API Tester

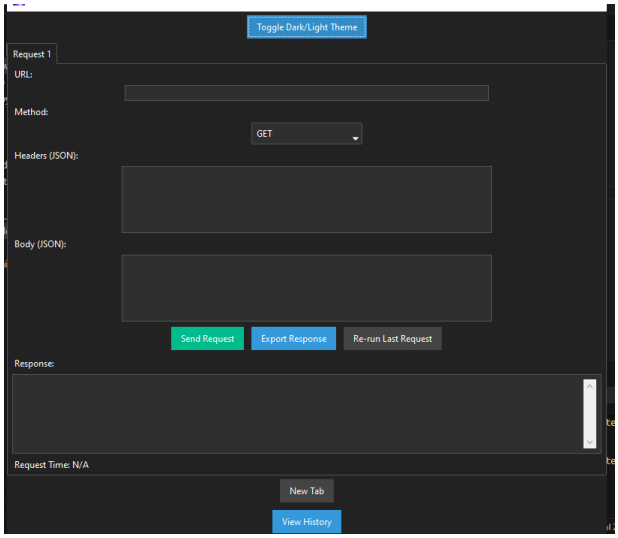
API testing is the process of verifying that an **Application Programming Interface (API)** works as expected — ensuring it's reliable, secure, and performs correctly for all possible requests and responses. API testing focuses on the **business logic layer** of an application (not the UI).
You send requests to an API endpoint and validate the responses — such as **status codes, data, headers, and response times**.

Target platform

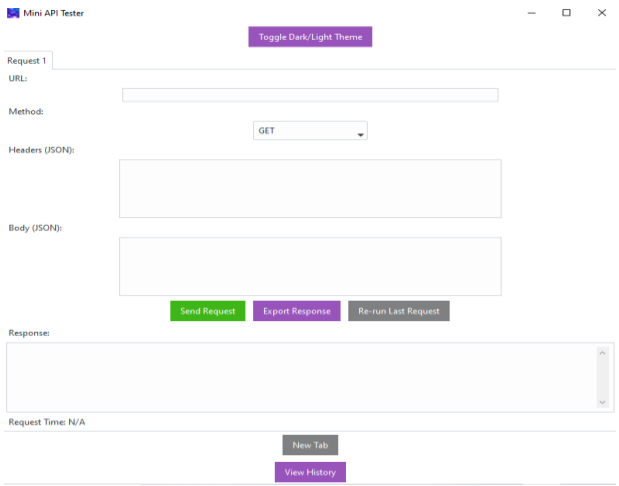
Windows 10 desktop (executable-ready). Built with Python and Custom Tkinter for a native-feeling GU that adapts automatically to dark and light themes.

Overview

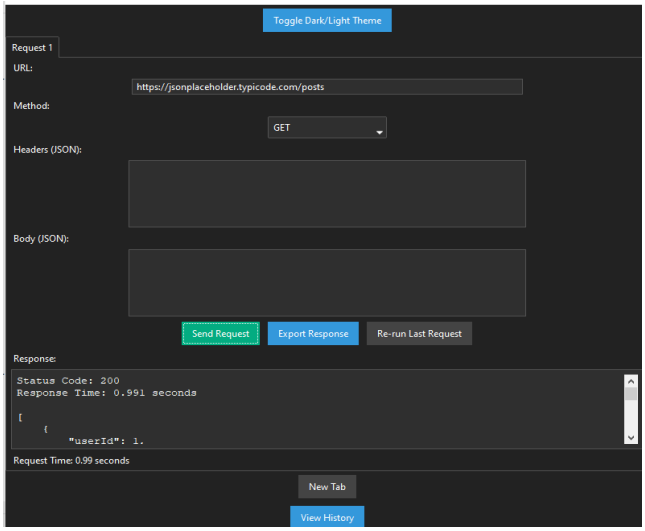
Dark Theme

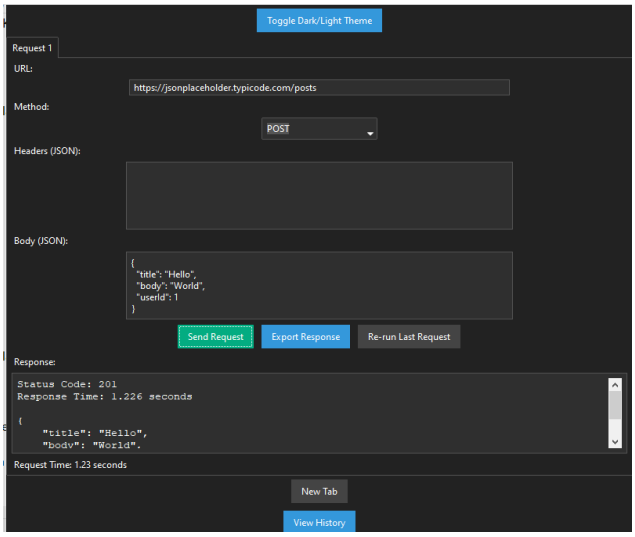


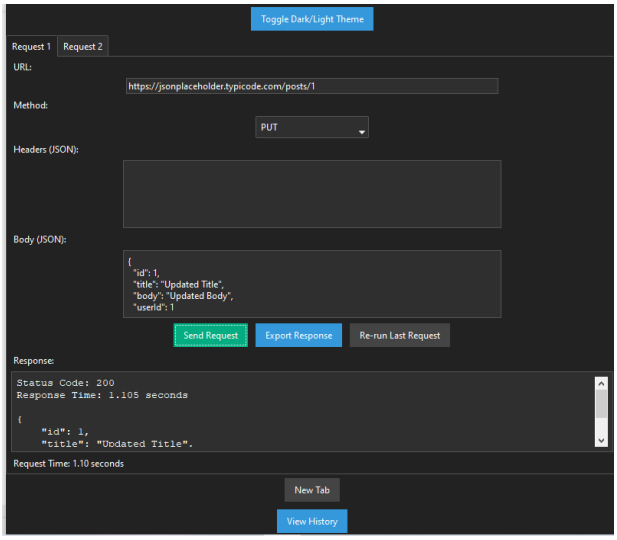
Light Theme

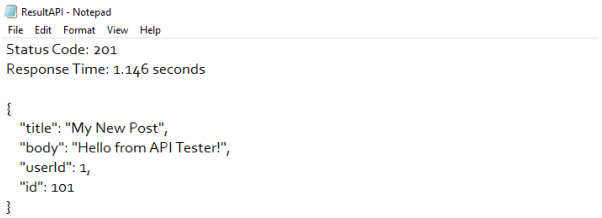


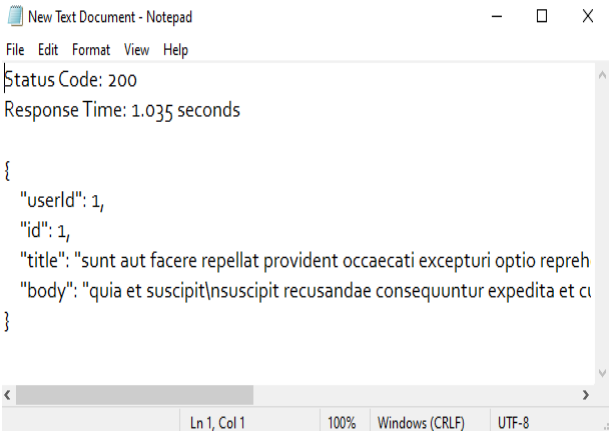
Making Requests



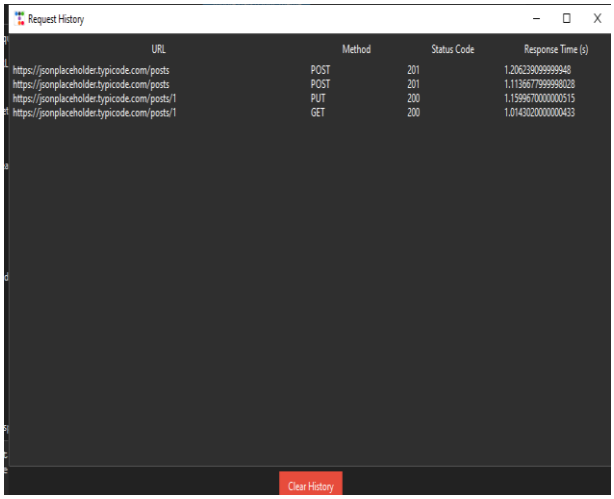








History Viewing



Icon

Features

 Graphical User Interface (GUI)

- Built with **Tkinter** (using ttkbootstrap for a modern look).
- User-friendly layout for quick API testing.

 HTTP Request Support

- Supports **GET**, **POST**, **PUT**, and **DELETE** methods.
- Allows full customization of the request.

 URL Input Field

- Enter any API endpoint URL.
- Validates before sending the request.

 Optional Headers & Body Input

- Headers and Body can be entered as **JSON** or **plain text**.
- Automatically formats JSON input for readability.
- Supports sending raw payloads.

 Send Request Button

- Executes the HTTP request instantly.
- Displays the response in the same window.
- Shows request execution time.

 Response Display Panel

- Shows **status code**, **headers**, and **body** clearly.
- Automatically detects JSON responses and pretty-prints them.
- Non-JSON responses shown as plain text.

 Request & Response History

- Saves every API request and response automatically.
- Two storage options available:

    - **SQLite** database (default)
    - **JSON** file storage (optional)
- Timestamped entries for each request.
- View, browse, and clear history directly from the app.

 Clear Fields Button

- Quickly resets all input and output fields to blank.
- Simplifies running multiple tests consecutively.

 Performance Timing

- Displays total time taken for each request (in seconds).

 Error Handling

- Gracefully handles network errors and invalid input.
- Displays descriptive error messages to the user.

 Cross-Thread Database Access

- Uses safe thread handling (check_same_thread=False) to log requests made from background threads.

 Extensible Design

- Modular code allows adding new HTTP methods or authentication features easily.
- History viewer and database layer are independent for easy upgrades.
-

Required Python Modules & Installation Commands

Tkinter (simple GUI):

```
pip install requests
```

```
pip install pillow
```

Packaging & Distribution

Tool	Purpose
PyInstaller	Convert Python script to .exe or .app
Virtualenv	Manage dependencies for packaging
requirements.txt	Track dependencies

Dependencies

- requests → for sending REST API calls
- jsonschema → to validate API response structure (optional but useful)
- python-dateutil → for parsing timestamps in your history
- customtkinter → modern GUI for your main window
- Datetime-→ Timestamp management

File Structure

```
api_tester/
├── main.py
├── ui/
|   └── main_window.py
├── data/
|   └── history.json
├── assets/
|   └── icons/
├── requirements.txt
|---READ.md
```

Open Source Code

https://github.com/ruheenatasneem/API-Testing/upload/main

Usage

1. Start the Application

Navigate to your project folder and run:

```
python main.py
```

The main window of the API Tester will open.

2. Send an API Request

1. **Enter the API URL** in the input box.
2. https://api.example.com/users

1. **Select the HTTP Method** from the dropdown — GET, POST, PUT, or DELETE.
2. *(Optional)* Add request headers or body if required (depending on your UI design).

3. Click **Send** to execute the request.

The app will display:

- **Response status code** (e.g., 200 OK)
- **Response time** (in seconds)
- **Response body** (JSON or text)

3. View Request History

Click the **Show History** button to display all previous requests.
Each entry shows:

- Timestamp
- HTTP method
- API endpoint URL
- Response status
- Execution time

Example:

> **[2025-11-12 13:05:21] GET https://api.example.com/users → 200 (0.342s)**
> **[2025-11-12 13:07:45] POST https://api.example.com/users → 201 (0.521s)**

4. Clear History

Click **Clear History** to remove all previously stored entries.
This will reset data/history.json to an empty array:

> **[]**

5. Persistent Storage

All requests are saved automatically to:

data/history.json

The file remains even after you close the app — so you can reopen and continue testing later.

History Management

All request history is stored in a local JSON file:

data/history.json

The file is automatically created if it doesn't exist.

Each record includes metadata such as:

API URL

HTTP Method (GET, POST, PUT, DELETE)

Response Status Code

Response Time

Optional Request Body

Timestamp

Contributing

Contributions are welcome!
Whether you want to report a bug, suggest a new feature, or improve the UI or documentation every contribution helps make **API Tester** better.

Author

Designed and developed by Ruheena Tasneem

Acknowledgements

We would like to express our gratitude to all the contributors, libraries, and tools that made the **API Tester** project possible.

- **Python** — for providing a powerful and versatile programming language.
- **Requests Library** — for simplifying HTTP request handling.
- **CustomTkinter** — for modern and user-friendly GUI components.
- **JSON & Python-Dateutil** — for managing and storing API request data efficiently.
- **Open Source Community** — for continued inspiration and learning resources.

A special thanks to all developers and testers who contributed to making API Tester a reliable and easy-to-use tool for API development and debugging.