**Project Design Phase**

**Proposed Solution Template**

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

**1. Architectural Style**

* **Chosen Style:** Client-Server Architecture
* **Rationale:** Clear separation of concerns enables modularity, scalability, and easier maintenance. All communication occurs via RESTful APIs, ensuring a clean and organized interaction between the frontend and backend.

**2. Core System Components**

**Frontend (Client-Side)**

* **Purpose:** Handles user interaction and UI/UX presentation.
* **Key Functions:**
  + User registration and login
  + Course enrollment and tracking
  + Learner dashboard (progress tracking)
  + Discussion forums for interactivity
  + Instructor dashboard for course uploads
  + Notifications for reminders and updates

**Backend (Server-Side)**

* **Purpose:** Manages business logic, data validation, database interactions, authentication/authorization, and external service communication.
* **Key Functions:**
  + User account management
  + Course enrollment and progress tracking
  + Forum and message handling
  + Payment handling for premium courses
  + RESTful API provision for frontend interaction
  + Security enforcement and validation

**Database**

* **Purpose:** Persistent and reliable data storage.
* **Key Responsibilities:**
  + Storing user profiles, course data, progress, and discussions
  + Ensuring data integrity and consistency
  + Efficient querying and data retrieval

**APIs (Application Programming Interfaces)**

* **Purpose:** Define the structured contract between the client and backend.
* **Key Responsibilities:**
  + Secure and standardized data exchange using JSON
  + Clear endpoints for user registration, course management, and progress tracking

**External Services/Integrations**

* **Purpose:** Provide functionalities beyond the core backend capabilities.
* **Key Functions:**
  + Email notifications for enrollment confirmations and reminders
  + Payment gateway integrations for premium content access

**3. Technology Stack**

**Frontend**

* **UI/Responsiveness:** Bootstrap, Material UI
* **API Communication:** Axios for RESTful calls to the backend

**Backend**

* **Framework:** Express.js
* **Real-Time Communication:** Socket.io for live discussions and real-time progress updates

**Database**

* **System:** MongoDB (NoSQL database for scalable data management)

**4. High-Level Data Flow**

1. **User Interaction:** Users (learners, instructors) interact with the frontend interface.
2. **Request to Backend:** The frontend sends API requests to the backend using Axios.
3. **Backend Processing:** The backend executes business logic and validation.
4. **Database Interaction:** The backend stores or retrieves data from MongoDB.
5. **Real-Time Updates:** Socket.io facilitates bidirectional communication for live discussions and status updates.
6. **External Services:** The backend triggers email notifications using external services upon key events.
7. **Response to Frontend:** The backend sends responses back to the frontend.
8. **UI Update:** The frontend updates the user interface with new data for the user.

**5. Key Non-Functional Considerations**

**Security**

* User authentication using hashing, JWT tokens, and potential 2FA for enhanced security.
* Role-based access control (RBAC) for authorization.
* Data encryption during transit (HTTPS) and at rest in MongoDB.
* Input validation to prevent malicious attacks.
* API security through rate limiting and monitoring.
* Confidentiality measures integrated across the system.

**Scalability and Performance**

* Stateless backend design for horizontal scalability.
* MongoDB sharding for distributed data handling as user load increases.
* Optimized API and database queries for efficient data retrieval and updates.
* Use of Socket.io for efficient and scalable real-time communication.

**Deployment Strategy**

* **Platform:** Render, providing a unified cloud deployment solution.
* **Strategy:**
  + Frontend deployed as a static site or web service on Render.
  + Backend deployed as a web service on Render with environment variable management.
  + Automatic deployments from Git integration for CI/CD workflows.
  + Automatic SSL certification and custom domain management using Render’s managed services.
  + MongoDB Atlas used for managed database services, securely connected to the backend deployed on Render.