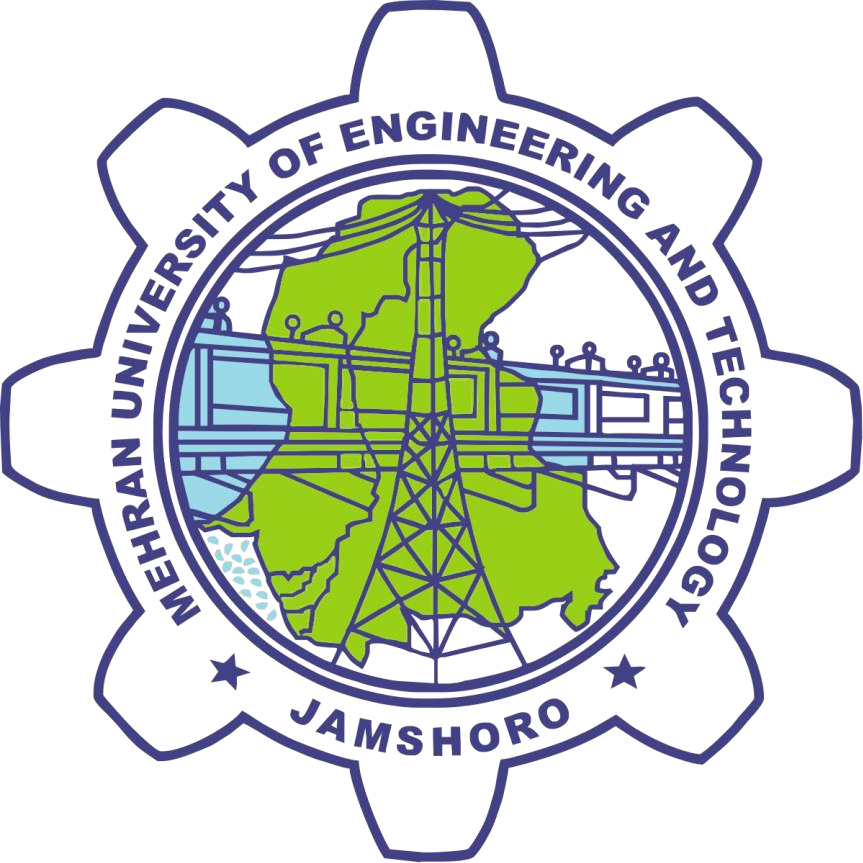
**Mehran University of Engineering and Technology, Jamshoro**

**Department of Computer System Engineering(4th semester, 2nd year)**



PROJECT REPORT: Terminal GPT

# Roll no: 22CS064 || 22CS064

**Subject: Operating Systems (OS) CS-261 Submitted to: Dr. Bushra Naz**

**Department of Computer Systems Engineering**

|  |  |  |  |
| --- | --- | --- | --- |
| **Course: Operating Systems (CS-261)** | | | |
| **Instructor** | Dr. Bushra Naz | Assignment Type | Complex Engineering Problem |
| **Semester** | 4th | Year | 2nd |
| **Submission Deadline** | 5-11-2024 | Assessment Score | 05 |

Semester project is designed in a way to able students to solve the complex engineering problem using the Operating systems. Following characteristics of complex engineering problem are targeted in this semester project of OS.

|  |  |  |
| --- | --- | --- |
| **Complex Engineering Problem – Characteristics** | | |
| 1 | Depth of knowledge Required |  |
| 2 | Range of Conflicting Requirements |  |
| 3 | Depth of Analysis Required |  |
| 4 | Infrequently Encountered Issues Involved |  |
| 5 | Beyond codes/standards of practice |  |
| 6 | Diverse groups of stakeholders with widely varying needs involved |  |
| 7 | Interdependence (high level problems including many components parts/sub-problems) |  |
| 8 | Have significant consequences in a range of contexts |  |
| 9 | Judgement (Require judgement in decision making) |  |

**Project Objectives:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Rubrics | | | | CEP  characteristics | Marks distribution |
|  | Unacceptable 2 | Acceptable 8 | Proficient 10 |
| R1: Idea/Initial Study |  |  |  | WP2 | 20% |
| R2: Project Proposal |  |  |  | WP1, WP3 | 20% |
| R3: Project Progress |  |  |  | WP3, WP2 | 20% |
| R4: Final Report |  |  |  | WP3, WP1 | 40% |

***Terminal GPT (tgpt)***

**Overview**

The *Terminal GPT (tgpt)* project is a command-line interface application that integrates GPT-based AI capabilities into terminal workflows, allowing users to run queries, generate code, execute shell commands, and perform other interactive operations without the need for external API keys.

**Project Structure**

The tgpt project is organized as follows:

22CS078/

│

├── client/

│ └── client.go

├── providers/

│ ├── openai/

│ │ └── openai.go

│ └── providers.go

├── structs/

│ └── structs.go

├── utils/

│ └── utils.go

├── workflows/

│ └── main.yml

├── build.sh

├── install

├── main.go

└── tgpt.json

**Key Components:**

* **client.go**: Implements the NewClient() function, sets up HTTP client configurations, handles proxy settings, and manages HTTP communication with secure connections.
* **openai.go**: Defines functions for creating requests to the OpenAI API, sets headers, and handles responses.
* **providers.go**: Manages provider-specific logic; supports OpenAI.
* **structs.go**: Contains structure definitions, including parameters for API requests and response handling.
* **utils.go**: Implements utility functions like RandomString() for generating random strings and LogToFile() for logging.
* **main.go**: The main entry point of the application, handling user interaction, command-line flags, input parsing, and program execution.
* **main.yml**: Workflow script for building the project and creating releases on GitHub.
* **build.sh**: Script for compiling the application for different architectures.
* **install**: Bash script for installation on Ubuntu systems.
* **tgpt.json**: Metadata file containing version, licensing, and architecture-specific download links.

**Installation and Usage**

**Installation Steps:**

1. **Download and Install**: Run the following command to download and install tgpt:

**curl -sSL https://raw.githubusercontent.com/itsmrzok/64CS078/main/install | bash -s /usr/local/bin**

This installs tgpt to /usr/local/bin (modifiable). Ensure the directory is in your PATH.

1. **Verify Installation**:

**tgpt -v**

**Usage:**

After installation, tgpt can be run with various flags. Some examples:

* Run a query:

**tgpt "What is the Internet?"**

* Execute shell commands interactively:

**tgpt -s "List all files"**

**Uninstallation:**

To remove tgpt, execute:

**sudo rm $(which tgpt)**

**Implementation Details**

**Core Functionalities:**

1. **Client Setup** (client.go):
   * Configures the HttpClient with a cookie jar and TLS settings.
   * Supports environment-variable-driven proxy configuration and fallback to configuration files if not set.
2. **API Request Handling** (openai.go):
   * Constructs and sends POST requests to OpenAI's API endpoint.
   * Extracts main response text by parsing JSON response payloads.
3. **Provider Logic** (providers.go):
   * Currently supports OpenAI. Additional providers can be integrated following a similar pattern.
4. **Utility Functions** (utils.go):
   * Logging to file and generating random strings for unique thread IDs in interactive sessions.

**Code Structure:**

* **Packages and Imports**: The project relies on external libraries like github.com/bogdanfinn/tls-client for HTTP client customization, and Go's standard libraries for file handling and string operations.
* **Flags**: Command-line arguments are processed using flag for customizable behavior such as enabling shell mode (-s), generating code (-c), and setting API parameters.

**Workflow:**

* **main.yml**: Automates the build process using GitHub Actions, compiling tgpt for Linux and uploading artifacts for release.
* **build.sh**: Cross-compiles tgpt for different architectures.

**Running the Project:**

To execute tgpt with a specific functionality:

**tgpt -c "Generate Python code for a addition of number"**

**Extensions and Implementation Essentials:**

* **File Extensions**:
  + .go: Go source files.
  + .sh: install: Shell scripts for building and installation.
  + .json: Configuration and metadata.
  + .yml: Workflow scripts for CI/CD.
* **Dependencies**:
  + Go version specified in main.yml (e.g., ^1.22.1).

**Conclusion**

The tgpt project by **22CS064 and 22CS078**, under the supervision of **Teacher Bushra Naaz**, presents a robust and extendable terminal-based interface for integrating GPT capabilities, tailored for developers and command-line users. The organized codebase, use of best practices in structuring, and inclusion of flexible installation and interactive modes make it a versatile tool for practical applications.