**Login\_API:**

**Class:** N/A (Flask application)  
This Flask application provides an API endpoint for handling user login operations.

* **Route:** /api/login
* **Methods:** POST

**Function: login**

* **Purpose:** Validates user login credentials and returns the user’s role and status.
* **Logic:**
  1. Hardcoded user credentials (user\_email and user\_password) are stored for comparison.
  2. Parses the JSON request body to retrieve the usermail and password fields.
  3. Validates that both usermail and password are provided; returns a 400 Bad Request if not.
  4. Compares the provided credentials with the hardcoded values:
     + Returns a 401 Unauthorized if the credentials are invalid.
     + Returns a 200 OK response with a success message if the credentials match.
  5. Handles exceptions and returns a 500 Internal Server Error if any error occurs.
* **Parameters:**
  1. request: Contains the JSON payload with usermail and password.
* **Returns:**
  1. **400 Bad Request:** If either usermail or password is missing.
  2. **401 Unauthorized:** If the credentials are invalid.
  3. **200 OK:** If the credentials are valid, along with user status and role.
  4. **500 Internal Server Error:** If an exception occurs during processing.

**LogWise\_API:**

**Class:** N/A (Flask application)  
This Flask application integrates with Google Gemini API to analyze log data and generate concise descriptions for error messages.

* **Route:** /api/analyze
* **Methods:** POST

**Function: analyze\_log\_data**

* **Purpose:** Analyzes log data provided in the request body and generates concise descriptions for errors using the Google Gemini API.
* **Logic:**
  1. Configures the Gemini API client with the provided API key and model.
  2. Parses the JSON request body to retrieve:
     + file\_content: The raw log data.
     + parsed\_log\_data: A list of log entries, each containing an error message and details.
  3. Validates the presence of file\_content; returns a 400 Bad Request if missing.
  4. Iterates through parsed\_log\_data, and for each log entry:
     + Creates a prompt combining the error message and details.
     + Sends the prompt to the Gemini API for generating a description.
     + Extracts the response text and appends it to a list of descriptions.
  5. Returns the original log data and their corresponding descriptions in the response.
  6. Handles exceptions and returns a 500 Internal Server Error if any error occurs.
* **Parameters:**
  1. request: Contains the JSON payload with file\_content and parsed\_log\_data.
* **Returns:**
  1. **400 Bad Request:** If file\_content is missing.
  2. **200 OK:** If analysis is successful, returns:
     + log\_analysis: The original log entries.
     + descriptions: The generated descriptions for each log entry.
  3. **500 Internal Server Error:** If an exception occurs during processing.

**Conclusion:**

* Both APIs are asynchronous and designed to handle errors gracefully, ensuring robust responses in case of user or system errors.
* The Login\_API focuses on user authentication, whereas the LogWise\_API integrates AI-powered error analysis to enhance log management and debugging processes.