DASI GPT Technical Documentation

Table of Contents

Introduction

Welcome to the technical documentation for DASI GPT. This project is developed by Simen Evenrud Blien and Daniel Johan Sørby. DASI GPT is a chatbot based on openAI.

Project Overview

DASI GPT is designed to assist you with your daily tasks. It aims to make it easy for you to ask for help, and make it easy to navigate.

Installation

To install DASI GPT, follow these steps:

1. Clone the repository:

```
git clone https://github.com/DanielOgSimen/DASI.git
```

2. Navigate to the project directory:

```
cd dasi-gpt
```

3. Install the dependencies:

```
npm install
```

Usage

To use DASI GPT, follow these steps:

1. Start the application:

```
npm start
```

2. Open your browser and navigate to `http://localhost:3000`.

Architecture

DASI GPT is built using the following technologies:

- **Svelte Kit**: A modern framework for building web applications. Svelte Kit is used for the frontend of DASI GPT, providing a reactive and efficient user interface.
- **TypeScript**: A statically typed superset of JavaScript that enhances code quality and maintainability. TypeScript is used throughout the project to ensure type safety and improve developer productivity.
- **OpenAI**: The core of DASI GPT's functionality is powered by OpenAI's GPT models. These models are used to generate natural language responses and perform various AI-driven tasks.
- **MySQL**: A relational database management system used to store and manage data. MySQL is used for persistent storage of user data, application settings, and other relevant information.
- **Linux**: The operating system on which the application is deployed. Linux provides a stable and secure environment for running the application.
- **Git**: A version control system used to manage the project's source code. Git enables collaboration, version tracking, and efficient code management.
- **Markdown**: A lightweight markup language used for writing documentation. Markdown is used to create the project's technical documentation and other text-based content.
- **JavaScript**: The primary programming language used in the project. JavaScript is used for both frontend and backend development, enabling dynamic and interactive functionality.

Components

1. **Frontend (Svelte Kit)**
    - **UI Components**: Reusable and modular components built with Svelte Kit to create the user interface. These components handle user interactions and display data.
    - **Routing**: Svelte Kit's built-in routing system is used to manage navigation within the application. It provides a seamless user experience by enabling client-side routing.
    - **State Management**: Svelte's reactive stores are used to manage the application's state. This ensures that the UI is always in sync with the underlying data.
2. **Backend (TypeScript & Node.js)**
    - **API Endpoints**: The backend exposes a set of RESTful API endpoints that the frontend interacts with. These endpoints handle requests, process data, and return responses.
    - **Business Logic**: The core functionality of the application is implemented in the backend. This includes processing user inputs, interacting with the OpenAI API, and managing data.
    - **Database Integration**: The backend communicates with the MySQL database to store and retrieve data. This includes user information, application settings, and other relevant data.
3. **OpenAI Integration**

- **GPT Models**: The application integrates with OpenAI's GPT models to generate natural language responses. This involves sending requests to the OpenAI API and processing the responses.
- **AI-driven Features**: Various features of the application are powered by AI, such as text generation, language translation, and more.

4. **Database (MySQL)**
    - **Schema Design**: The database schema is designed to efficiently store and manage data. This includes tables for users, settings, logs, and other entities.
    - **Data Access**: The backend uses an ORM (Object-Relational Mapping) library to interact with the MySQL database. This provides a high-level abstraction for database operations.
5. **Deployment (Linux)**
    - **Server Configuration**: The application is deployed on a Linux server. This includes configuring the web server, database server, and other necessary services.
    - **Continuous Integration/Continuous Deployment (CI/CD)**: Git and other CI/CD tools are used to automate the deployment process. This ensures that new changes are tested and deployed efficiently.

API Reference

Endpoint 1: Generate Chat Title

- **URL**: `/api/title`
- **Method**: `POST`
- **Description**: Generates a chat title based on the provided message using OpenAI's GPT model.
- **Parameters**:
    - **message** (string, required): The input message to generate a title from.
- **Request Body**:

```
{
  "message": "Your input message here"
}
```

- **Response**:

```
  {
  "title": "Generated title"
  }
```

Endpoint 2

- **URL**: `/api/`
- **Method**: `POST`

- **Description**: Generates a chat completion based on the provided messages using OpenAI's GPT model.
- **Parameters**:
  - **message** (array, required): An array of message objects to generate a completion from.
  - **model** (string, optional): The model to use for generating the completion. Default is "gpt-3.5-turbo".
- **Request Body**:

```
{
"messages": [
    { "role": "user", "content": "Hello, how are you?" },
    { "role": "assistant", "content": "I'm good, thank you!" }
],
"model": "gpt-3.5-turbo"
}
```

- **Response**:

```
{
"message": "I'm here to help you with anything you need."
}
```

Error Handling:

- **Describtion**: Both endpoints handle errors by returning a JSON response with an error message and a status code of 500.
- **Error Response**:

```
{
"error": "Internal Server Error"
}
```

Contributing

We welcome contributions to DASI GPT. To contribute, follow these steps:

1. Fork the repository.
2. Create a new branch:

```
git checkout -b feature-branch
```

3. Make your changes and commit them, please make a describtion of your changes:

```
git commit -m "Description of your changes"
```

4. Push to the branch:

```
git push origin feature-branch
```

5. Create a pull request.

License

DASI GPT is licensed under the. See the LICENSE file for more information.

Contact

For questions or feedback, please contact us at contact@dasigpt.com.