GUI Task

Task: Build a Modular Robot Control Panel

Objective

Create a modular, web-based robot control panel using **HTML**, **CSS**, **JavaScript**, and **jQuery**. This project will help you practice front-end development, modular design, and dynamic content loading while simulating a robotics interface.

Features to Implement

1. Robot Status Display

- Display the robot's status (e.g., "Online" or "Offline").
- Use CSS to change the color of the status indicator:
  - Green for "Online".
  - Red for "Offline".

2. Basic Controls

- Add buttons to control the robot's movement:
  - "Forward"
  - "Backward"
  - "Left"
  - "Right"
- Use jQuery to handle button clicks and display the selected action (e.g., "Moving Forward").

3. Camera Feed Placeholder

- Add a placeholder for a camera feed (e.g., a static image or a video element).
- Allow the user to toggle the camera feed on/off using a button.

4. Real-Time Data Display

- Simulate real-time data (e.g., speed, temperature) and display it in a dashboard.
- Use JavaScript to update the data dynamically.

5. Emergency Stop Button

- Add a big red button for emergency stop.
- When clicked, it should stop all actions and display an alert.

Design Guidelines

1. Dynamic Loading Unit

- Implement a jQuery-based dynamic loading system to load HTML components dynamically.
- Use a function like `loadComponent('path_to_component', 'target_container')` to load components into specific containers.
  Hint: Use Ajax to load the component
- Example:

```
function loadComponent(path, target) {
  $(target).load(path);
}
```

2. Modular Component Design

- Break the system view into reusable components (e.g., `status.html`, `controls.html`, `camera.html`, `data.html`).
- Load these components dynamically into the main layout using the `loadComponent` function.
- Example:

```
$(document).ready(function () {
  loadComponent('components/status.html', '#status-container');
  loadComponent('components/controls.html', '#controls-container');
  loadComponent('components/camera.html', '#camera-container');
  loadComponent('components/data.html', '#data-container');
});
```

3. Three-Column Dashboard Layout

- Design the dashboard with a **three-column layout**:
  - **Left Column**: Robot status and basic controls.
  - **Middle Column**: Camera feed and emergency stop button.
  - **Right Column**: Real-time data display.
- Use CSS Flexbox or Grid to create the layout.

4. Code Organization

- Organize your codebase into separate files and folders:
  - `index.html`: Main HTML file.
  - `styles.css`: Global styles.
  - `script.js`: Main JavaScript/jQuery logic.
  - `components/`: Folder for reusable HTML components (e.g., `status.html`, `controls.html`).

- Follow software design principles. Front-end stack does have its specific design patterns as well. However, a clean code is just a clean code. That's your pattern to follow for now :D.

5. Fault Tolerance

- Ensure the dynamic loading system handles errors gracefully (e.g., missing components or failed requests).
- Display a user-friendly message if a component fails to load.

Requirements

HTML

- Create a structured layout for the control panel with a **three-column design**.
- Include placeholders for dynamically loaded components:

  ```html
  <div id="status-container"></div>
  <div id="camera-container"></div>
  <div id="controls-container"></div>
  <div id="data-container"></div>
  ```

CSS

- Use CSS Flexbox or Grid to create a **three-column layout**.
- Style the control panel to make it visually appealing and user-friendly.
- Ensure the layout is responsive and works well on different screen sizes.

JavaScript/jQuery

- Implement the `loadComponent` function to load HTML components dynamically.
- Add interactivity to the control panel:
  - Handle button clicks for movement controls.
  - Toggle the camera feed on/off.
  - Update real-time data dynamically.
  - Implement the emergency stop functionality.

Example Output

Your control panel should look something like this:

```
-----------------------------------------------------------
| Left Column              | Middle Column        | Right Column
-----------------------------------------------------------
| [Status: Online]         | [Camera Feed]        | [Real-Time Data]
| [Forward] [Backward]     | [Toggle Camera]      | - Speed: 0 m/s
```

```
| [Left]    [Right]     | [Emergency Stop]   | - Temperature: 25 °C
----------------------------------------------------------
```

Key Guidelines

- Try to design each component independently.
- Avoid the god-unit anti-pattern.
- Keep the design simple, modular, and user-friendly.
- Follow best practices for code organization and maintainability. We haven't talked about any yet but they are not hard to find though.
- **HAVE FUN :D**