

**TOBB ETU**

**Economy & Technology University**

**BIL 495 / YAP 495**

**Software Design Description (SDD)**

**Reference:** IEEE 1016-2009

## Document Control Table

|  |  |
| --- | --- |
| Document Title | Project Closure Report |
| Document ID | (To be assigned) |
| Prepared By (Author) | KoRN |
| Reviewed By |  |
| Approved By |  |
| Preparation Date |  |
| Approval Date | 27.12.25 |
| Version / Revision | 1.0 |
| Confidentiality Level | Internal / Restricted / Public |
| Reference Standards | IEEE 15288, IEEE 12207, INCOSE SE Handbook v5 |

## Change Record (Revision History Table)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Revision | Date | Prepared By | Reviewed/Approved By | Description of Change |
| 1.0 | (Date) | (Author) | (Approver) | Initial Release |
| 1.1 |  |  |  | Minor editorial updates |
| 1.2 |  |  |  | Added stakeholder feedback section |

[1. Introduction](#_heading=)

[1.1 Purpose](#_heading=)

[1.2 Scope](#_heading=)

[1.3 References](#_heading=)

[2. System Overview](#_heading=)

[2.1 System Architecture](#_heading=)

[2.2 Design Rationale](#_heading=)

[3. Detailed Design](#_heading=)

[3.1 Component Descriptions](#_heading=)

[3.2 Interface Descriptions](#_heading=)

[3.3 Data Design](#_heading=)

[3.4 Algorithm Design](#_heading=)

[4. Traceability](#_heading=)

## 1. Introduction

This document provides the **detailed software design specification** of the AI-Based Adaptive Course Generator.  
It translates the architectural concepts defined in the SAD into concrete **modules, interfaces, data structures, and algorithms** to guide implementation and maintenance.

## Purpose

This document covers the internal design of:

* User Interface subsystem (Streamlit)
* Backend API subsystem (FastAPI)
* LLM interaction layer
* Data management and adaptation subsystems
* Caching and optimization mechanisms

## Scope

This document covers the internal design of:

* User Interface subsystem (Streamlit)
* Backend API subsystem (FastAPI)
* LLM interaction layer
* Data management and adaptation subsystems
* Caching and optimization mechanisms

## 1.3 References

System Architecture Document (SAD)

Software Requirements Specification (SRS)

IEEE 1016-2009 Software Design Description

IEEE 42010 Architecture Description

ISO/IEC 25010 Software Quality Model

## 2. System Overview

## 2.1 System Architecture

The system follows a **layered service-oriented architecture** consisting of Presentation, Application, Model Adapter, and Data layers, as defined in the SAD document.

## 2.2 Design Rationale

Key design decisions include:

* **Layer separation** to ensure maintainability and testability
* **Service adapter pattern** to avoid LLM vendor lock-in
* **Prompt caching** to reduce API cost and latency
* **EMA-based adaptation engine** for stable mastery tracking
* **Lightweight persistence** (SQLite + FAISS) to minimize infrastructure overhead

## 3. Detailed Design

## 3.1 Component Descriptions

| ***Component*** | ***Responsibility*** |
| --- | --- |
| StreamlitUI | Collects user input, displays lessons/quizzes |
| APIGateway (FastAPI) | Orchestrates system workflow |
| LessonGenerator | Builds lesson content from topic & context |
| QuizConstructor | Generates quizzes and exams |
| FeedbackHandler | Grades responses and produces feedback |
| EMAAdaptiveEngine | Updates mastery score and adapts difficulty |
| LLMServiceAdapter | Communicates with external LLM APIs |
| PromptCache | Stores hashed prompts and responses |
| SQLiteRepository | Stores session and cache metadata |
| FAISSIndex | Performs semantic retrieval over datasets |

## 3.2 Interface Descriptions

| **From** | **To** | **Data** |
| --- | --- | --- |
| StreamlitUI | APIGateway | Topic input, content upload |
| APIGateway | FAISSIndex | Semantic search query |
| APIGateway | LLMServiceAdapter | Prompt text |
| LLMServiceAdapter | APIGateway | Generated lesson/quiz |
| StreamlitUI | FeedbackHandler | Quiz answers |
| EMAAdaptiveEngine | LessonGenerator | Mastery score |

## 3.3 Data Design

**SQLite Tables (Simplified)**

* session(id, topic, mastery\_score, timestamp)
* prompt\_cache(hash, prompt\_text, response\_text)
* quiz\_results(session\_id, question\_id, is\_correct)

**FAISS Index**

* Vector embeddings of dataset paragraphs
* Metadata: source, token count, reading level

## 3.4 Algorithm Design

The system operates through four main algorithmic processes: lesson generation, quiz generation, adaptive learning, and prompt optimization.

When the user provides a topic or content, the system retrieves relevant contextual information from the FAISS vector index and constructs a structured prompt for the language model.  
Before sending the request to the LLM, the system checks whether the same prompt has already been processed using a SHA-256 hash. If a cached response exists, it is returned immediately; otherwise, the prompt is sent to the LLM and the generated lesson is stored for future use.

Based on the lesson content, the system automatically generates a quiz that covers different cognitive levels. After the user submits the answers, the system evaluates performance and produces personalized feedback.

The system adapts future content using an Exponential Moving Average (EMA) mastery model. Low mastery reinforces previous topics, while high mastery introduces more advanced material, ensuring personalized learning progression.

This design reduces API cost, improves response time, and enables stable and effective adaptive learning.

## 4. Traceability

| **Requirement (SRS)** | **Design Element** |
| --- | --- |
| Generate lesson from topic | LessonGenerator |
| Generate quizzes | QuizConstructor |
| Collect feedback | FeedbackHandler |
| Adaptive learning | EMAAdaptiveEngine |
| Performance ≤ 10 sec | PromptCache + FAISS |
| Security & privacy | Input sanitization + session storage |

## 5. Appendix

Future extensions supported by design:

* User authentication
* Progress analytics dashboard
* LMS integration
* Multilingual content support

## 6. References

IEEE 42010 – Systems and Software Engineering — Architecture Description

ISO/IEC 25010 – Software Quality Model