**Project Design Phase-II**

**Technology Stack (Architecture & Stack)**

|  |  |
| --- | --- |
| **DATE:** | **26-06-2025** |
| **Team ID :** | **LTVIP2025TMID58052** |
| **Project Name :** | **LearnHub: Your Center for Skill Enhancement** |

**1. Architectural Style**

The system adopts a **Client-Server Architecture**, ensuring a clear separation between the user-facing interface and the underlying data processing and storage layers. All communication is handled via **RESTful APIs**, promoting modularity, scalability, and maintainability.

**2. Core System Components & Their Roles**

The system’s functionality is delivered through distinct but interconnected components:

**• Frontend (Client-Side)**

* **Purpose:** Manages user interactions and presents the UI/UX.
* **Functions:** Handles user registration/login, course enrollment, displays user/instructor/admin dashboards for progress tracking, supports in-app discussions and messaging, and delivers notifications.

**• Backend (Server-Side)**

* **Purpose:** Acts as the central hub for business logic, data validation, and core processing.
* **Functions:** Manages user accounts, processes course enrollments, handles assignment submissions, facilitates real-time discussions, manages notifications, provides secure APIs, and enforces security and role-based access control.

**• Database**

* **Purpose:** Provides persistent storage of all system data.
* **Functions:** Stores and retrieves user profiles, course details, assignment submissions, forum discussions, and messaging history efficiently while ensuring data integrity.

**• APIs (Application Programming Interfaces)**

* **Purpose:** Define the communication contract between the frontend and backend.
* **Functions:** Standardizes data exchange using JSON, ensuring secure, efficient, and consistent communication between system components.

**• External Services/Integrations**

* **Purpose:** Handle specialized functionalities outside the core application.
* **Functions:** Deliver automated real-time notifications to users via email or SMS for course updates, enrollment confirmations, or assignment deadlines.

**3. Technology Stack**

The technology stack is chosen for robustness, scalability, and developer efficiency:

**• Frontend Development**

* **UI/Responsiveness:** Bootstrap and Material UI for building a clean, responsive, and consistent interface across devices.
* **API Communication:** Axios for making efficient HTTP requests to backend APIs.

**• Backend Development**

* **Framework:** Express.js for building robust RESTful APIs with a minimalist and scalable architecture.
* **Real-time Communication:** Socket.io for enabling real-time, bidirectional in-app messaging and live updates between learners and instructors.

**• Database**

* **System:** MongoDB (NoSQL) for its flexibility in handling diverse data structures, scalability, and efficiency, particularly for dynamic data such as course content, discussions, and chat history.

**4. Scalability and Performance Considerations**

The system is designed to support scalability and high performance:

* **Stateless Backend:** The Express.js backend will be stateless, allowing horizontal scaling by adding more server instances as user demand grows.
* **MongoDB Sharding:** Enables horizontal database scaling as data volume increases.
* **Optimized APIs:** API endpoints and database queries will be optimized for fast response times and efficient data handling.
* **Socket.io Efficiency:** Designed for low-latency real-time updates, minimizing overhead for chat and live notifications.

**5. Deployment Strategy (Render)**

**• Platform:**

Render is selected as the unified cloud deployment platform for seamless management.

**• Strategy:**

* The **Frontend** and **Backend** will be deployed as separate services on Render, allowing independent scalability.
* Render’s integrated GitHub/GitLab integration will enable automatic deployments on code pushes.
* Features such as automatic SSL certificates, custom domains, and environment variable management will be utilized for a secure and smooth deployment pipeline.
* The MongoDB database will be provisioned through a managed service (e.g., MongoDB Atlas) and securely connected to the backend via environment-protected connection strings.