HOME WORK

Front end

nng.com

FRONT END DEVELOPER HOMEWORK

JSON API Runner

1. PROJECT OVERVIEW

The goal of this project is to build a JSON-driven API execution system that allows triggering various backend services based on a structured JSON input. The system processes these requests asynchronously and returns results to the user.

It includes:

- a client-side interface (UI),
- backend services,
- a logging module,
- and a central dispatcher for routing API calls.

Project Structure:

```
project-root/
│
├── backend/
│   ├── dispatcher.js         # central logic that processes the JSON input
│   ├── apis/
│   │   ├── userService.js    # various async functions
│   │   └── imageService.js
│   └── logger.js             # logger module
│
├── frontend/
│   ├── index.html            # base HTML page
│   ├── style.css             # simple styling
│   └── main.js               # handles API calls and UI interactions
│
└── server.js                 # Express server (or other backend)
```

2. SUBCOMPONENTS

2.1 Logger

Simple timestamped backend logging for API requests and errors.
The `logger.js` module exposes a `log()` function that can log to console or file.

2.2 Backend APIs

TExample APIs:

- **userService:** e.g., `getUserProfile`
- **imageService:** e.g., `getImageByName`
- **mathService:** advanced computational APIs including:
    - `getFibonacci`: asynchronously computes the n-th Fibonacci number using memoization or dynamic programming.
    - `multiplyMatrices`: asynchronously multiplies two matrices (2D arrays) and returns the resulting matrix.

The dispatcher receives an array of calls (via JSON), parses them, dynamically loads the target API module, and invokes the specified method with parameters.

2.3 Frontend UI

The user can:

- write JSON input manually,
- choose an API from a dropdown,
- toggle between light/dark themes,
- switch button alignment (left or right),
- click “Run” to send the request.

Results are displayed in a `<pre>` block, styled using CSS.

Optional improvements:

- API Parameter Form Generation: Dynamically generate an input form based on the selected API. Each function's parameters are turned into editable input fields.
- Improved Response Display: Use syntax highlighting libraries (e.g., Prism.js or Highlight.js) instead of raw `<pre>` tags.
- Loading Animation & Status Feedback: Show a loading spinner or progress bar during async calls. Display error messages clearly if something fails.

3. UI Layout Summary

UI Elements:

- `select` – API selector: `userService`, `imageService, etc`
- `button` – “Toggle Theme”: light ↔ dark mode

- `button` – “Toggle Align”: align controls left ↔ right
- `textarea` – JSON input
- `button` – “Run”: submit the API request
- `pre` – result display block

By default, the layout uses:

- light theme,
- left-aligned control buttons.

Both can be changed live with their respective buttons.

4. Submission & Version Control Requirements

- The project **must be version-controlled using Git**.
- All code should be committed regularly with meaningful commit messages.
- The complete project must be **hosted on a public GitHub repository**.
- The repository should include:
  - A `README.md` file with a brief project description and usage instructions
  - The full source code (frontend + backend)
  - Any sample JSON inputs used for testing
- Submit the project by **sharing the GitHub repository URL**.

5. How to Run the Project (Node.js)

Prerequisites

- Node.js (v14 or newer recommended)
- npm (comes with Node.js)

Steps

```
# 1. Navigate to the project folder
cd project-folder/

# 2. Install dependencies
npm init -y
npm install express

# 3. Start the backend server
node server.js
```

The backend server will run by default on http://localhost:3000.

Access the Frontend

Open your web browser and go to:

```
http://localhost:3000
```