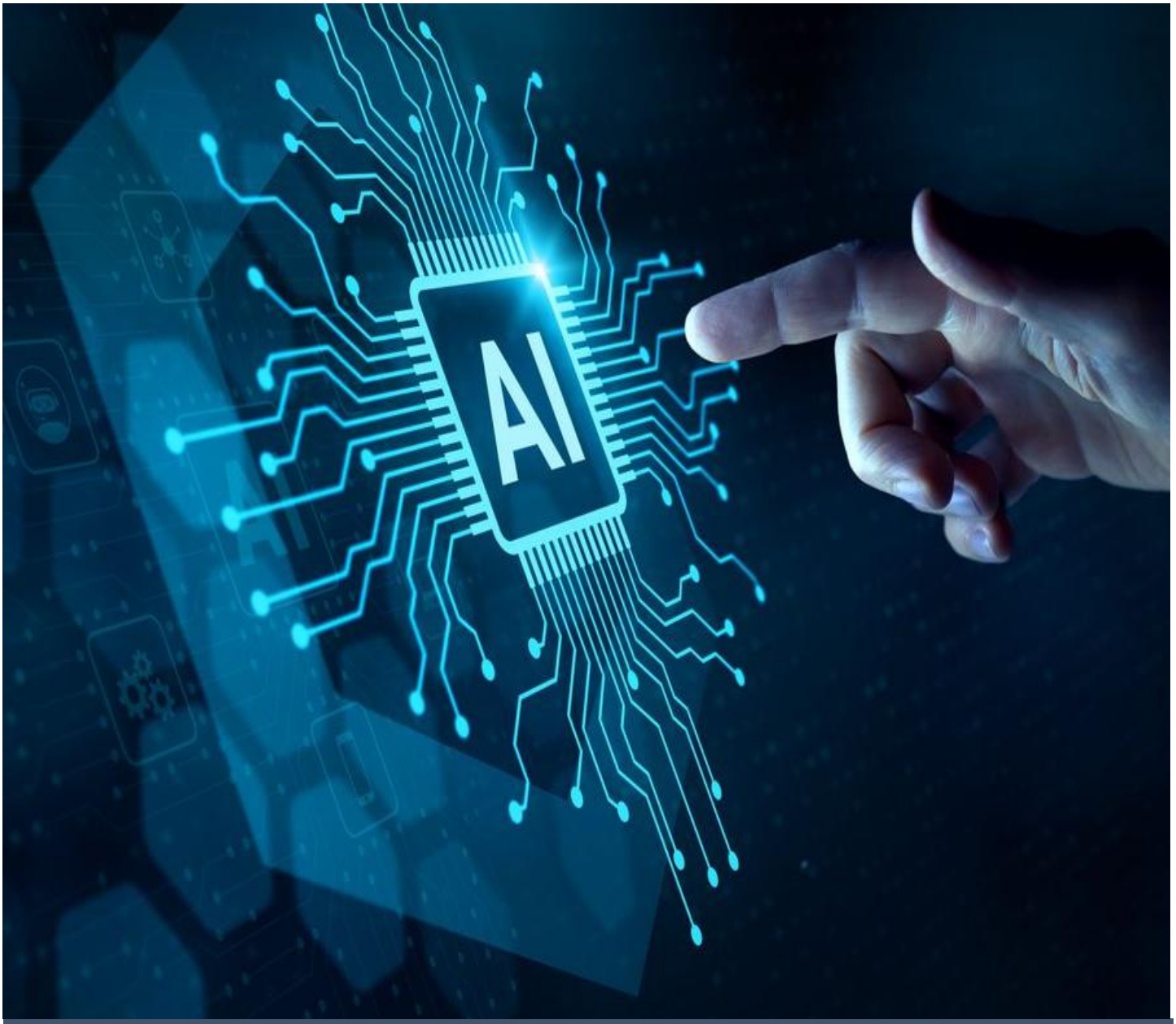

PROJECT REPORT

AI TEACHER AGENT

With Adaptive Difficulty

AI Teacher Agent – Comprehensive Project Report

1. Introduction & Problem Statement

1.1 Introduction

The **AI Teacher Agent** is an intelligent tutoring system designed to provide personalized educational assistance using AI-powered reasoning and natural language processing. The system integrates Large Language Models (LLMs) with structured data management to deliver comprehensive learning experiences, including topic explanations, quiz generation, performance tracking, and interactive study paths.

1.2 Problem Statement

Traditional educational platforms face several limitations: - **Lack of Personalization**: One-size-fits-all approaches fail to adapt to individual learning needs. - **Limited Interactivity**: Static content does not effectively engage students. - **Insufficient Feedback**: Students lack immediate and detailed feedback on their performance. - **No Learning Path Guidance**: Learners struggle to identify optimal learning sequences. - **Manual Content Creation**: Educators spend excessive time creating quizzes and study materials.

The AI Teacher Agent addresses these challenges by: - Providing AI-generated, context-aware explanations tailored to user queries. - Automatically generating diverse quiz questions from topics or PDF documents. Tracking student performance and dynamically adapting difficulty levels. - Creating personalized study paths with milestone tracking. - Maintaining conversation memory for contextual interactions. - Offering advanced dataset querying capabilities (search, filter, categorize).

1.3 Objectives

1. Develop an intelligent agent capable of understanding and responding to educational queries.
2. Implement AI reasoning features such as memory, summarization, and classification.
3. Create a robust validation layer to ensure data integrity and quality.
4. Build a comprehensive logging system for tracking all operations.
5. Design an intuitive and responsive user interface.
6. Support multiple data sources and query types.

---

2. System Architecture

2.1 Overall Architecture

The system follows a **three-tier architecture**:

```
Presentation Layer  →  Application Layer  →  Data Layer
(Frontend UI)          (FastAPI Backend)     (Databases & Files)
```

Presentation Layer - HTML, CSS, JavaScript - Interactive UI components - Real-time updates and visualizations

Application Layer - FastAPI REST APIs - Pydantic-based request/response validation - Business logic orchestration and AI integration

Data Layer - SQLite databases (main system DB and student DB) - JSON/CSV datasets - File storage for PDF uploads

2.2 Component Architecture

2.2.1 Frontend Components

- Single Page Application (SPA)
- Sections: Home, Topics, Quiz, Performance, Study Path, Feedback, Authentication
- Key features:
- Dynamic content rendering
- Real-time quiz generation and evaluation
- Interactive study path visualization
- PDF upload and processing
- Print and PDF export support

2.2.2 Backend Components

- **FastAPI App (`app.py`)** – RESTful API server
- **AI Agent (`agent.py`)** – Core intelligence engine
- **Memory Service (`memory.py`)** – Conversation memory management
- **Reasoning Service (`reasoning.py`)** – Summarization and classification
- **Data Service (`data_service.py`)** – Dataset operations

- **Database Layer (`db.py` , `student_db.py`)** – Data persistence

2.2.3 Data Flow

```
User Request → API Endpoint → Validation → AI Agent → LLM / Database → Response
```

---

3. Agent Design & Reasoning Logic

3.1 Agent Architecture

The AI agent class serves as the central intelligence component:

```
class AIAgent:
    memory: AgentMemory
    reasoning: ReasoningService
    data_service: DataService
    llm: ChatGoogleGenerativeAI | None
```

3.2 Core Capabilities

3.2.1 Topic Explanation

- Intelligent detection of content type (introduction, definition, examples, applications, advanced concepts, research papers).
- Dynamic prompt generation based on topic and difficulty.
- Integration of external learning resources (YouTube, websites).
- Template-based fallback when LLM is unavailable.

3.2.2 Quiz Generation

- Topic-based and PDF-based quiz generation.
- Difficulty-aware question creation (Beginner, Intermediate, Advanced).
- Generation of surplus questions for user selection.
- LLM-powered generation with robust template fallbacks.

3.2.3 Quiz Evaluation

- Semantic evaluation of student answers using LLMs.
- Automated scoring and percentage calculation.

- Detailed, per-question feedback.
- Persistent performance tracking in the database.

3.3 Reasoning Features

3.3.1 Agent Memory (Mandatory)

- Maintains conversational context across sessions.
- User-specific memory retrieval with timestamp ordering.
- Enables context-aware and coherent responses.

3.3.2 Summarization

- Condenses content while preserving key points.
- Sentence-based extraction with quality scoring.

3.3.3 Classification

- Keyword-based content categorization.
- Confidence scoring for classification accuracy.

4. Dataset Description

4.1 Dataset Formats

The system supports both **JSON** and **CSV** datasets containing: - Topic metadata - Category and difficulty -
Tags and key concepts

4.2 Dataset Operations

- **Search**: Case-insensitive matching across all fields.
- **Filter**: Category, difficulty, and tag-based filtering.
- **Categorize**: Grouping topics by category with configurable limits.

---

5. LLM & Algorithmic Methods

5.1 LLM Integration

- **Provider**: Google Gemini

- **Model**: `gemini-pro`
- **Temperature**: 0.7
- **Library**: LangChain

5.2 Fallback Strategies

- Template-based quiz generation.
- Keyword-based classification.
- Sentence-extraction summarization.

---

6. Validation Using Pydantic

6.1 Validation Principles

- Strong type enforcement
- Range and constraint validation
- Default values for resilience
- Standardized scoring metrics

6.2 Scoring Metrics

- **Completeness Score**: Measures response completeness (0–1).
- **Confidence Score**: Indicates reliability and quality (0–1).

---

7. Database Schema & Logging

7.1 Databases

- **Main DB**: System logs, agent decisions, chat history.
- **Student DB**: Student profiles, quiz attempts, progress tracking.

7.2 Logging Strategy

- All operations are logged atomically.
- Indexed tables for high-performance queries.
- User-isolated logging with timestamps.

---

8. User Interface Design

8.1 Design Philosophy

- Modern, clean, and responsive UI.
- High accessibility and readability.
- Visual feedback through animations and states.

8.2 Key UI Sections

- Home dashboard
- Topic explanations
- Quiz generation and evaluation
- Performance analytics
- Study path visualization
- Feedback and authentication

---

9. Testing & Evaluation

9.1 Testing Methods

- Manual functional and UI testing.
- Integration testing between frontend and backend.
- Scenario-based user acceptance testing.

9.2 Performance Targets

- API response time: < 2 seconds
- Database queries: < 100 ms
- PDF processing: < 5 seconds

---

10. Challenges & Limitations

- Dependency on LLM availability and rate limits.
- Limited PDF text extraction for image-based files.
- English-only language support.
- SQLite scalability constraints.

11. Conclusion & Future Enhancements

11.1 Conclusion

The AI Teacher Agent successfully demonstrates how AI-driven reasoning, structured validation, and modern UI design can be combined to create an effective intelligent tutoring system. The modular architecture ensures flexibility, robustness, and future scalability.

11.2 Future Enhancements

- OCR support for image-based PDFs.
- Multi-language capabilities.
- Adaptive learning and personalized recommendations.
- Advanced analytics and gamification.
- Migration to scalable databases and microservices.