     null
```

Response

```
1 Response Body:  
2     {  
3         pokemon_id: string  
4         pokemon_image: binary or string ?  
5         property: string  
6         hp: number  
7         attack: number  
8         IQ: number
```

```
9     MBTI: string  
10    }[]
```

(4) Operation: Delete a Friend

HTTP Methods: POST

```
1 Path  
2 http://127.0.0.1:8000/friend/delete
```

Request

```
1 Query Parameters:  
2     user_id: string  
3  
4 Request Body:  
5     friend_id: string
```

Response

```
1 Response Body:  
2     message: string
```

3.3 Frontend

3.3.1 Framework : React

React is an open-source JavaScript library for building user interfaces, primarily for single-page applications. It was developed and is maintained by Facebook, with the first public release in May 2013. React's primary goal is to make it easier to create interactive and dynamic UI components that efficiently update and render as data changes.

Why use React for our application?

- 1. Component-based architecture:** React promotes the creation of reusable UI components, which can be combined and composed to build complex user interfaces. This modular approach makes it easier to maintain and scale applications.

2. **Virtual DOM:** React uses a virtual representation of the actual DOM (Document Object Model) to optimize performance. When a component's state changes, React calculates the difference between the new virtual DOM and the current one (called "diffing") and updates only the parts of the actual DOM that have changed. This process is known as "reconciliation" and helps to minimize browser manipulation, improving performance.
3. **Unidirectional data flow:** React enforces a one-way data flow, ensuring that data is passed down from parent components to their children through "props" (properties). This makes the flow of data more predictable and easier to understand, which is particularly useful when working with large applications.
4. **JSX:** React introduces a syntax extension called JSX (JavaScript XML), which allows developers to write HTML-like code within their JavaScript code. JSX makes it more convenient to define and manipulate the UI components and their respective states.
5. **React Hooks:** Introduced in React 16.8, Hooks are a set of functions that enable developers to use state and other React features in functional components, without the need to write class components. Hooks simplify component logic and promote the use of functional components, which tend to be more concise and easier to test.
6. **Ecosystem and community:** React has a large and active community, which has contributed to a rich ecosystem of libraries, tools, and resources. This makes it easy to find solutions, support, and additional functionality for various use cases.

Overall, React is a popular and powerful library for building user interfaces, with a strong emphasis on performance, maintainability, and ease of use. Its component-based architecture and extensive ecosystem make it an excellent choice for developers looking to build modern web applications.

3.3.2 Design Principle

Before designing the prototype, we have set several design principles to follow:

1. Color Scheme:

The primary color palette consists of two main colors: #947FF8 and #FED851. These colors will be used consistently across all design elements, including backgrounds, text, and graphics. The secondary color palette can be developed by selecting complementary colors that harmonize well with the primary colors.

 #947FF8

 #ED9111

2. Typography:

The font family used for this design is Inter, which is a clean and modern sans-serif typeface. The two font sizes specified are 16px and 20px, both of which have an italic font style. For the font weight, 300 and 900 will be used to create contrast between different types of text. The line height for both font sizes is set to 16px and 20px, respectively. Finally, letter spacing is set to 0em, which ensures that text is evenly spaced out and easy to read.

3. Layout and Composition:

The design will have a left-aligned text layout to create a clear and organized structure. White space will be used generously to separate different design elements and create a visual hierarchy. Visual elements like images, icons, and graphics will be used sparingly to avoid clutter and distraction.

4. Accessibility:

The design will be created with the goal of being accessible to all users. The text will be large and easy to read with a sufficient contrast between the text and background. Alt-text will be used for all images to ensure those visually impaired users can access the same information as sighted users.

By following these design principles, the resulting design will be visually appealing, easy to read, and accessible to all users.

3.3.3 UI Design

Firstly, we use [Figma](#) for UI design.

Figma is a powerful platform that boasts an array of useful features. It provides a **convenient** way to create and replicate design styles. For instance, Figma offers a variety of pre-designed UI components that can be customized to fit the desired aesthetic of a project. Additionally, Figma facilitates collaboration among team members, enabling seamless sharing of designs and real-time feedback.

In addition, Figma is a **cloud-based** platform, meaning that it is accessible from anywhere with an internet connection, making it easy for team members to work remotely. Figma also offers a robust set of tools for creating and editing vector graphics, making it easy to create custom icons and graphics for a project.

You can view the design of our project from the following link:

Link: [PokemonGAI](#)

Overview:

Screenshot of Figma interface showing the 'Log In' page of a mobile application. The page has a blue-to-pink gradient background. It features a logo at the top left, a 'Register' button at the top right, and a 'Log In' button. Below these are two input fields: 'Name' (with placeholder 'Hollie77') and 'Password' (with placeholder '*****'). A 'CONFIRM' button is at the bottom. The Figma sidebar on the left shows a list of pages, and the right sidebar shows design, prototype, and inspect tools.

Screenshot of Figma interface showing multiple screens of a mobile application. The screens include:

- MacBook Pro 14" - 1:** Log In screen with a blue-to-pink gradient background, featuring a logo, 'Register' and 'Log In' buttons, and input fields for Name and Password.
- MacBook Pro 14" - 2:** Register screen with a blue-to-pink gradient background, featuring a logo, 'Register' and 'Log In' buttons, and input fields for Email, Password, and Email.
- MacBook Pro 14" - 3:** Home screen for user 'Hollie77' with sections for 'How to level up?', 'Poké Ball', 'Pokeball', and 'Graph'.
- MacBook Pro 14" - 4:** Profile screen for user 'Hollie77' showing stats like 'Day Off Cap' and 'Pokeball'.
- MacBook Pro 14" - 5:** Inventory screen showing items like 'Pokeball', 'Potion', and 'Elixir'.
- MacBook Pro 14" - 6:** Friends screen showing a list of friends: 'Hollie77', 'Hollie77', 'Caroline_123', 'Felicie', and 'Zoofyfate'.

The Figma sidebar on the left shows a list of pages and components, and the right sidebar shows design, prototype, and inspect tools.

Apart from Figma, we also use [Ant Design](#) for our projects. Ant Design is a popular UI library that provides a wide range of pre-built components for creating high-quality web applications. It is built on top of the React framework and offers a variety of tools and resources to help developers create visually appealing and responsive web applications with ease. We have included components such as [Popover](#), [Slider](#), and [Tag](#).

3.3.4 Usability

- **URL link:** We recognize the importance of ensuring that all links are functional, as broken URL links can negatively impact the user experience and lead to penalties from search engines. In our project, we will direct you to an error page if the URL is not available.

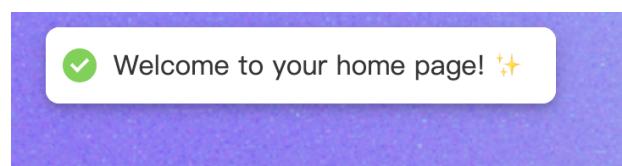
404 Page not found

Please check if you use the correct URL QwQ

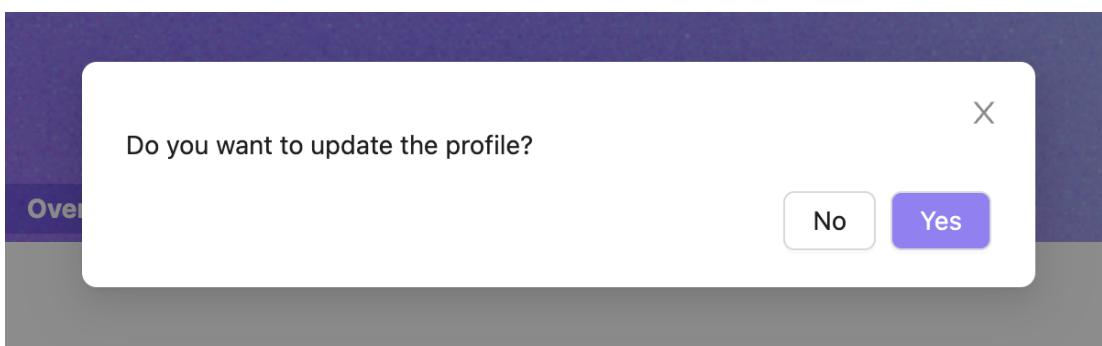


- **Popover and Message Box:** In PokemonGAI, all icons and buttons are designed with Popover or Message box. To ensure the best possible experience for users, we have conducted testing on multiple browsers to ensure compatibility and smooth performance.

- Welcome page message:



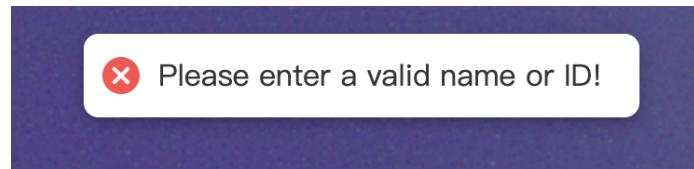
- Edit file message:



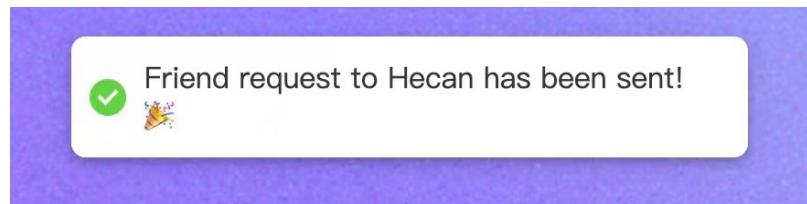
- Avatar popover message:



- Add a friend message:



- Add a friend message:



- **Form validation:** Finally, our project includes form validation, which is crucial to ensuring that user input is accurate and does not cause errors or other issues down the line. For example, on the login page, the email format will be automatically checked to see if it's consistent with the required format.



By prioritizing these critical aspects of usability and performance, we aim to create a seamless and enjoyable user experience that meets the needs of all users.

3.3.5 Documentation

We have created comprehensive documentation for our code, which includes both inline and full descriptions. For instance, we provide comments about the purpose of each component and also include inline comments to explain how specific pieces of logic work.

```
1 const Overview = () => {
2     /*
3         Overview page should display:
4         - User's name
5         - User's number of victories
6         - User's number of battles
7         - User's gallery of pokemon
8     */
9     const [name, setName] = React.useState("");
10    const [victories, setVictories] = React.useState(0);
11
```

By including these detailed explanations, we make it easier for other developers to understand our code and quickly make any necessary updates or modifications. The inline comments style is as follows:

```
1 const handleSubmit = async (e: React.FormEvent) => {
2     e.preventDefault();
```

```
3
4      // validate email and password
5      if (!email || !password) {
6          toast.error("Please enter your email and password");
7          return;
8      }
9      try {
10         const response = await postLogin(email, password);
11     } catch (error) {
12         toast.error("An error occurred, please try again later");
13     }

```

3.3.6 Features

- **Route**

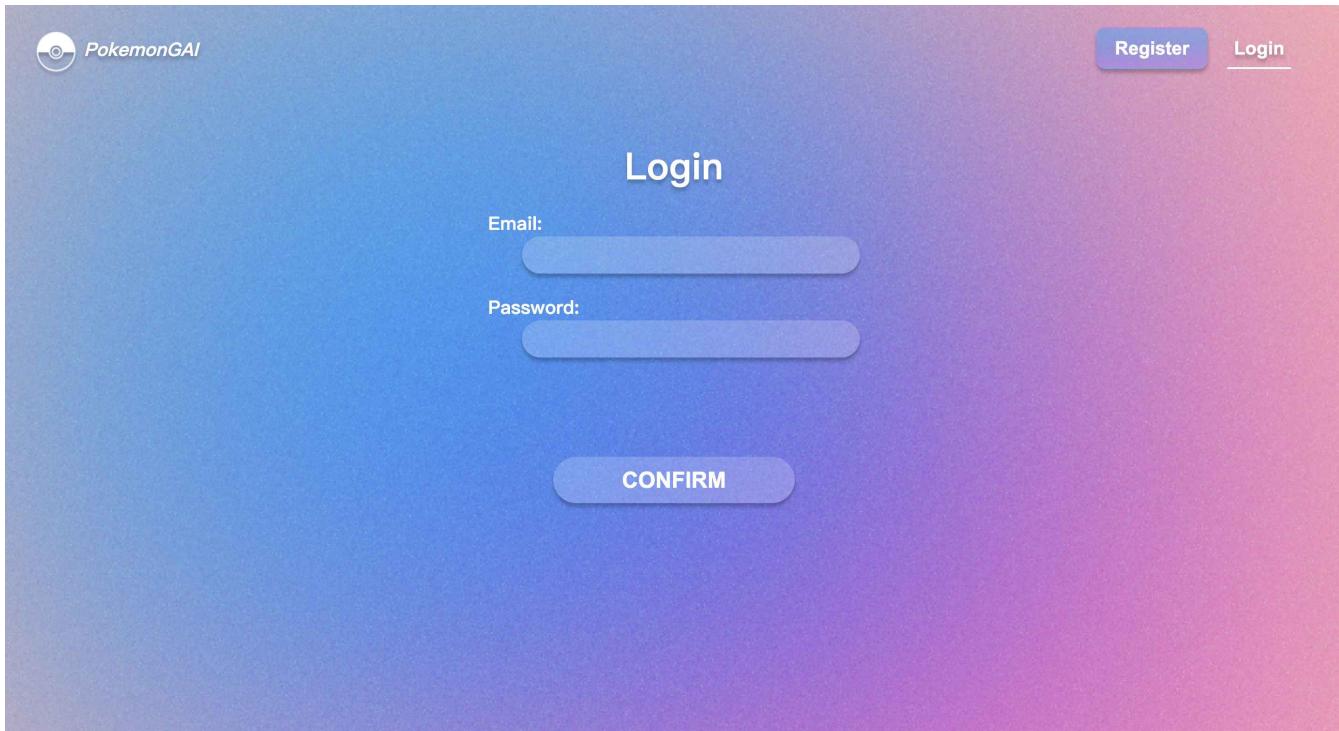
- We use router components from `react-router-dom` package.

```
1 import { BrowserRouter as Router, Routes, Route } from "react-router-dom";
```

- This is a route component in a React application, which uses the React Router library to define and render different pages based on the current URL path. The component defines several routes, including a login page, a home page with several sub-pages, and an error page to handle any invalid URLs.
 - The code defines each route using the `<Route>` component, which specifies the URL path and the component to render for that path. The `element` prop is used to specify the component to render for each route.

```
1 <Router>
2     <Routes>
3         <Route path="/" element={<LoginPage />} />
4         <Route path="/login" element={<LoginPage />} />
5         <Route path="/home" element={<HomePage />} >
6             <Route path="overview" element={<Overview/>} />
7             <Route path="create" element={<Create />} />
8             <Route path="gallery" element={<Gallery/>} />
9             <Route path="friends" element={<Friends/>} />
10        </Route>
11        <Route path="*" element={<ErrorPage/>} />
12    </Routes>
```

- For the home page, the component defines several nested routes using the <Route> component within the <HomePage> component. These nested routes correspond to different sub-pages within the home page, such as the overview page, create page, gallery page, and friends page.
- **Pages**
 - **Login page:** <http://localhost:3000/login>
 - i. This page is where users can enter their login credentials to access the application. It includes fields for entering a username and password, as well as options for recovering a forgotten password or creating a new account.



- **Overview page:** <http://localhost:3000/home/overview>
 - This page provides a high-level view of the application, typically including summary information or statistics related to the user's account or activity. It includes charts, graphs, or other visualizations to help users quickly understand the key data of PokemonGAI.

Hollie77
ID: PK230204

Age: 23
Role: Pokemon Trainer
Like: Pikachu
Motto: Since you asked sincerely
Contact: hollie77@pkc.org

Hi, this is Hollie77!

- I have accomplished...
 - 14 victories in battles
 - 12 battles all together
- I have collected...
 - 5 Pokemon
- This is my Collection...

Attack Intelligence Speed Power Characteristic

Original Performance

Jan Feb Mar Apr May Jun

Mon Wed Fri

- **Create page:** <http://localhost:3000/home/create>

- This page is used to create new Pokemon within the application. Depending on the application, this includes creating a new Pokemon's name, adding new descriptions, and selecting the type of the pokemon.

Hollie77
ID: PK230204

Age: 23
Role: Pokemon Trainer
Like: Pikachu
Motto: Since you asked sincerely
Contact: hollie77@pkc.org

Enter your Pokemon's name
For example, Pikachu

Describe your Pokemon's characteristics
For example, its height, weight, and abilities

Select the type of your Pokemon
2D

GENERATE

localhost:3000/home/gallery

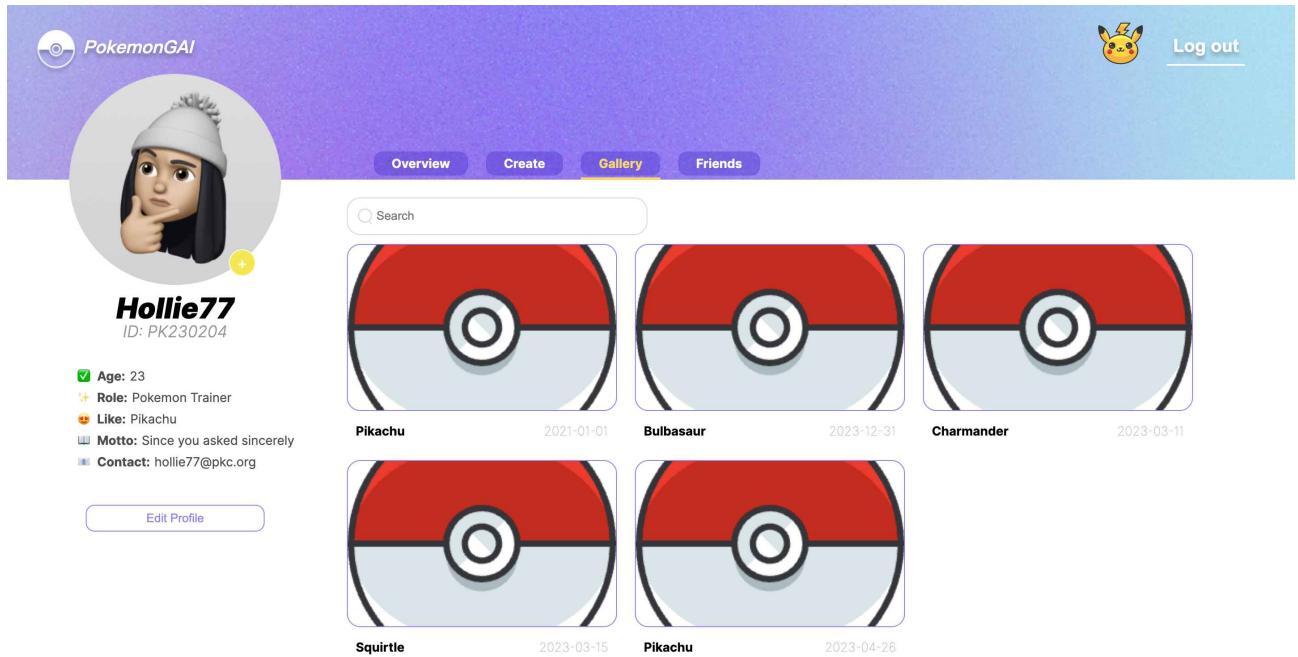
There is no generation task right now

Z

Pikachu sleeping

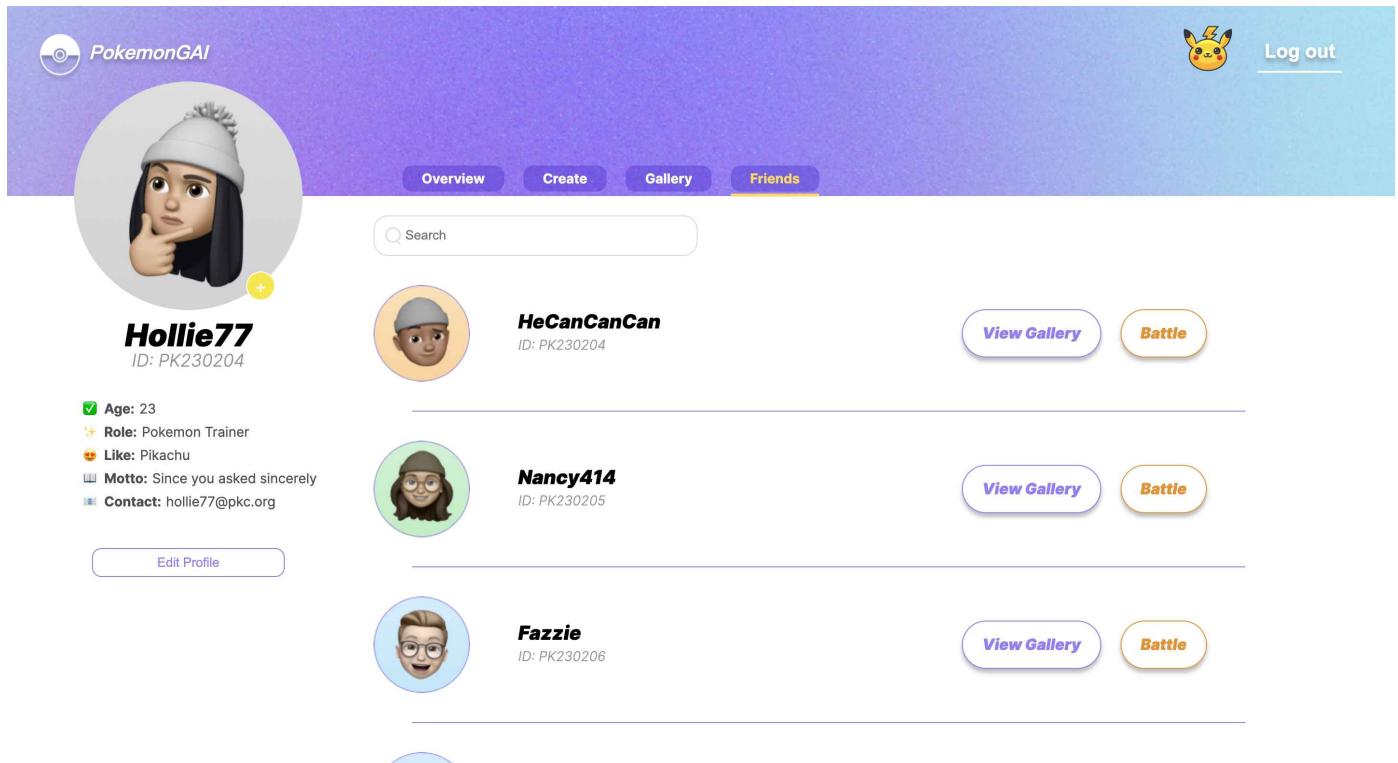
- **Gallery page:** <http://localhost:3000/home/gallery>

- This page displays a collection of images or other media, organized in a visually appealing way. Users can browse through the gallery and view or download individual items as desired.



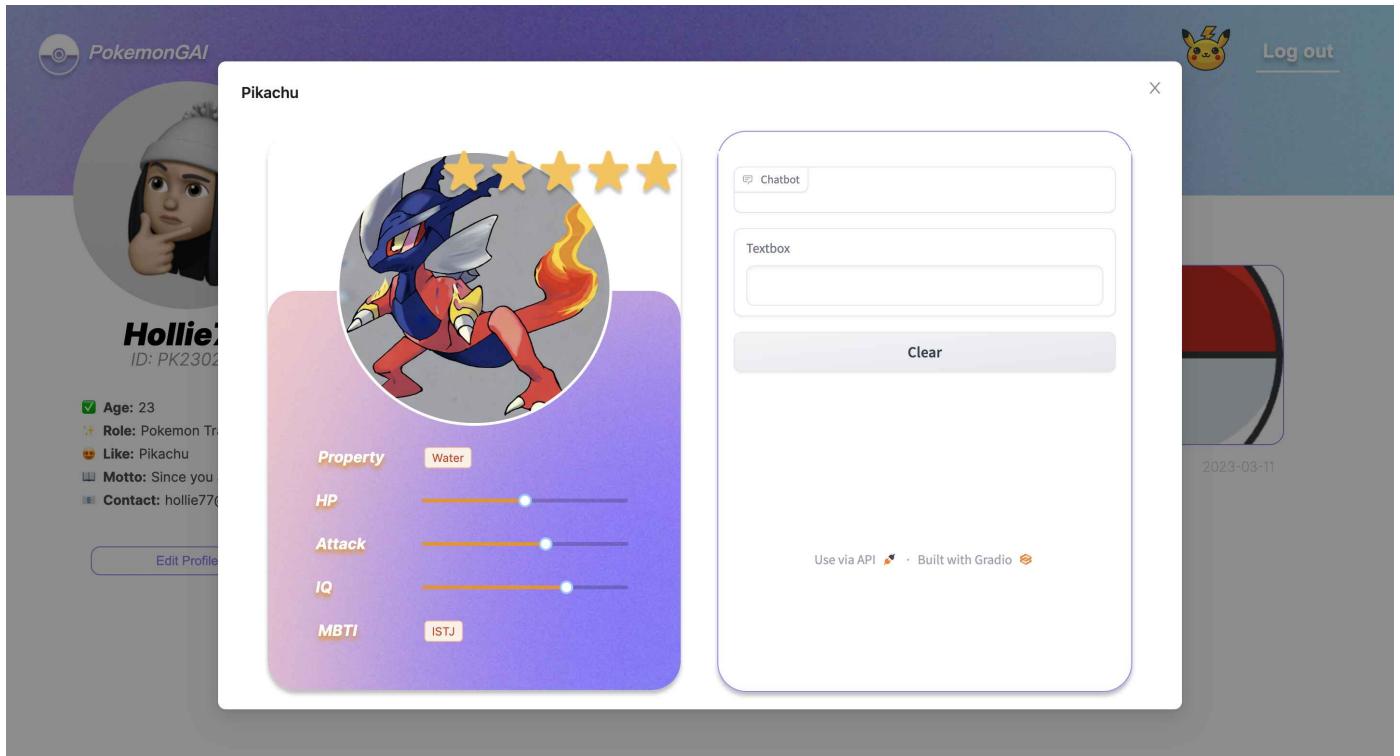
- **Friends page:** <http://localhost:3000/home/friends>

- This page is used in social applications to display a user's friends or contacts. It includes options for viewing existing friends, viewing their gallery, and starting a battle with friends.

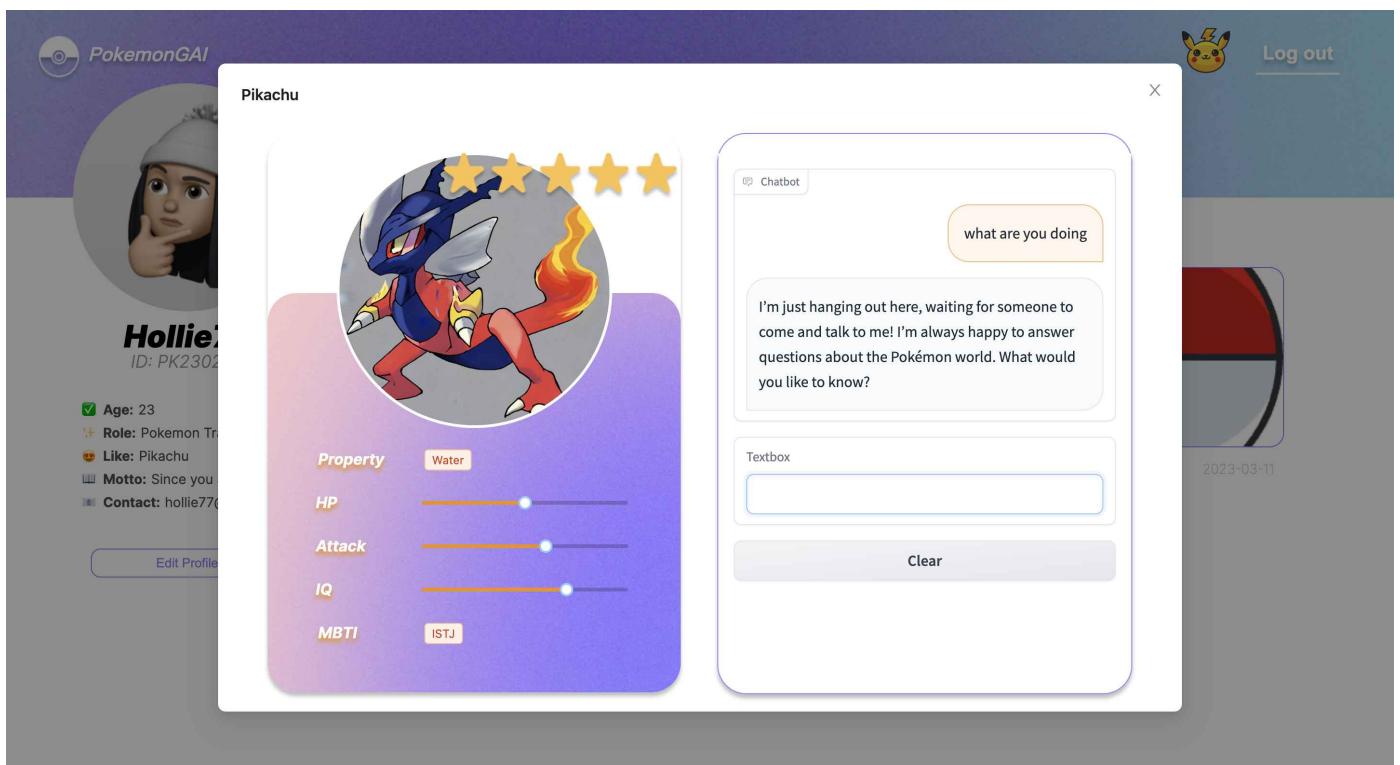


- **Chatbot**

- We have incorporated a chatbot for each Pokemon. The UI is designed as an embedded component next to the Pokemon card.



The chatbot is designed with a prompt-fitting Pokemon world. So whenever starting a conversation with it, it will reply to the identity of your Pokemon.



As we can see, the chatbot is working smoothly. Try it! It's fun :)

Pikachu

can you make a fire

Unfortunately, I'm not able to make fire. However, I do know some Pokémons that can! Charizard, Magmar, and Arcanine are all capable of using their fire-type moves to create fire.

what can you do

Textbox

Hollie ID: PK2302

Age: 23
 Role: Pokemon Trainer
 Like: Pikachu
 Motto: Since you asked
 Contact: hollie7702@gmail.com

Edit Profile

Pikachu

Magmar, and Arcanine are all capable of using their fire-type moves to create fire.

what can you do

I'm not able to do much, but I can provide interesting perspectives and suggestions about the Pokémons world. I'm also able to answer questions about Pokémons and provide advice on how to best use them in battle.

Textbox

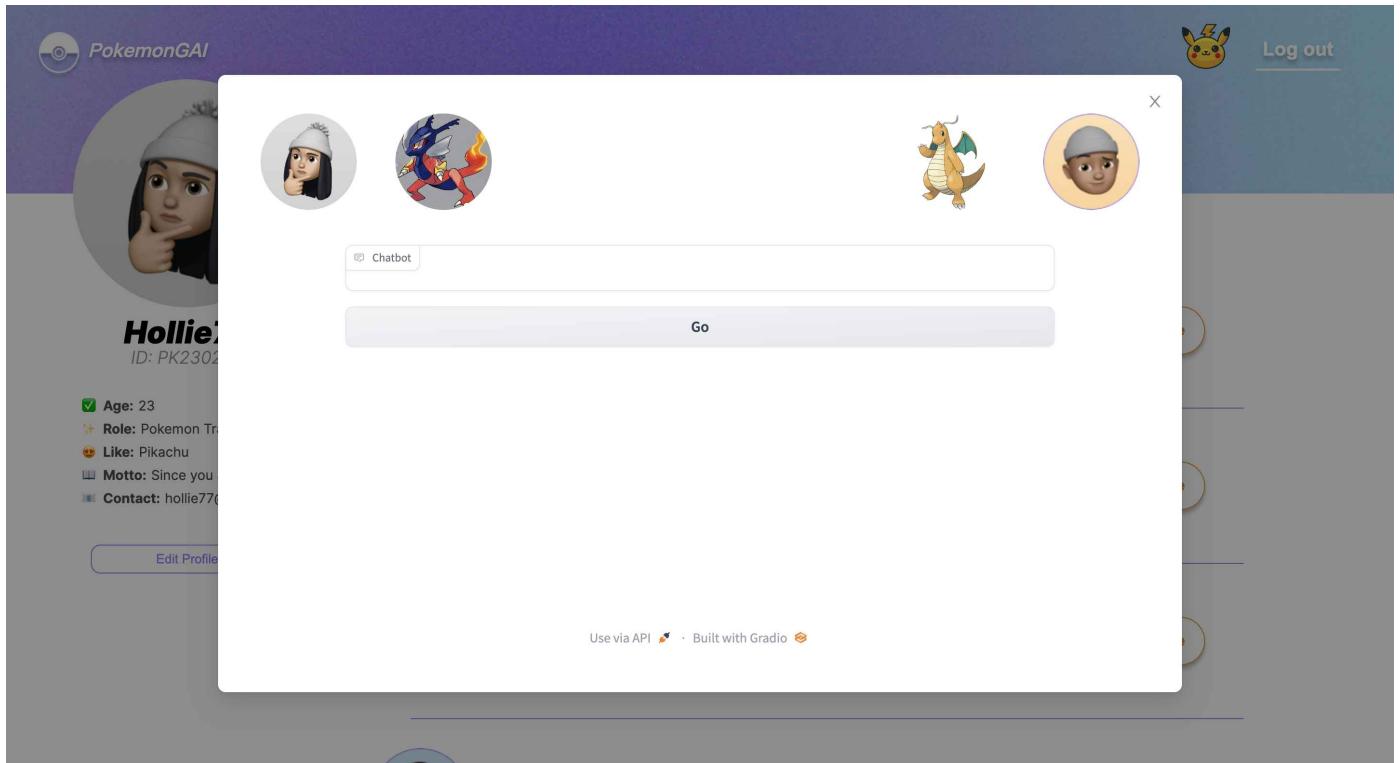
Hollie ID: PK2302

Age: 23
 Role: Pokemon Trainer
 Like: Pikachu
 Motto: Since you asked
 Contact: hollie7702@gmail.com

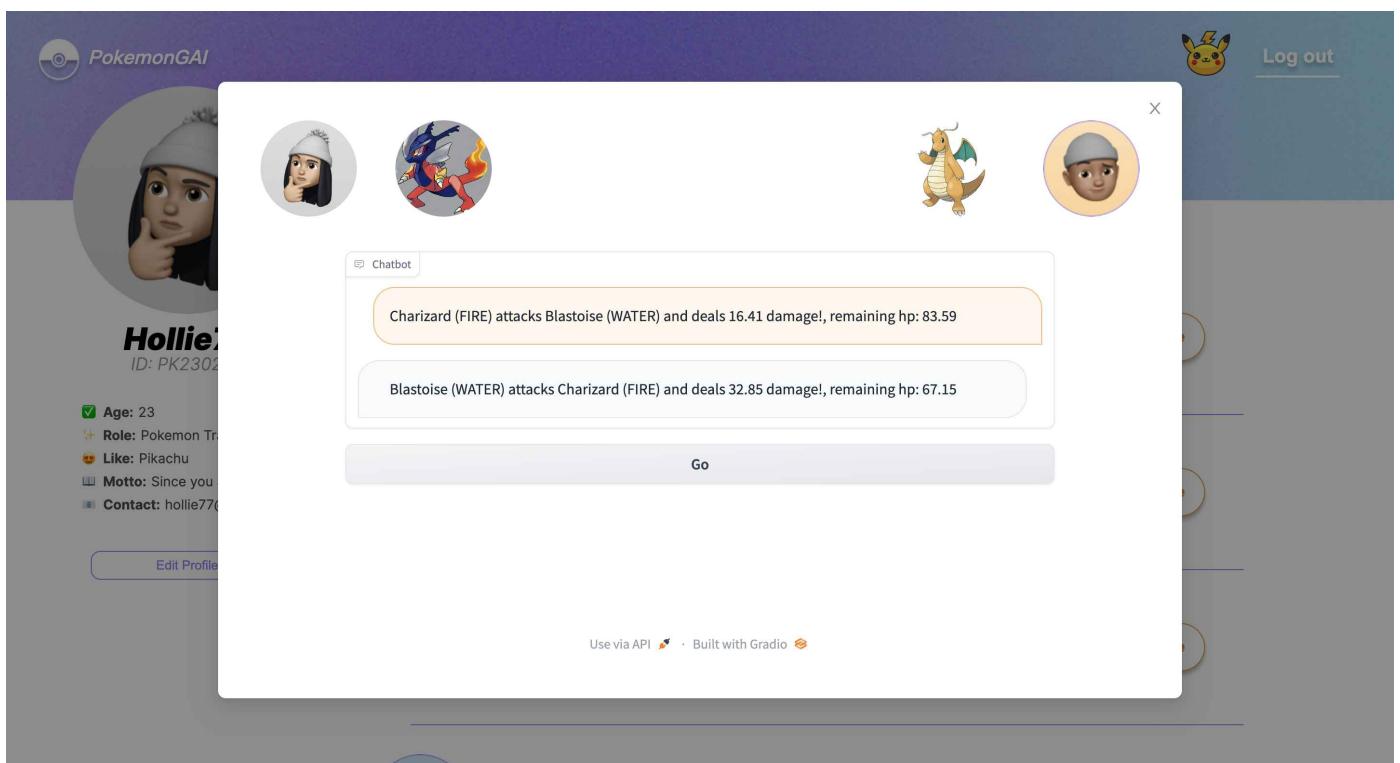
Edit Profile

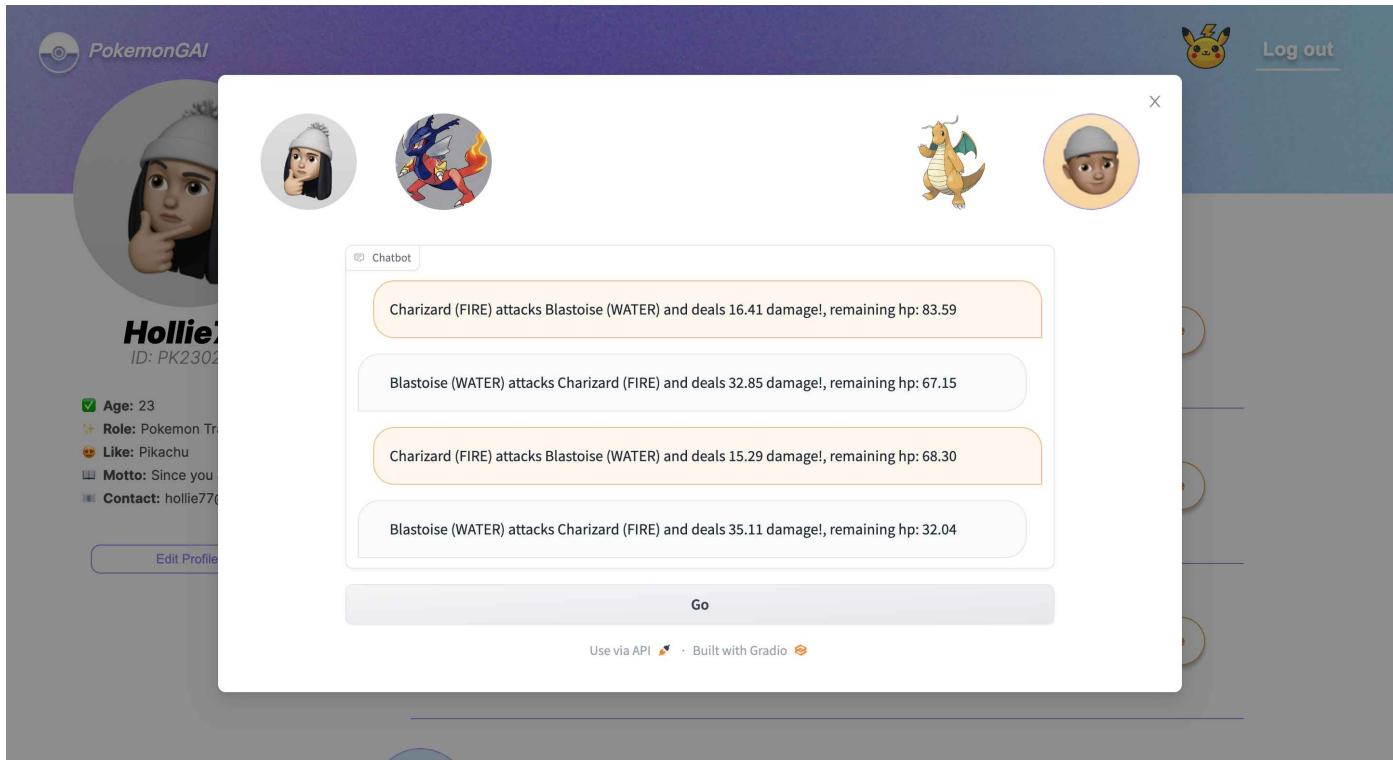
• Battle

- To maintain design consistency, the battle layout is also designed in the format of chatting.



As we can see, for each round, you can click on the button to start a battle.



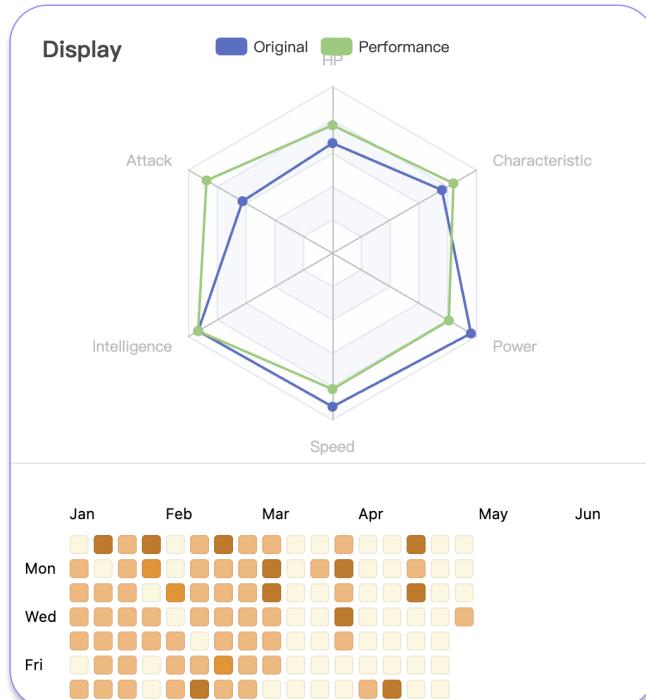


- **Visualization**

- We use visualization methods to enrich our front-end display.

```
1 import * as echarts from "echarts";
2 import { Heatmap } from "contribution-heatmap";
```

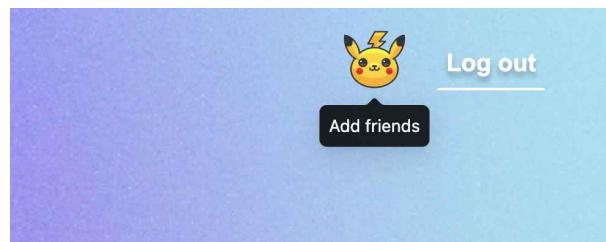
- Namely, we use `echarts` to display the `Radar chart` and use `contribution-heatmap` to display the `Heatmap`.



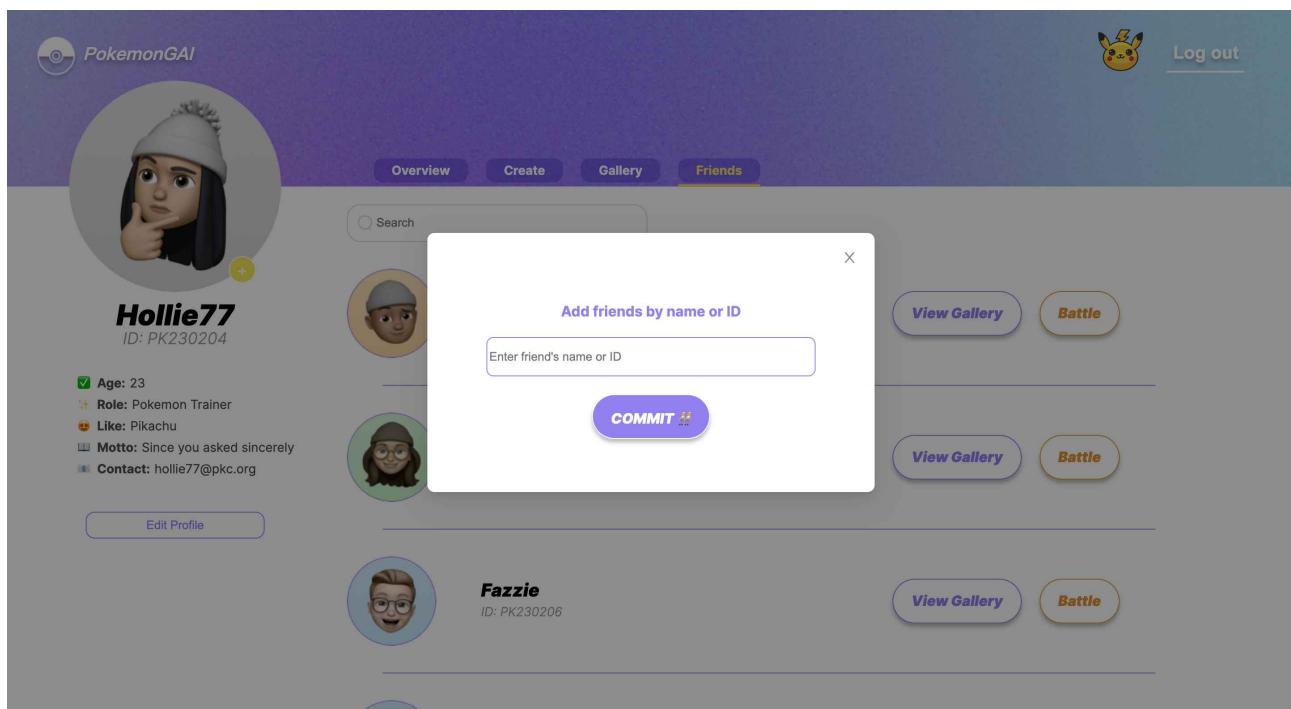
- **Others**

- Besides the above functions, we also have a complete system of ADD, DELETE, UPDATE, and VIEW functions.

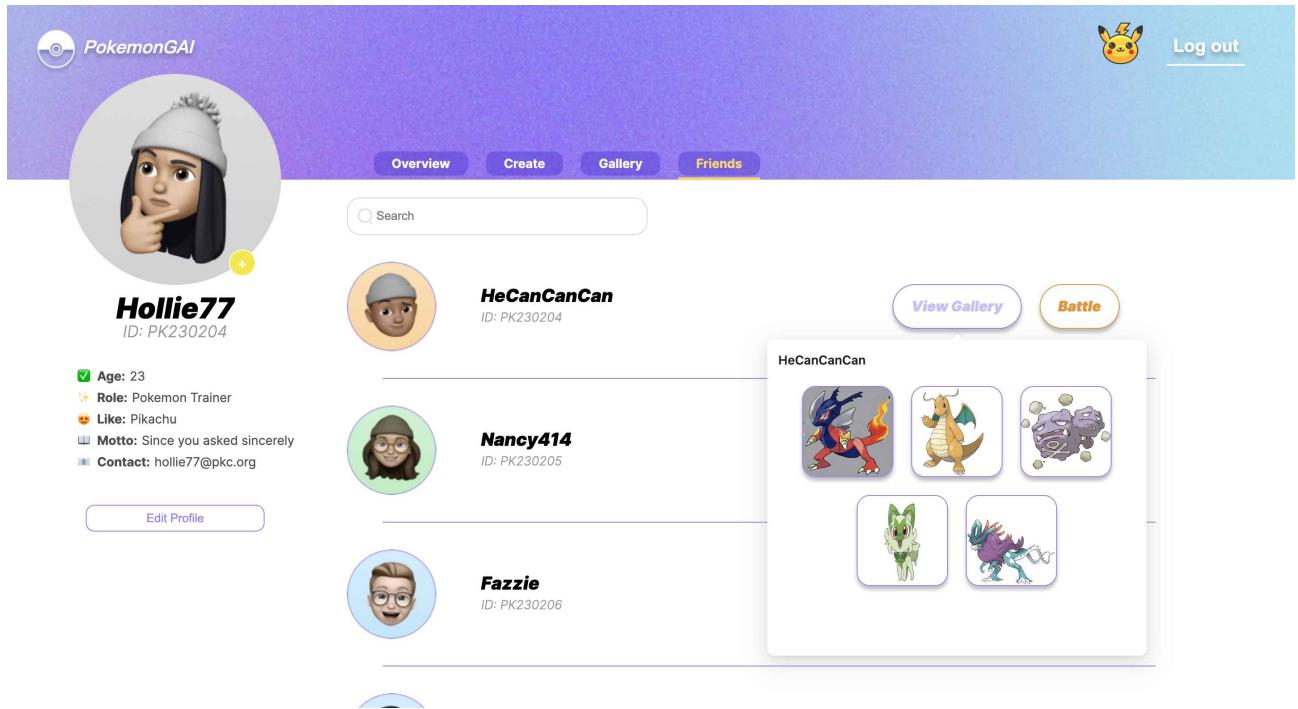
- Add a friend



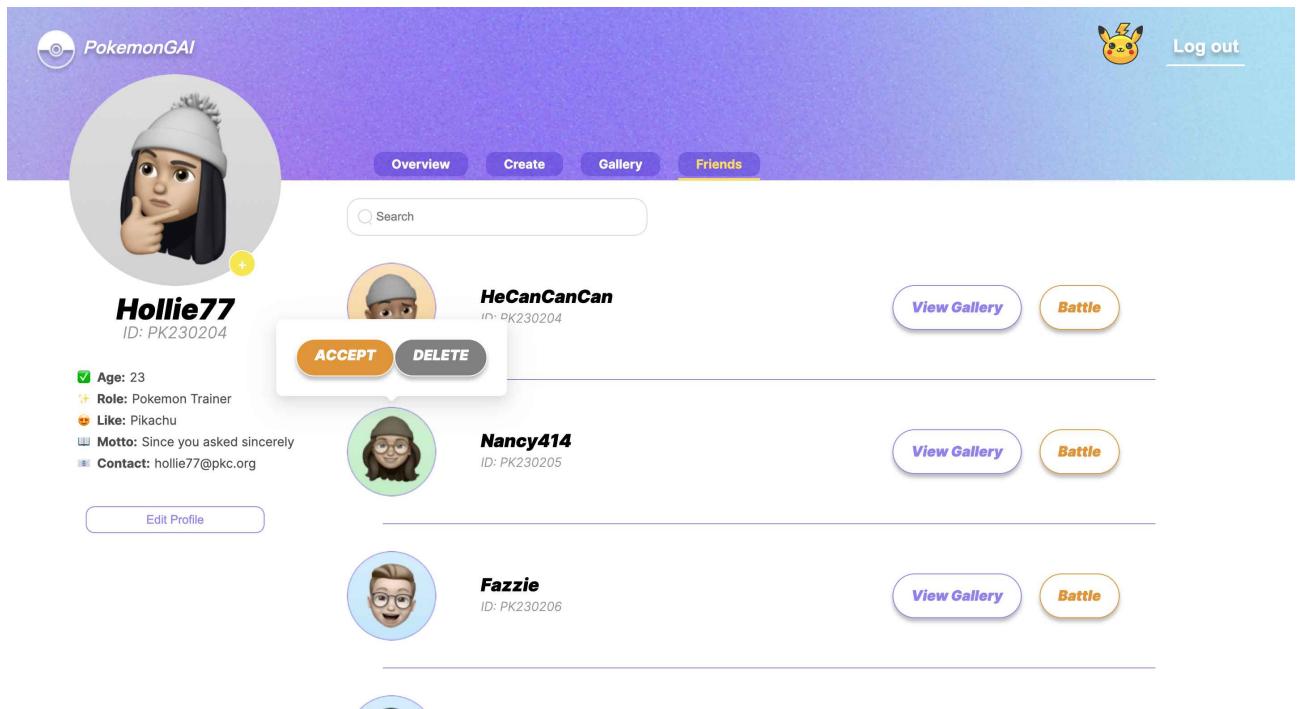
By clicking the right-top icon, we can open a **Modal** to add a friend.



- View friend's gallery



■ Delete friend



3.4 AI

In the AI section, we have implemented the following two core functions, Pokemon image generation and a chatbot for users to chat with Pokemons

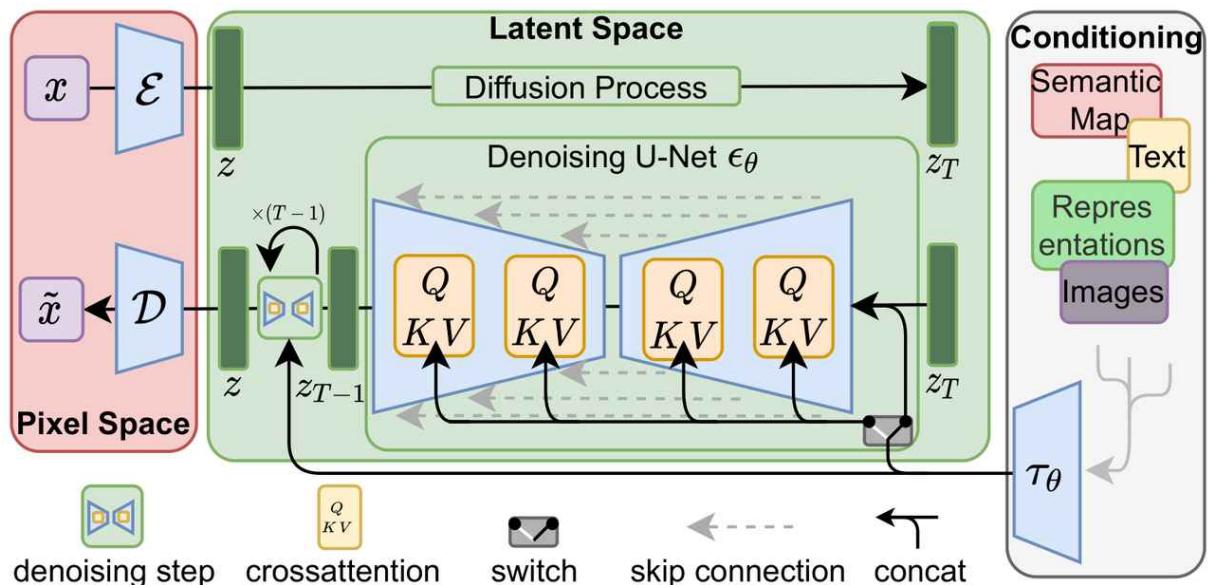
3.4.1 Pokemon image generation

3.4.1.1 Stable diffusion

The idea of the Diffusion model (Diffusion model) was first proposed in the 2015 paper Deep Unsupervised Learning using Nonequilibrium Thermodynamics, and the 2020 paper Denoising Diffusion Probabilistic Models (DDPM) took it to a new level, after which the diffusion model-based DALL-E 2, Imagen, and Stable Diffusion achieved far better results on generative tasks than traditional generative models such as Generative Adversarial Networks (GAN), Variational Differential Autoencoders (VAE), and Autoregressive Models (AR). effects.

Stable Diffusion is a text-to-image latent diffusion model created by the researchers and engineers from CompVis, Stability AI, and LAION. With the open-source stable diffusion model, we fine-tune it to generate the desired model.

The diffusion model consists of two processes: the forward diffusion process, which gradually adds Gaussian noise to an image until it becomes random noise, and the reverse generation process, which is the noise removal process, which gradually removes noise from a random noise using multiple U-Net until an image is generated, which is also part of the training of the diffusion model.



3.4.1.2 Fine-tuning

DreamBooth is a method to personalize text2image models like stable diffusion given just a few(3~5) images of a subject.

Fine-tuning is the common practice of taking a model which has been trained on a wide and diverse dataset, and then training it a bit more on the dataset you are specifically interested in. This is common practice on deep learning and has been shown to be tremendously effective in all manner of models, from standard image classification networks to GANs. In this example we'll show how to fine tune Stable Diffusion on a Pokémon dataset to create a text to image model which makes custom Pokémon inspired images based on any text prompt.

With the datasets offered by [lambdalabs](#), we fine-tone our own pokemon models, here is some example.



3.4.1.3 Diffusers

Diffusers is the go-to library for state-of-the-art pre-trained diffusion models for generating images, audio, and even 3D structures of molecules. Whether you’re looking for a simple inference solution or want to train your own diffusion model, Diffusers is a modular toolbox that supports both. This library is designed with a focus on usability over performance, simplicity over ease, and customizability over abstractions.

With the diffusers library, we quickly trained our Pokémon diffusion model and built our pipeline,

```
1 model_id = "Fazzie/PokemonGAI"
2
3 pipe = StableDiffusionPipeline.from_pretrained(model_id, torch_dtype=torch.float
4
5 prompt = "fire greninja, wings, (fire claws), smoke, cityscape"
6 negative_prompt = 'pixelated, distorted'
7
8 image = pipe(prompt=prompt, negative_prompt=negative_prompt).images[0]
```

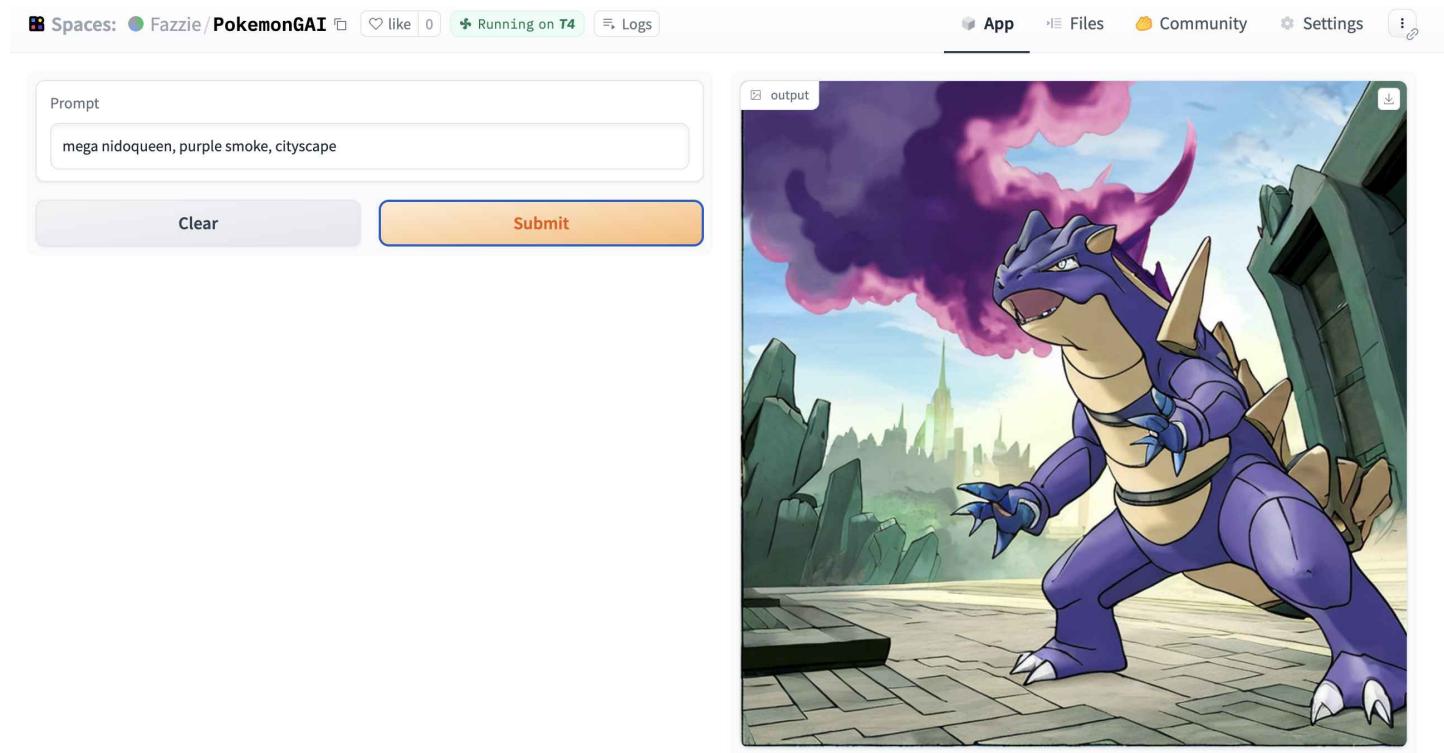
3.4.1.4 Huggingface space & gradio

Traditional front- and back-end web applications usually do not have AI functionality as an AI native application. We need to have an AI service, and AI model inference is usually huge overhead and needs to be supported by hardware.

Huggingface Space is a simple way to host ML demo apps on the Hub. They allow you to build your ML portfolio, showcase your projects at conferences or with stakeholders, and work collaboratively with other people in the ML ecosystem.

Meanwhile, Gradio is the fastest way to demo your machine learning model with a friendly web interface so that anyone can use it. So we use Huggingface space and gradio to build our AI text-to-image servering. You can find our space here:

<https://huggingface.co/spaces/Fazzie/PokemonGAI>. Huggingface space can get the request from the client and return the image generated by models. Here is the space Interface.



3.4.2 Chat with your pokemon

3.4.2.1 ChatGPT

ChatGPT is an artificial intelligence language model that uses deep learning techniques to generate human-like responses to natural language inputs. It is based on the GPT-3.5 architecture and has been trained on vast amounts of text data, allowing it to understand a wide range of topics and respond in a conversational manner.

With the API key offered by open AI, we can build our chatbot server faster.

3.4.2.2 Langchain

Large language models (LLMs) are emerging as a transformative technology, enabling developers to build applications that they previously could not. But using these LLMs in

isolation is often not enough to create a truly powerful app - the real power comes when you are able to combine them with other sources of computation or knowledge.

LangChain is aimed at assisting in the development of those types of applications.

By default, Chains and Agents are stateless, meaning that they treat each incoming query independently (as are the underlying LLMs and chat models). In some applications (chatbots being a GREAT example) it is highly important to remember previous interactions, both in the short-term but also at a long-term level. The concept of “Memory” exists to do exactly that.

LangChain provides memory components in two forms. First, LangChain provides helper utilities for managing and manipulating previous chat messages. These are designed to be modular and useful regardless of how they are used. Secondly, LangChain provides easy ways to incorporate these utilities into chains. Here is part of the code we used to build chatbot

```
1 from langchain import OpenAI, LLMChain
2 from langchain.memory import ConversationBufferWindowMemory
3 chatgpt_chain = LLMChain(
4     llm=OpenAI(temperature=0),
5     prompt=prompt,
6     verbose=True,
7     memory=ConversationBufferWindowMemory(k=2),
8 )
```

The new way of programming models is through prompts. A “prompt” refers to the input to the model. This input is rarely hard coded, but rather is often constructed from multiple components. A PromptTemplate is responsible for the construction of this input. LangChain provides several classes and functions to make constructing and working with prompts easy.

We have defined a template that allows our chatbot to play Pokémon and talk to the user. Here is the prompt template

```
1 template = """You are having a conversation with a magical Pokémon character who
2     This Pokémon has a unique personality and knowledge, and is willing to answe
3     Please note that its responses may not be entirely accurate, but it will do
4     please call me master.
5 {history}
6 Human: {human_input}
7 Pokemon:"""
```

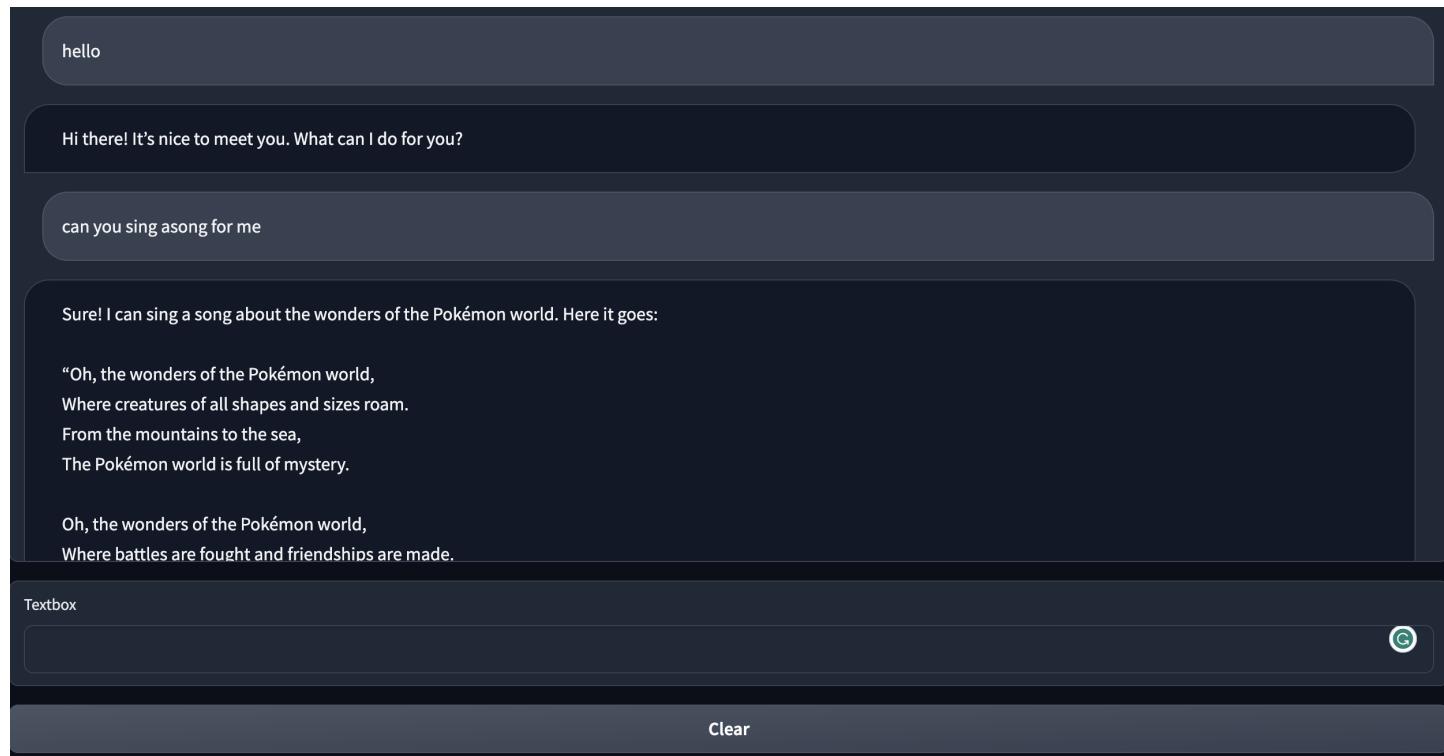
3.4.2.3 Build the chatbot severing

We build the chatbot severing with Openai API and langchain, and we also use gradio to build the application. It will run on <http://127.0.0.1:7860/>, and it can be displayed by the frontend by code

```
1 <iframe  
2     src="http://127.0.0.1:7860/"  
3     frameborder="0"  
4     width="850"  
5     height="450"  
6 ></iframe>
```

Here is some example of the chatbot

- Sing a song



- Make lightning and fire

can you make a lightning?

Unfortunately, I cannot make lightning. However, I can use my electric-type moves to create a powerful electric shock!

can you make a fire?

Unfortunately, I cannot make fire. However, I can use my fire-type moves to create a powerful flame!

what can you do?

I can use my various moves to battle other Pokémon, as well as use my special abilities to help my trainer in various ways. I can also use my knowledge to answer questions about the Pokémon world.

Textbox

Clear

4. Legal/Other Aspects

Use of open source code in the project as well as the possibility of open sourcing the project itself need to be explored. Ideas for protecting the project from being copied will be evaluated.

4.1 Legal

- 1. Intellectual Property:** Pokemon GAI should ensure that it respects the intellectual property rights of the original Pokemon creators (Nintendo, Game Freak, and The Pokemon Company). Acquiring the necessary licenses or permissions to use the Pokemon brand and related assets is crucial.
- 2. Privacy and Data Protection:** Complying with international privacy laws and regulations, such as GDPR and CCPA, is essential. Pokemon GAI should incorporate robust data protection measures to safeguard user information and ensure transparent data collection and usage practices.

4.2 Use of Open Source Code:

- 1. Integration:** Leveraging open-source code can significantly reduce development time and costs. Pokemon GAI could explore existing open-source projects related to AI, NFTs, and social gaming to accelerate the development process.

2. **Licensing:** When using open-source code, we comply with the specific licenses attached to the projects. This may include attributing the original authors, releasing derivative works under the same license, or other obligations outlined in the license terms.

4.3 Open Sourcing the Project:

1. **Pros:** Open-sourcing Pokemon GAI can foster a community of contributors, help identify bugs and vulnerabilities, and promote transparency in the project. It can also increase trust among users and encourage adoption.
2. **Cons:** Open-sourcing may expose the project to potential copycats and make it challenging to protect the unique aspects of the platform. Additionally, it might require increased effort to manage community contributions and maintain quality control.

4.4 Protecting the Project from Being Copied:

1. **Patents:** Applying for patents on unique aspects of Pokemon GAI technology, such as specific AI algorithms, NFT implementation, or social gaming mechanics, can offer legal protection against copycats.
2. **Trade Secrets:** Keeping specific algorithms, technologies, or strategies as trade secrets can prevent competitors from replicating the platform's unique features.
3. **Branding and Community:** Building a strong brand and a loyal user base can make it difficult for copycats to compete. Engaging with the community, providing excellent customer support, and continuously updating and improving the platform can create a competitive advantage that is hard to replicate.

In conclusion, the legal aspects, open-source integration, and protection strategies are carefully considered and implemented to ensure Pokemon GAI's success and sustainability. Balancing openness with adequate protection will help the project thrive in a competitive market while respecting the rights of others and protecting its unique innovations.

Competition Analysis

To evaluate the competitive landscape for Pokemon GAI, we can examine similar products in the gaming and collectibles markets that incorporate AI, NFTs, or social gaming features. Key competitors include:

- 1. Axie Infinity:** A blockchain-based game where users collect, breed, and battle fantasy creatures called Axies. Players can earn cryptocurrency through gameplay, and Axies are represented as NFTs. While Axie Infinity focuses on a different creature universe, it competes with Pokemon GAI in the NFT-based gaming and collectibles market.
- 2. CryptoKitties:** As one of the first blockchain games to gain mainstream attention, CryptoKitties allows users to collect, breed, and trade virtual cats represented as NFTs. Although it lacks the battling and social gaming features found in Pokemon GAI, CryptoKitties competes in the NFT-based collectibles market.
- 3. Sorare:** A global fantasy football platform based on blockchain technology, Sorare enables users to collect, trade, and manage a team of digital player cards represented as NFTs. While it targets sports enthusiasts, Sorare competes with Pokemon GAI in the broader NFT-based gaming and collectibles markets.
- 4. Artificial Intelligence-based games:** Various AI-driven games that leverage procedural generation and machine learning to create unique gaming experiences could be potential competitors. These games may not focus on NFTs or Pokemon, but they compete for the attention of users seeking novel and engaging gaming experiences.
- 5. Traditional Pokemon games:** Mainstream Pokemon games, such as Pokemon Sword and Shield or Pokemon Go, are direct competitors in the Pokemon gaming market. Although they may not incorporate AI, NFTs, or social gaming features to the same extent as Pokemon GAI, their established fan base and brand recognition pose a significant competitive challenge.

In conclusion, Pokemon GAI faces competition from several market segments, including NFT-based games, AI-driven games, and traditional Pokemon games. To differentiate itself and succeed in this competitive landscape, Pokemon GAI must focus on delivering a unique and engaging experience by effectively combining AI, NFTs, and social gaming features. Additionally, building a strong community and continuously improving the platform will help Pokemon GAI stand out among competitors.

Project division

- Ma Qianli
 - AI (Imagen, chatbot)
 - Backend
- Song Tianqi
 - UI design
 - Frontend

- He Can
 - Backend
 - Database

Reference

1. <https://huggingface.co/datasets/lambdalabs/pokemon-blip-captions>
2. Ruiz N, Li Y, Jampani V, Pritch Y, Rubinstein M, Aberman K. Dreambooth: Fine tuning text-to-image diffusion models for subject-driven generation. arXiv preprint arXiv:2208.12242. 2022 Aug 25.
3. <https://huggingface.co/docs/diffusers/index>
4. <https://github.com/hwchase17/langchain>