**Planning Phase**

**Project Charter**

**Overview:**

The **IOMAD** platform is designed as a multi-tenant Learning Management System (LMS) that allows organizations to create, sell, and manage courses. Each company has an independent workspace to handle user management, access control, and financial transactions.

**Project Goals:**

* Enable organizations to create and sell courses to other companies.
* Provide a scalable multi-company architecture.
* Ensure secure payment and user access control.
* Offer reporting and analytics for performance tracking.
* Maintain high security, usability, and system scalability.

**Scope:**

* **Course Creation & Management:** Companies can create and sell courses.
* **Multi-Tenant Architecture:** Each client has a private space for managing users and course access.
* **E-Commerce Integration:** Companies can purchase courses using secure payment methods.
* **User Role & Access Control:** Admins, learners, and instructors have different access permissions.

**Stakeholders:**

* **Clients (Organizations):** Purchase and manage courses for their employees.
* **Admins:** Oversee platform settings and user management.
* **Instructors:** Create and manage course content.
* **Learners:** Access and complete purchased courses.
* **Developers & QA Team:** Implement and test the platform features.

**Feasibility Study**

**Technical Feasibility:**

* **Technology Stack:** The system will use **MERN Stack (MongoDB, Express.js, React.js, Node.js)** to ensure scalability and flexibility.
* **Cloud Hosting:** The platform will be deployed on **AWS/Azure** for global availability.
* **Microservices Architecture:** Ensures modular development and easier maintenance.
* **Security Measures:** Implementation of **SSL encryption and data encryption**.

**Financial Feasibility:**

* **Development Cost:** Initial cost includes software development, hosting, and security compliance.
* **Revenue Model:** Revenue generation via course sales, subscription models, and premium features.
* **Scalability Costs:** Cloud-based infrastructure ensures cost-effective scaling.

**Operational Feasibility:**

* **User-Friendly Interface:** Ensuring intuitive navigation for companies and learners.
* **Automation:** Automated course enrollment, payment confirmation, and progress tracking.
* **Support & Maintenance:** Dedicated support for issue resolution and system upgrades.

**Project Plan**

**Timeline & Milestones:**

|  |  |
| --- | --- |
| **Phase** | **Tasks** |
| Planning | Define requirements, feasibility study, risk analysis |
| Design | UI/UX design, database schema, architecture planning |
| Development | Backend & frontend development, API integration |
| Testing | |  | | --- | |  |  |  | | --- | | Unit testing, security testing, performance optimization | |
| Deployment | Cloud hosting, system deployment, monitoring setup |
| Maintenance | |  | | --- | |  |   Ongoing bug fixes, feature updates |

**Resource Allocation:**

* **Developers:** Backend, frontend, and database engineers.
* **UI/UX Designers:** Creating an intuitive user interface.
* **QA Engineers:** Ensuring security and functional testing.
* **Project Managers:** Overseeing development progress.

**Business Case Document**

**Justification:**

* The demand for online learning is increasing, and **IOMAD** provides a solution tailored for businesses to create and sell courses.
* Existing LMS platforms do not offer **multi-tenant capabilities** where one company can create and sell courses to another.
* Secure transactions and user role-based access provide a competitive edge.
* The **revenue model** allows companies to monetize their knowledge.

**Development Methodology:**

* **Agile Approach:** Due to the complexity and dynamic nature of LMS requirements, an **Agile methodology** (Scrum) will be followed.
  + Iterative sprints for feature releases.
  + Continuous feedback from stakeholders.
  + Testing and bug fixes in parallel to development.
* **Waterfall Approach (for initial planning):**
  + Requirement gathering and feasibility studies will be documented before the Agile sprints begin.

**Requirement Phase**

The **Requirements Phase** in the **Software Development Life Cycle (SDLC)** is crucial as it defines the business needs, functional and non-functional requirements, and user interactions with the system. This phase consists of:

* **Business Requirement Document (BRD)**: Outlines high-level business needs and objectives.
* **Software Requirement Specification (SRS)**: Details functional and non-functional system requirements.
* **User Stories**: Describes user interactions and their expectations from the system.
* **Use Case Diagrams**: Provides a visual representation of interactions between users and the system.

**Business Requirement Document (BRD)**

**Business Needs**

The **IOMAD** platform is designed as a **multi-tenant Learning Management System (LMS)**, where multiple companies can independently create, manage, and sell courses. Unlike traditional LMS solutions that serve a single organization, **IOMAD** allows companies to share a common infrastructure while maintaining private workspaces.

**Key Business Objectives**

🔹 Provide **multi-client support** to allow multiple organizations to operate independently within the same system.  
🔹 Facilitate **course creation, management, and sales** between businesses.  
🔹 Implement **secure payment processing** for course transactions.  
🔹 Offer **role-based access control** for admins, instructors, and learners.  
🔹 Ensure **scalability** to support multiple organizations with thousands of concurrent users.  
🔹 Integrate **reporting and dashboards** for tracking user engagement and financial transactions.

**High-Level Requirements**

|  |  |
| --- | --- |
| **Requirement** | **Description** |
| Multi-Tenant Support | Each company has its own private space to manage courses and users. |
| Course Marketplace | Companies can create and sell courses to other organizations. |
| Secure Payments | Payment gateway integration for transactions. |
| User Roles & Access Control | Define roles like Admin, Instructor, and Learner with specific permissions. |
| Scalability & Security | Handle high traffic and secure data transmission. |

**Software Requirement Specification (SRS)**

The **Software Requirement Specification (SRS)** includes the functional and non-functional requirements necessary to develop the system effectively.

**Functional Requirements**

**1. User Management**

**Description:** The platform must support user registration, role assignment, and access control.  
a) Each company should be able to **register independently** and have separate environments.  
b) Users can be assigned **Admin, Instructor, or Learner** roles.  
c) Role-based access ensures **data security and restricted functionality**.

**2. Course Creation & Management**

**Description:** The system must allow course creation with various content formats.  
a) Instructors should be able to **upload videos, PDFs, quizzes, and assignments**.  
b) Companies should manage courses independently in **a private environment**.  
c) Courses should be categorized for **better navigation and accessibility**.

**3. Course Purchase & Payment Integration**

**Description:** The platform must enable seamless course transactions.  
a) Companies can list courses for sale with **pricing and descriptions**.  
b) Secure payment gateways like **Stripe and PayPal** must be integrated.  
c) Course purchase should provide **immediate access to content**.

**4. Multi-Tenant Architecture**

**Description:** The system must support multiple companies with isolated data environments.  
a) Each company should **have its own database records** for users and courses.  
b) No data should be shared between organizations to maintain **privacy**.

**5. Reporting & Dashboards**

**Description:** The system should offer reporting tools for insights and analytics.  
a) Admins should see **course engagement, financial performance, and user activities**.  
b) Learners should have dashboards showing **progress and completion rates**.

**Non-Functional Requirements**

|  |  |
| --- | --- |
| **Requirement** | **Description** |
| Performance & Scalability | The system should support thousands of concurrent users without performance degradation. |
| Security Measures | Implement SSL encryption, secure login mechanisms (OAuth, JWT), and data encryption. |
| Accessibility & Usability | Follow **WCAG 2.0** standards to ensure usability for all users, including those with disabilities. |
| Multi-Device Compatibility | The platform should work seamlessly on desktop, tablets, and mobile devices. |
| API Support | Allow external systems to integrate with the platform using **RESTful APIs**. |

**User Stories**

User stories define how different users interact with the system. Each story represents a **real-world use case** that guides development.

**As an Admin, I want to:**

a) Manage users by assigning them roles (Instructor, Learner).  
b) View analytics on **course engagement and sales**.  
c) Configure company branding (**logo, themes**).

**As an Instructor, I want to:**

a) Create and manage courses using an **easy-to-use editor**.  
b) Track student **progress and engagement**.  
c) Restrict course access to **specific learner groups**.

**As a Learner, I want to:**

a) Browse available courses and **purchase them** securely.  
b) Track my learning progress through **personalized dashboards**.  
c) Download **certificates upon course completion**.

**As a Company Owner, I want to:**

a) Sell courses and **receive payments** securely.  
b) View **revenue reports** and transaction history.

**Use Case Diagrams**

Use case diagrams **visually represent** interactions between users and the system.

**Use Case 1: User Role Management**

**Actors:** Admin, Instructor, Learner  
 **Flow:**

1. Admin registers a company and **adds users**.
2. Admin **assigns roles** (Instructor/Learner).
3. Users **log in** and access content based on **role permissions**.

Admin registers a company and adds users

Users log in and access content based on role permissions

End Use Case 1

Start

Admin assigns roles (Instructor/Learner)

**Use Case 2: Course Purchase & Enrollment**

**Actors:** Company Owner, Learner  
 **Flow:**

1. A company lists **courses for sale**.
2. Another company purchases the course via **payment gateway**.
3. Learners get **instant access** to the purchased course.

End Use Case 1

Admin assigns roles (Instructor/Learner)

Start

Admin assigns roles (Instructor/Learner)

Admin registers a company and adds users

**Use Case 3: Course Creation & Delivery**

**Actors:** Instructor, Learner  
 **Flow:**

1. Instructor creates a course with **videos, quizzes, and PDFs**.
2. Learners enroll and **complete lessons**.
3. Learners receive **certificates** upon completion.

Start

End Use Case 3

Learners receive certificates upon completion

Instructor creates a course with videos, quizzes, and PDFs

Learners enroll and complete lessons