Assignment: AI-Powered Frontend Project using LangGraph

Overview

Develop a system that processes a **Software Requirements Specification (SRS) document** as input, analyzes its content using **LangGraph**, and generates a complete **AI-powered frontend project**. The system should follow best practices in frontend development, including component-based architecture, API integration, UI testing, debugging, deployment, and documentation.

LangGraph Workflow Design

Defining the Workflow Architecture

1. Identify the necessary **nodes, agents, and tools** to support:
    a. Component generation
    b. State management
    c. API integration
    d. UI testing
    e. Debugging and refinements
    f. Iterative improvements
2. Implement the workflow using **LangGraph**:
    a. Define nodes and their interactions.
    b. Establish edges to determine the sequence of execution.
    c. Maintain a **GraphState** to persist data (e.g., components, styles, errors, iterations).
    d. Implement automated feedback loops for self-improving AI-driven development.

Milestones

Note: Milestones **1, 2, 3, 5, and 6** are core components of the **LangGraph workflow** and should be implemented as either **nodes, agents, or tools**. LangGraph will orchestrate the workflow, leveraging **LLM-driven automation** for AI-generated outputs and **engineering logic** for structured execution.

Milestone 1: Analysis

- Develop an AI workflow to analyze the SRS document and Screenshot and extract relevant details for frontend generation.
- **Key components to extract:**
    - Required **UI components** (buttons, forms, tables, modals, navigation, etc.).
    - State management requirements (**global state, API data handling, UI interactions**).
    - API endpoints and expected responses.
    - UI accessibility requirements.
    - Styling and branding guidelines (from SRS or design references).
- Extract Design language and other details about the look and feel of the UI from **UI screenshots** using **Llama 3 Vision (Groq Preview)** for structured data extraction.

Milestone 2: Generate Project Setup

- **Frontend Framework:**
    - Initialize a structured **Angular** project.
    - Set up state management (**NgRx or Services-based state management** as required).
    - Install necessary dependencies (**RxJS, Angular Material**).
- **Folder Structure:**
    - Define a modular project structure to ensure scalability and maintainability.

Sample Modular Folder Structure

```
project_root/
├── src/
│   ├── app/
│   │   ├── components/
│   │   │   ├── button.component.ts
│   │   │   ├── modal.component.ts
│   │   │   ├── form.component.ts
│   │   │   └── index.ts
│   │   ├── pages/
│   │   │   ├── home.component.ts
│   │   │   ├── dashboard.component.ts
```

```
│   │   │   ├── login.component.ts
│   │   │   ├── signup.component.ts
│   │   │   └── index.ts
│   │   ├── services/
│   │   │   ├── api.service.ts
│   │   │   ├── auth.service.ts
│   │   │   ├── user.service.ts
│   │   │   └── index.ts
│   │   ├── state/
│   │   │   ├── store.ts
│   │   │   ├── user.reducer.ts
│   │   │   ├── settings.reducer.ts
│   │   ├── assets/
│   │   ├── styles/
├── tests/
├── Dockerfile
├── angular.json
├── package.json
├── .env
├── README.md
```

Milestone 3: Autonomous UI Generation Workflow

- **Generate UI Components using LLM:**
  - Generate reusable **Angular components** for extracted UI elements.
  - Ensure components follow best practices, such as:
    - **Component-based architecture** (small, single-responsibility components).
    - **Accessibility** (proper ARIA attributes, keyboard navigation, etc.).
    - **Styling consistency** (SCSS, or Angular Material themes).
    - **Modular and reusable design**.
- **Integrate API Calls:**
  - Use **HttpClientModule** for API integration.
  - Implement proper **error handling** and state management for API responses.

Testing & Debugging

- **Generate UI tests using LLM:**

    - Use Cypress.
    - Ensure proper **unit tests, integration tests, and end-to-end tests**.
- **Automated Debugging & Refinements:**
    - Implement **error detection** in the workflow.
    - If UI tests fail or components break, the system should **iteratively fix issues**.
    - Debugging agent should analyze logs, UI behavior, and regenerate affected code.

Benchmarking AI-Generated Code

- **Correctness:** Are the generated components functionally correct?
- **Performance:** Do components load efficiently?
- **Code Quality:** Is the code modular and maintainable?
- **Testing Coverage:** Do all interactions have corresponding tests?
- **Execution Success:** Does the project build and run successfully?

Milestone 4: Persistence & Iterations

- Ensure LangGraph retains previously generated UI components and maintains context.
- Example: If the system has generated a **Login form**, it should remember form field names and validation logic when generating the **Signup form**.
- The workflow should align new generations with previous iterations to ensure consistency.
- Track UI dependencies to prevent redundant re-generation.

Milestone 5: Deployment

- Use **LLM** to generate a **Dockerfile** for the frontend project.

Milestone 6: Documentation

- Generate a **graph visualization** of the LangGraph workflow using **draw_mermaid_png()**, which utilizes **Mermaid.Ink's API** to generate diagrams.
- Use an LLM to generate essential documentation:
    - **README.md** (setup, usage, project structure).
    - **Component documentation** (Props, States, API integration details).
    - **Code comments** for improved clarity.

Milestone 7: LangSmith Logging & Debugging

1. Create a **LangSmith** project to track logs for each execution.
    a. Log key details, including:
        - Graph execution steps
        - API calls and responses
        - Errors and debugging insights
    b. Iterations and refinements in code generation
    c. Logs should be structured to provide insights into system behavior over multiple runs.
2.
3. By default, draw_mermaid_png() uses Mermaid.Ink's API to generate the diagram analysis.

Milestone 8: Frontend Application as a Service

- Develop an API where users can:
    - Submit **SRS documents** (only .docx format).
    - Validate UI requirements and design specifications.
    - Generate a fully functional **frontend project**.
    - Receive a **hosted preview link** and LangSmith logs.