Technical Documentation

# 1. Project Overview

The AI-Powered Personalized Learning System is an intelligent full-stack web application that aims to assist learners in their educational journey by providing customized learning content. Users provide a topic of interest, available time, and their current level of understanding. Based on this input, the system dynamically creates a roadmap, quizzes, and additional learning materials using modern web technologies and AI services.

# 2. Backend

The backend of the application is developed using Flask, a Python micro web framework. It provides a RESTful API to handle various functionalities such as generating roadmaps, quizzes, translating content, and producing AI-based study resources. CORS is enabled to ensure seamless communication between the frontend (React) and the Flask backend.

* Key Backend Files:
* • base.py: Entry point of the Flask app. Contains route definitions for handling client API calls.
* • roadmap.py: Contains logic for generating a structured learning roadmap using Google Generative AI.
* • quiz.py: Handles the creation and response generation for topic-related quizzes.
* • generativeResources.py: Provides additional personalized resources (articles, explanations, etc.) for a better learning experience.

# 3. Frontend

The frontend is developed using React.js and offers a smooth user interface for learners. It allows users to input their preferences, view generated content, and interact with quizzes. It communicates with the Flask backend through Axios and handles routing using React Router.

* Key Frontend Files:
* • input/topic.js: Manages the topic input, time and level selection, and triggers the roadmap generation.
* • quiz/quiz.js: Displays quizzes and handles user interaction with quiz questions and answer selection.
* • roadmap/roadmap.js: Renders the personalized learning roadmap in a structured format using visual components.

# 4. API Routes

* The following API endpoints are available:
* • POST /api/roadmap: Accepts a JSON payload with topic, time, and knowledge level. Returns a personalized learning roadmap.
* • POST /api/quiz: Accepts course details and returns a set of questions tailored to the topic and subtopic.
* • POST /api/translate: Accepts an array of text and a target language. Returns translated strings.
* • POST /api/generate-resource: Generates supporting content such as summaries or related articles using AI.

# 5. Dependencies

* Backend Dependencies (from requirements.txt):
* • flask: Lightweight Python web framework to build REST APIs.
* • flask-cors: Extension to handle Cross-Origin Resource Sharing (CORS).
* • python-dotenv: Loads environment variables from .env file into the application.
* • google-generativeai: Library to interact with Google's Generative AI APIs.
* Frontend Dependencies (from package.json):
* • axios: Promise-based HTTP client for making API requests.
* • react: JavaScript library for building user interfaces.
* • react-router-dom: Enables routing and navigation in the React app.
* • chart.js: Library for rendering visual data in charts and graphs.

# 6. Setup & Installation

Follow the steps below to set up and run the system locally:

* Frontend Setup:
* • Navigate to the project root directory.
* • Run `npm install` to install all frontend dependencies.
* • Run `npm start` to start the development server at http://localhost:3000.
* Backend Setup:
* • Navigate to the `backend/` directory.
* • Run `pip install -r requirements.txt` to install required Python packages.
* • Make sure you have a `.env` file with your Google API key.
* • Run `python base.py` to start the Flask server at http://localhost:5000.

# 7. User Interface Overview

Here are the key components of the user interface:

* • Homepage: Allows the user to enter a topic and choose time and knowledge level.
* • Roadmap Page: Displays a personalized roadmap based on user input.
* • Quiz Page: Shows dynamically generated quiz questions based on the selected topic.
* • Resource Page: Presents additional study materials created using AI.

Note: You can capture and insert screenshots of these pages in your final documentation or presentation.

# 8. Developer Guide

This section outlines how developers can extend or customize the platform:

* Adding New Features:
* • Create new components in the `src/components/` or `src/pages/` directory.
* • Define new routes and API endpoints in `base.py` and connect them to frontend using Axios.
* Improving AI Integration:
* • Modify `roadmap.py` and `generativeResources.py` to improve prompt design or add new AI capabilities.
* • Ensure any new AI services use environment-based keys and are handled securely.
* Testing & Debugging:
* • Use browser developer tools (Console & Network tab) to debug API calls.
* • Add logging to backend files for better visibility during development.