Documentation for LLM Model Development and Optimization

Overview

This project involved developing and optimizing a language model using Google Gemini, hosted on Streamlit, to generate frontend code (HTML, CSS, JS) based on user prompts. The goal was to create a tool that could translate design requirements into functional code, simplifying the process of web development for users. At the end there are screenshots attached, along with instruction on how to use the application.

Initial Model Development

Objective:

To develop a basic LLM model using Google Gemini, hosted on Streamlit, for the purpose of generating frontend code.

Approach:

- **Model Selection:** Chose Google Gemini for its advanced capabilities in natural language understanding and generation.
- **Deployment:** Utilized Streamlit for hosting the model due to its ease of use and integration with Python-based applications.
- **Functionality:** Enabled basic functionality to generate frontend code (HTML, CSS, JS) based on user-provided prompts.

Prompt Optimization

Objective:

To improve the accuracy and relevance of the generated frontend code through optimized prompting techniques.

Approach:

- **Initial Prompts:** Started with basic prompts to gauge the model's ability to generate frontend code.
- **Optimization:** Iteratively refined the prompts to better align with design requirements and user expectations. This included adjusting prompt structure, incorporating example-based guidance, and enhancing prompt specificity.

Addition and Removal of HTML Download Functionality

Objective:

To provide users with an easy way to download generated frontend code.

Approach:

- **Initial Implementation:** Added a feature to download the generated code as an HTML file for user convenience.
- **Issue Encountered:** Experienced problems with the downloaded HTML files, rendering them unusable.
- **Resolution:** Removed the download functionality after identifying that it did not meet user needs and caused operational issues.

Final Prompt Optimization

Objective:

To further enhance the functionality and performance of the LLM model by optimizing the prompts based on user feedback and performance analysis.

Approach:

- **Analysis:** Reviewed model performance and user feedback to identify areas for prompt improvement.
- **Refinement:** Implemented additional refinements to the prompts to improve code generation accuracy, usability, and relevance.

Setup Instructions:

1. Clone the Repository:

- Load the code from the GitHub repository using the following command:

  ‘’’ git clone [repository link] ‘’’

2. Install Dependencies:

- Navigate to the project directory and install the necessary Python packages:

  ‘’’ pip install streamlit google-generativeai ‘’’

3. Run the Application:

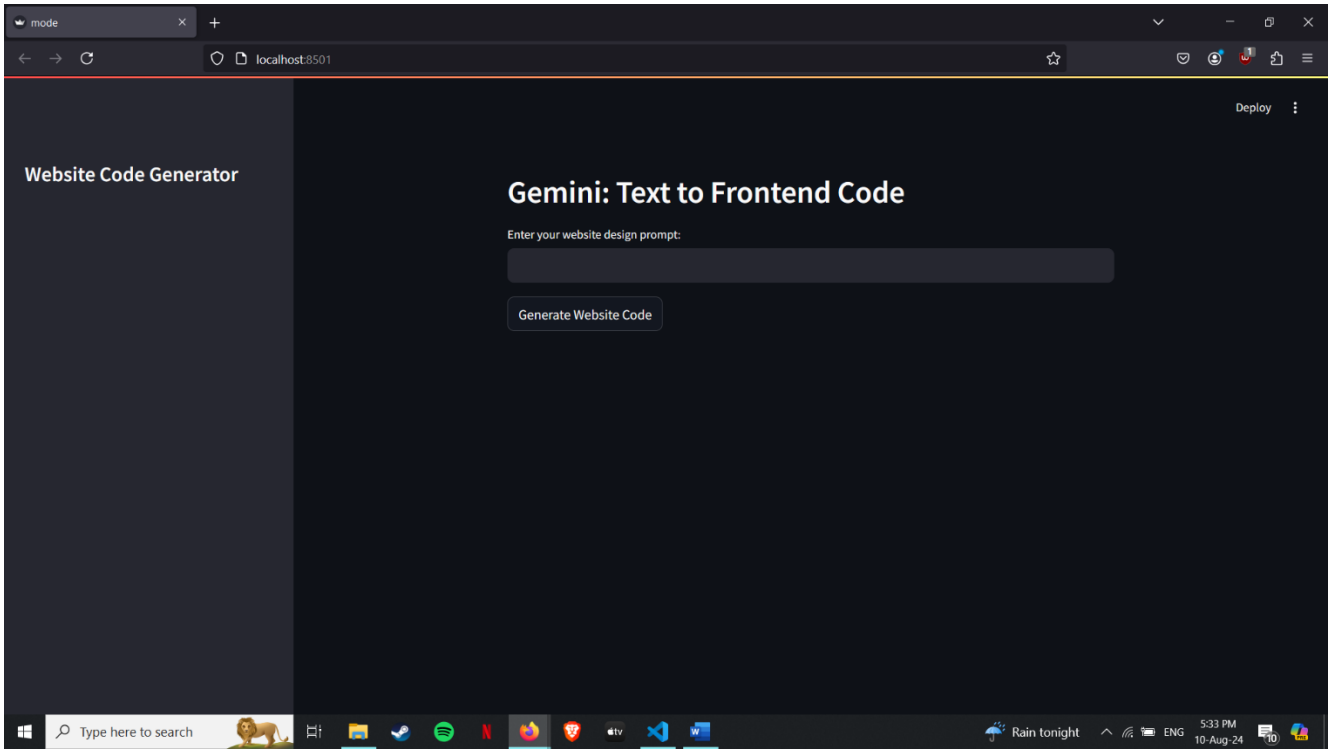
- Launch the Streamlit application by running the following command:

  ‘’’ streamlit run mode.py ‘’’

Using the Application:

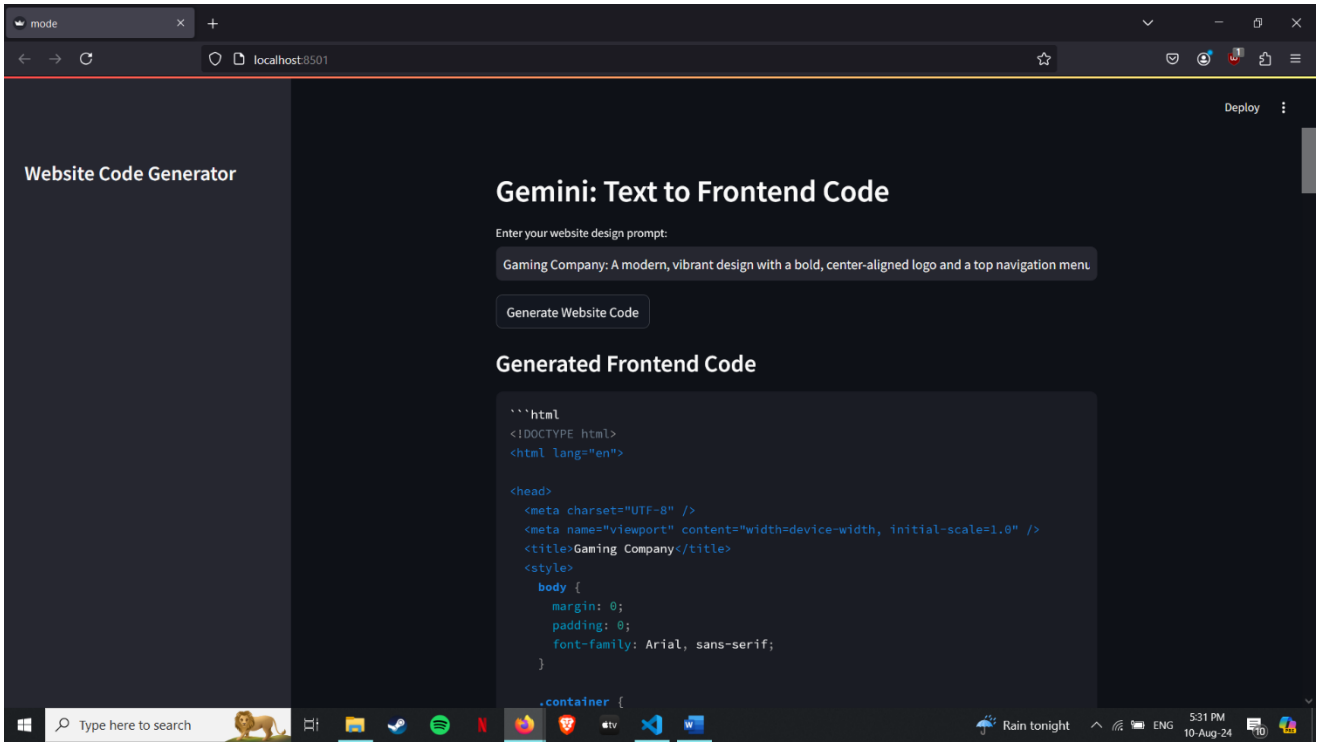
1. **Entering a Prompt:**

   - Open the Streamlit application. In the input field, enter a prompt describing the webpage you want to generate.
   - Click on the "Generate Code" button.



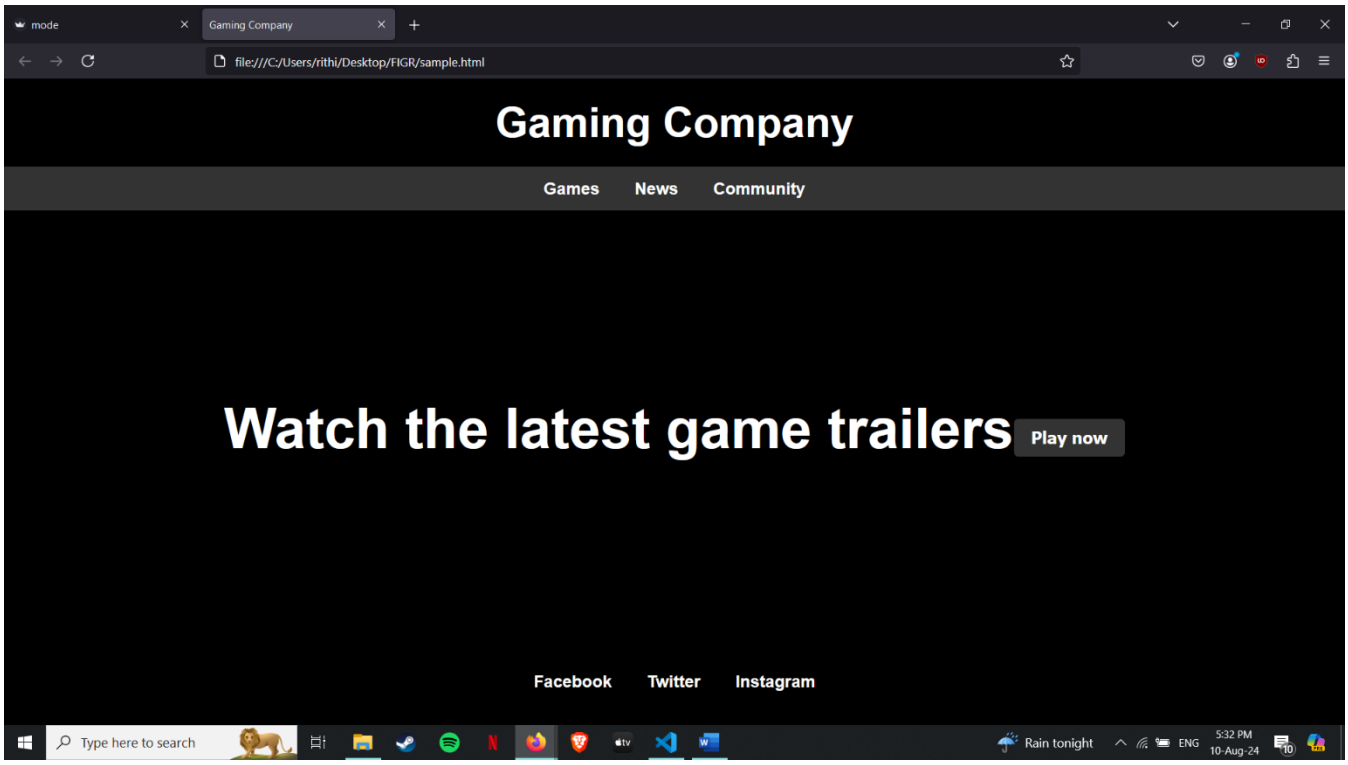
2. **Copying the Generated Code:**

- After running the prompt, the application will display the generated code.
- Copy the code directly from the application and paste it into an HTML file.



3. **Viewing the Webpage:**

   - Open the HTML file in a web browser to view the generated webpage.



Conclusion

The project involved iterative development and optimization of an LLM model using Google Gemini, with a focus on generating frontend code. Key activities included initial model development, prompt optimization, and adjustments to functionality. The final iteration of the project achieved improved prompt efficacy and user satisfaction, resulting in a more reliable and effective tool for generating frontend code.

This documentation, along with the attached screenshots, provides a comprehensive guide to the development process, the rationale behind key decisions, and instructions for using the application.