MXB2026-Sylhet-Fakibazz-Master-Moshai

Please test our beginner app. This is the **first stage** of the application and **not the final version**.

Click the link below and open it in **Microsoft Edge** or **Google Chrome**.
Then, click on the **three-dot (⋮) menu**, select **"Add to Home Screen"**, and tap **Install**.
The application will then be installed on your device.

Link: https://mastermoshai-millionxbangladesh-fakibazz.lovable.app

03. AI & System Architecture

Project Name: Master-Moshai — The AI Learning Companion

The **Master-Moshai** platform is built on a scalable, modular, and AI-first architecture that combines modern web technologies with multiple large language models to deliver personalized learning experiences.

3.1 System Overview

The system is divided into four main layers:

1. **Frontend (Client Layer)**
2. **Backend (Application Layer)**
3. **AI Model Layer**
4. **Database & Storage Layer**

Each layer is independently scalable and communicates through secure APIs.

3.2 Frontend Architecture

The frontend is developed using **Lovable.ai**, which internally utilizes:

- **TypeScript**
- **Tailwind CSS**
- Modern component-based UI architecture

Key responsibilities:

- Student dashboard and progress visualization
- Personalized study roadmap display
- AI chat interface for instant tutoring
- Exam and practice interfaces
- Responsive design for mobile and desktop users

All user actions (study progress, quiz attempts, AI interactions) are sent to the backend through secure REST APIs.

3.3 Backend Architecture

The backend is built using:

- **Node.js**
- **Express.js**
- **Postman** (for API testing and validation)

Core responsibilities:

- User authentication and role management
- Study data processing and performance analysis
- AI prompt orchestration and response handling
- Exam logic, scoring, and progress tracking
- Communication between frontend, AI models, and database

The backend acts as the **decision-making brain** of the system.

3.4 AI Model Layer

Master-Moshai integrates multiple AI models to ensure reliability, flexibility, and higher-quality responses:

- **OpenAI API**
- **Google Gemini**
- **DeepSeek**
- **Grok**

How AI is used:

- Personalized study planning
- Weak topic detection
- Question generation and evaluation
- Step-by-step explanations
- Motivational and emotional support

A **model-routing strategy** selects the most suitable AI model based on task type (reasoning, content generation, explanation, or emotional support).

3.5 Data Flow & Decision Logic

1. User interacts with the frontend (study, exam, or AI chat).
2. Request is sent to the backend.
3. Backend analyzes performance data.
4. Backend constructs structured prompts.
5. AI model processes the request.
6. Response is returned and stored.
7. Frontend updates dashboard and recommendations.

This creates a **continuous feedback loop** that improves personalization over time.

3.6 Database & Storage Layer

The system uses **MongoDB and lovableCloud** for data storage.

Stored data includes:

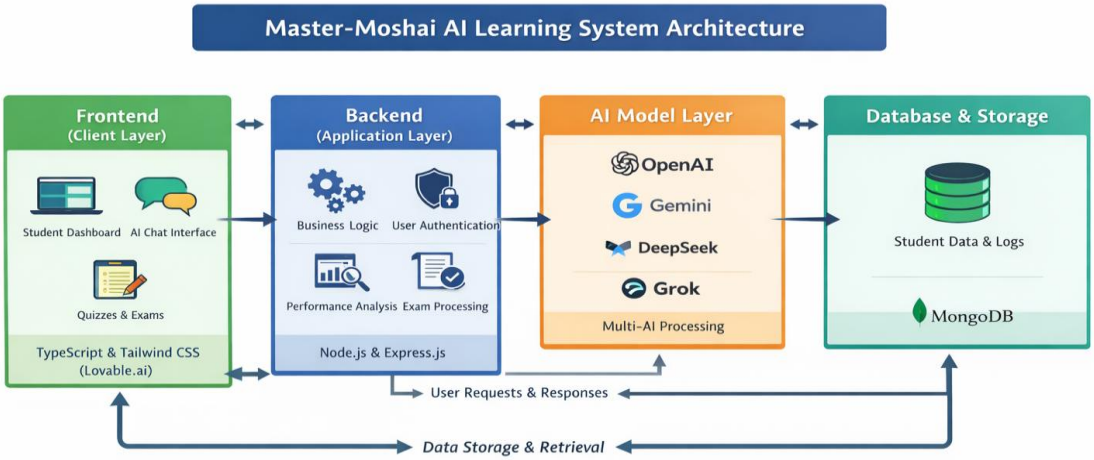
- User profiles
- Study history
- Exam results
- AI interaction logs
- Learning progress and recommendations

MongoDB’s flexible schema allows the system to adapt as new features are introduced.

3.7 Architecture Goals

- **Scalability:** Supports millions of students
- **Explainability:** Clear AI decision reasoning
- **Reliability:** Multiple AI models reduce dependency risk
- **Inclusivity:** Optimized for low-bandwidth environments
- **Security:** Secure API access and data handling

3.8 Architecture Diagram



Behind This Project: