CipherGPT: An LLM Arcade Game

Roan Morgan, Jonathan Attanasio, Holden Roaten

Brogrammers

Introduction:

CipherGPT is an arcade that leverages large language models. Currently the project includes a game where the player answers trivia while interacting with the LLM. The LLM can only interact with you through the use of emojis for added difficulty.

The motivation for this project stemmed from the curiosity of utilizing LLM not just for getting answers to every question you might have, but to utilize LLMs to entertain a player. In order to approach this project, the group decided on a React web app that uses the power of a local Gemini 2 instance (2B) supplied by LM Studio's REST API.

At the inception of this project, the team agreed on making a game that worked as a game of telephone with two players and the LLM as the middleman encoding the message in various manners. This interaction would have been to have player A communicate the bomb defusal instructions to player B. About a month into the project, the team decided we needed to pivot in order to get the project done on time. This is when we decided on the trivia gameplay.

With our pivoted project goals in mind, the results of our app were more than satisfactory and even included variation in gameplay with the addition of multiple trivia categories. With this being said, we did complete our project goals of making a fun and unique game that included player-llm interaction complete with score keeping and a visually appealing user interface.

Customer Value:

The primary customer is going to be anybody that is familiar with LLMs or anybody curious as to how they can be used for more fun things than cheating in academia. The customer wants an easy to use interface with quick response times no matter the application. This does not inherently solve any problem for our users. Our solution will deliver a fun way to interact with generative AI in a game-like environment with leaderboards for the trivia game. Everyone that we've voiced the idea to seems to love it. Similar to the New York Times games they are fairly simple, the difference is that all of our's are based around generative AI. Our customer-centric measures of success would be that our leaderboard fills up

because people keep coming back to play over and over, that would quantify success in a measurable way.

Technology:

- **Framework (Next.js):** A React-based framework that enables server-side rendering, static site generation, and API routes for scalable web applications.
- **Programming Language (TypeScript):** A statically typed superset of JavaScript that enhances code maintainability and developer productivity.
- **Database Layout (PostgreSQL):** A modern database for storing the leaderboard information
- **Generative AI (Gemini 2):** An advanced AI model for text generation, coding assistance, and content creation.
- **LLM API (LM Studio):** A local inference platform for running and fine-tuning large language models efficiently.
- **Component Library (ShadCN UI):** A collection of customizable and accessible UI components built on Radix and Tailwind CSS.
- **Styling (Tailwind CSS):** A utility-first CSS framework for rapidly building modern and responsive user interfaces.

Team:

Roan Morgan:

Software Engineer

- Programmed base game functionality
- Established connection between the web app and the LM Studio REST API
- Established UI/UX for the trivia game page

Lead Product Manager

- Made sure the contributions of other developers aligned with project goals
- Guided project design

Jonathan Attanasio:

Frontend Developer

- Designed UI and layouts for the application

Software Engineer

- Chose the tech stack (besides LLM tools)
- Programmed the “three strike” logic

Holden Roaten:

Software Engineer

- Managed user to language model interaction
    - Prevented user override of game functionality
    - Filtered Language Model output to fit parameters of the game
- Set game parameters
    - Established groups of categories for the Language model to describe to the player
    - Created Logic for random category/item selection

Project Management:

Many of the original goals of the project were not met as a result of two primary causes- The initial game concept as a whole was greater than could be accomplished in the given time and Language Model output is unpredictable and difficult to manageAfter scaling back the concept of the game and focusing on one particular facet of the original, our goals related to creating a single game type were mostly met.The main shortcoming of the project was the language model's lack of output that consistently fit the game parameters given.

Reflection:

Through consistent team communication and collaboration, we were able to implement a version of CipherGPT that functioned almost exactly as intended. With thorough planning and the readjustment of goals where needed the development of the project moved very smoothly from concept to prototype. The delegation of tasks as well as an organized code base allowed our team to work independently where necessary while also allowing our completed portions to function together when testing.

Ultimately the project was a success, but one that could be improved upon significantly. Future iterations of the project may find more success after more thorough research and discovery into Language Model control and management.