|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Student name: | Student 1: Karen Ferreira Magalhaes  Student 2: Thales Campos  Student 3: Vitor Freitas | | | | | |
| Student number: | Student 1: 3146094  Student 2: 3151261  Student 3: 3152612 | | | | | |
| Faculty: | Computing Science | | | | | |
| Course: | BSCH/BSCO/EXCH | | | Stage/year: | 2 | |
| Subject: | Software Development 2 | | | | | |
| Study Mode: | Full time | Icon  Description automatically generated |  | Part-time |  |  |
| Lecturer Name: | Haseeb Younis/ Muhammad Shoaib | | | | | |
| Assignment Title: | Project Final Documentation | | | | | |
| Date due: | 27/04/2025 | | |  | | |
| Date submitted: | 23/03/2025 | | |  | | |
| **Plagiarism disclaimer:**  *I understand that plagiarism is a serious offence and have read and understood the college policy on plagiarism. I also understand that I may receive a mark of zero if I have not identified and properly attributed sources which have been used, referred to, or have in any way influenced the preparation of this assignment, or if I have knowingly allowed others to plagiarise my work in this way.*  *I hereby certify that this assignment is my own work, based on my personal study and/or research, and that I have acknowledged all material and sources used in its preparation. I also certify that the assignment has not previously been submitted for assessment and that I have not copied in part or whole or otherwise plagiarised the work of anyone else, including other students.*  **Signed: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Date: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** | | | | | | |
| **Please note:** Students **MUST** retain a hard / soft copy of **ALL** assignments as well as a receipt issued and signed by a member of Faculty as proof of submission. | | | | | | |

A picture containing text, outdoor, building, white

Description automatically generated

Software Development 2

BSCH-SD2

Chatbot Project

* **ChatGPTClient Class**

**Overview:**

The ChatGPTClient class is a Spring @Component that handles communication with the OpenAI ChatGPT API. It sends a user-provided conversation context to the API and receives a generated response from the AI model. The class uses **Spring's @Value annotation** to inject the API key and API URL from the application's configuration file (e.g., application.properties or application.yml).

**Dependencies:**

* **Spring Framework**:
  + @Component for defining a Spring-managed bean.
  + @Value for injecting property values.
* **Google Gson**:
  + Used for building and parsing JSON payloads.
* **Java Standard Libraries**:
  + HttpURLConnection for HTTP communications.
  + InputStreamReader, BufferedReader, and OutputStream for I/O operations.
  + URL and networking tools for setting up API requests.
  + StandardCharsets for character encoding.

**Attributes:**

* API\_KEY (String): Injected from the property ${openai.api.key}, used for authenticating with the OpenAI API.
* API\_URL (String): Injected from the property ${openai.api.url}, defines the endpoint to which requests are sent.

**Public Methods:**

* **String getChatResponse(String conversationContext)**
  + **Parameters**:
    - conversationContext: A string containing the user's input or context for the conversation.
  + **Returns**:
    - A String containing the response generated by the ChatGPT model.
    - Returns an error message if the request fails.
  + **Process Flow**:
    - Establishes an HTTP POST connection to the OpenAI API using HttpURLConnection.
    - Sets the necessary request headers:
      * Authorization (Bearer token with the API key)
      * Content-Type (application/json)
      * OpenAI-Project (specific project identifier)
    - Builds the request payload:
      * Defines the model as gpt-3.5-turbo.
      * Adds a system message that instructs the assistant to act as Tripper, a helpful travel clothing planner.
      * Adds the user-provided conversation context as a user message.
    - Sends the JSON payload to the API.
    - Reads the response stream and parses it using Gson.
    - Extracts the AI's reply from the first element of the "choices" array in the JSON response.
    - If successful, returns the AI's message; otherwise, returns a default error message.
  + **Error Handling**:
    - Catches any exceptions during the request or parsing process.
    - Returns a generic error message containing the exception details.
* **GoogleMapsService Class**

**Overview:**

The GoogleMapsService class is a Spring @Service that interacts with the **Google Maps Geocoding API**.  
Its main purpose is to validate whether a given place name corresponds to a real geographic location by sending requests to the Geocoding API. The API key is securely injected from the application's configuration using Spring's @Value annotation.

**Dependencies:**

* **Spring Framework**:
  + @Service to declare the class as a Spring-managed service bean.
  + @Value for property injection (Google API key).
* **Google Maps Java Client Library**:
  + GeoApiContext for building the API context.
  + GeocodingApi for accessing geocoding services.
  + GeocodingResult for handling API responses.

**Attributes:**

* context (GeoApiContext):
  + Holds the configuration for connecting to the Google Maps API, including the API key.

**Constructor:**

* **GoogleMapsService(@Value("${google.api.key}") String apiKey)**
  + **Parameters**:
    - apiKey: The Google Maps API key injected from the application configuration.
  + **Behavior**:
    - Initializes a GeoApiContext with the provided API key.
    - Prepares the service to make authorized requests to Google Maps APIs.

**Public Methods:**

* **boolean isValidLocation(String place)**
  + **Parameters**:
    - place: The name of the place (e.g., "Paris", "Times Square", "Mount Everest") to validate.
  + **Returns**:
    - true if the location is recognized by the Google Geocoding API.
    - false if the location is not found or an error occurs.
  + **Process Flow**:
    - Sends a request to the Google Geocoding API using the provided place.
    - If the API returns one or more results, the method considers the place valid.
    - If the API throws an exception (e.g., invalid input, connection issue), the method catches it and prints an error message.
    - Returns false if no results are found or an exception is raised.