CHAPTER 5

SYSTEM DESIGN

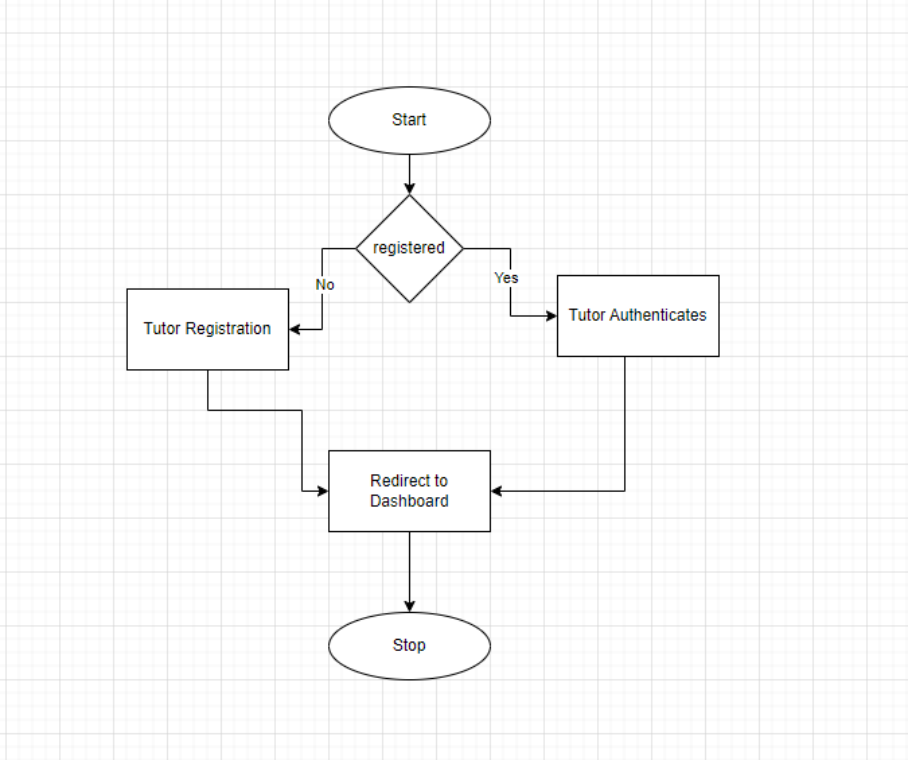
5.0 Introduction

System design is the process of defining the elements of a system such as the architecture, modules and components, the different interfaces of those components and the data that goes through that system. It is meant to satisfy specific needs and requirements of a business or organization through the engineering of a coherent and well-running system.

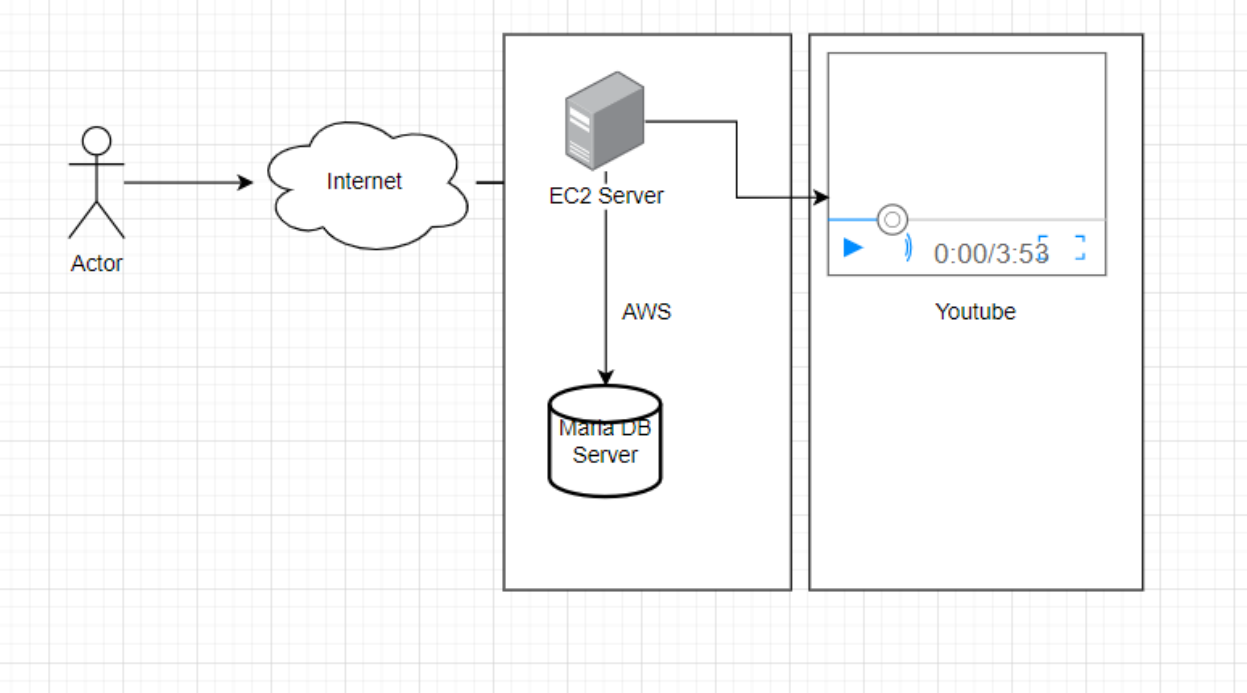
5.1 Current Diagram

Tutors, essential contributors to the educational ecosystem, are provided with dedicated registration and login functionalities within the system. Through these features, tutors can easily register their profiles, providing necessary information such as credentials, expertise, and contact details.

Once registered, tutors gain access to the LMS through a secure login portal, where they can authenticate their identities and access the platform's suite of teaching and learning resources. This ensures that tutors have a personalized experience tailored to their roles and responsibilities within the educational framework.

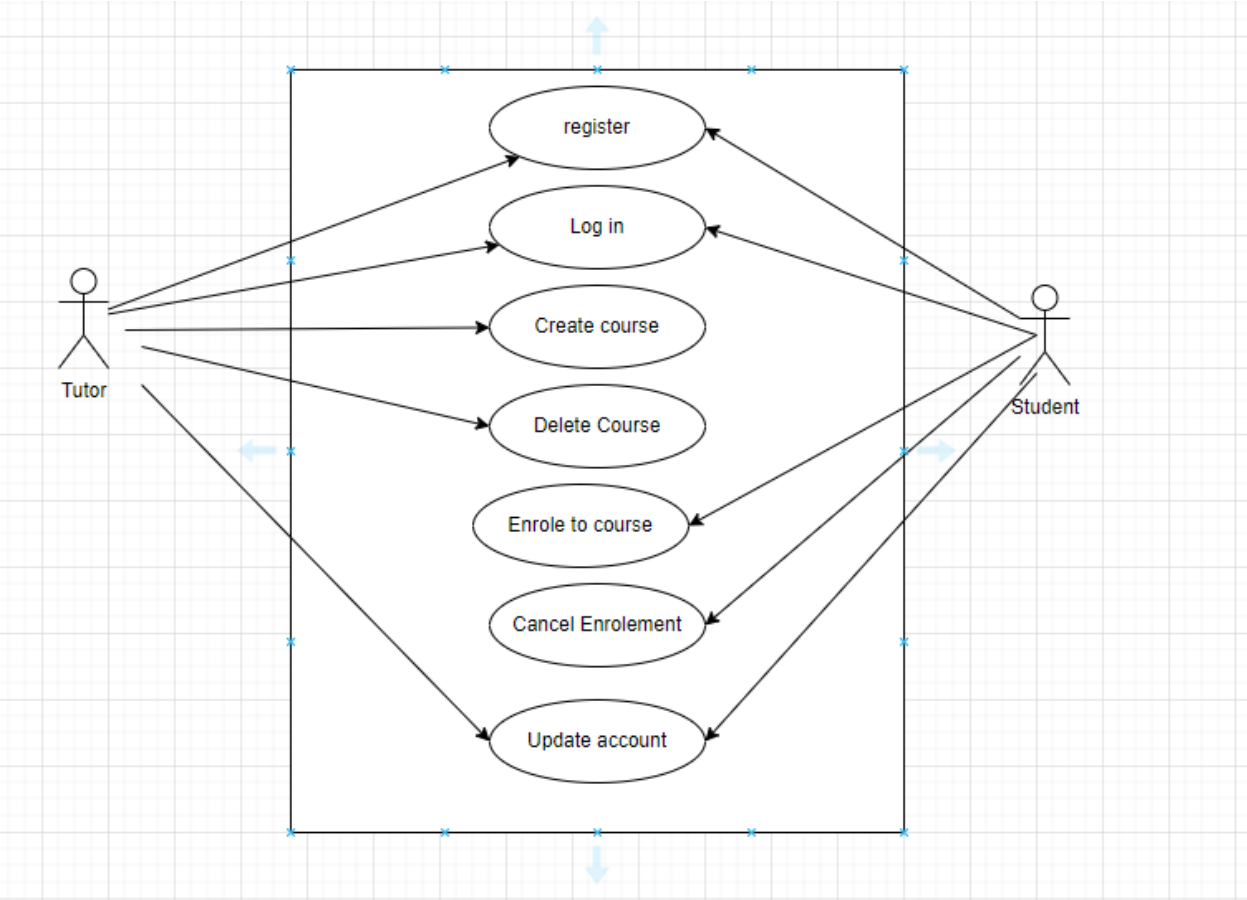


5.2 Proposed System Architecture



5.3 Use Case Diagram

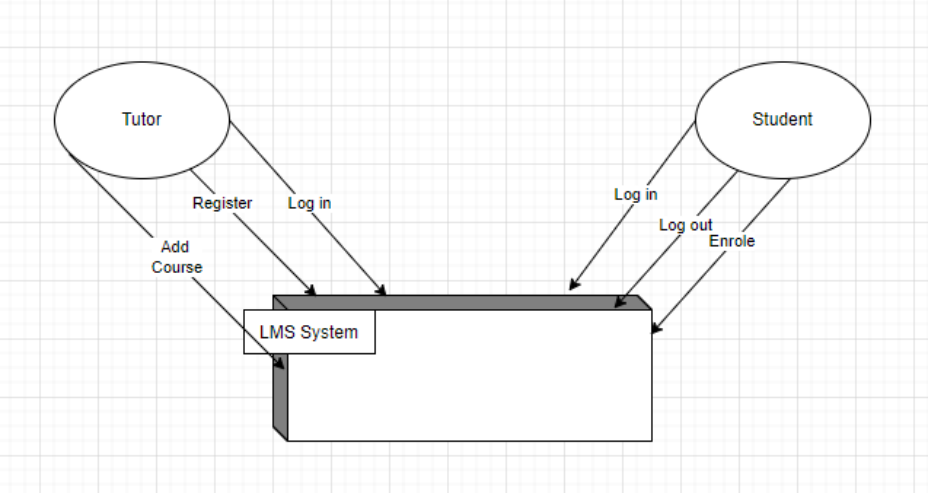
In their simplest form, a use case identifies the actors involved in an interaction and names the type of interaction. Use case model the system from the end users point of view. Use case describes the manner in which the user interacts with the system. Users interact with the system to achieve the required system functionality and derive the intended benefit from the system. Below is the use case of the LMS



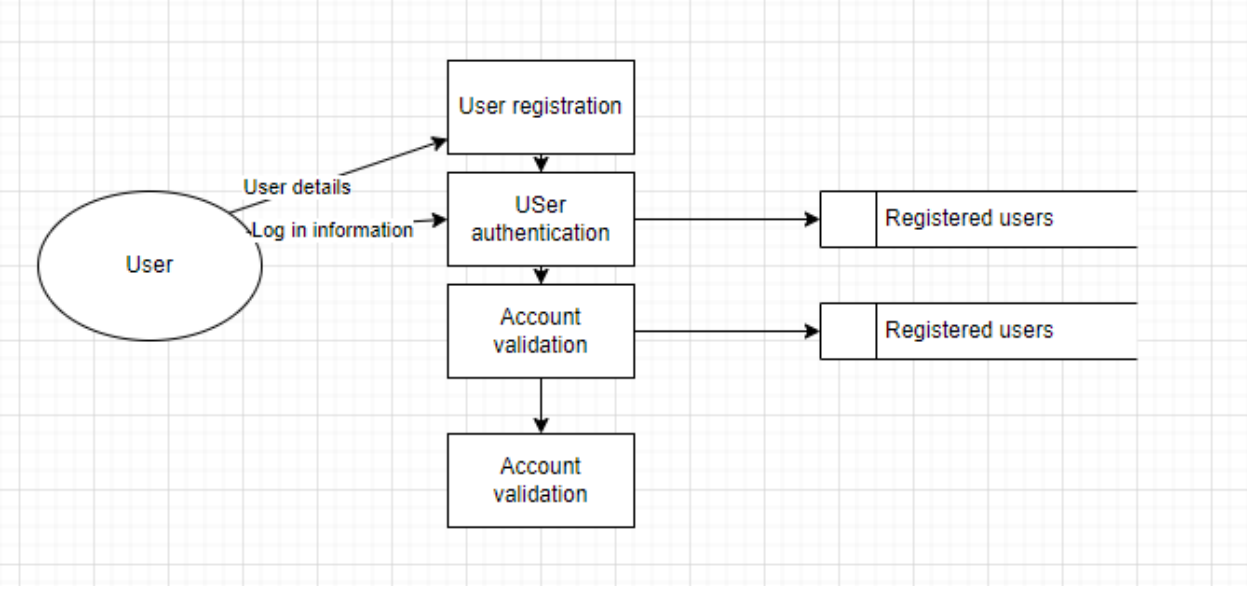
5.4 Data Flow Diagrams

A data flow diagram (DFD) maps out the flow of information for any process or system. It uses defined symbols like rectangles, circles and arrows, plus short text labels, to show data inputs, outputs, storage points and the routes between each destination. Data flowcharts can range from simple, even hand-drawn process overviews, to in-depth, multi-level DFDs that dig progressively deeper into how the data is handled

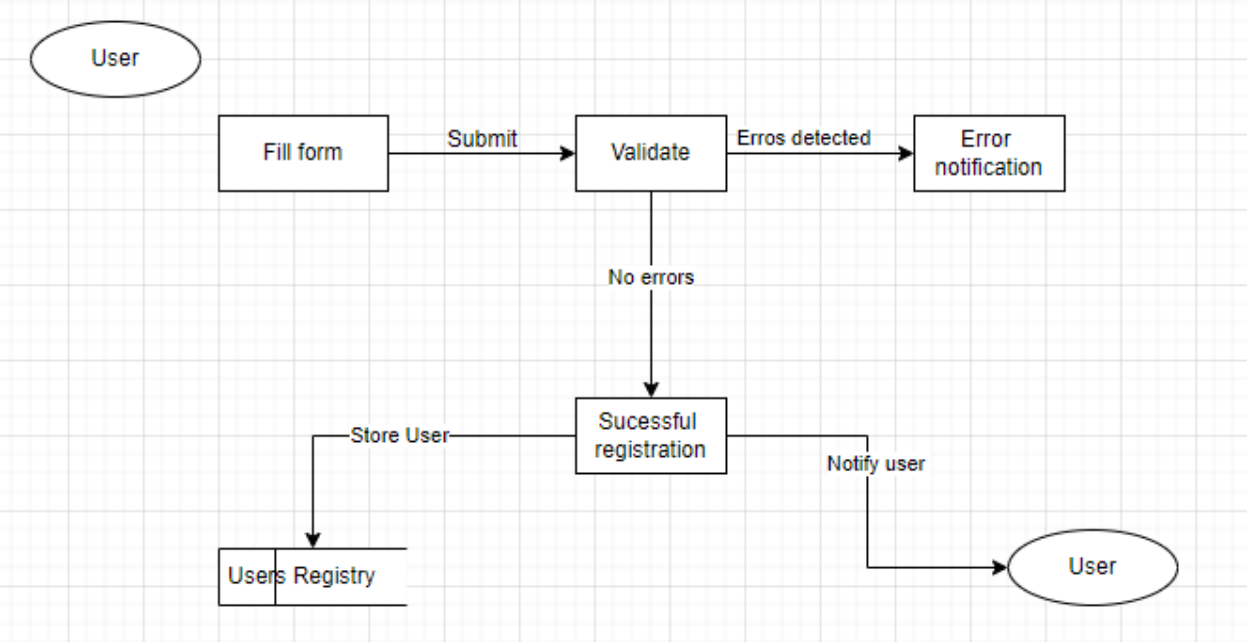
5.4.1 Context Diagram



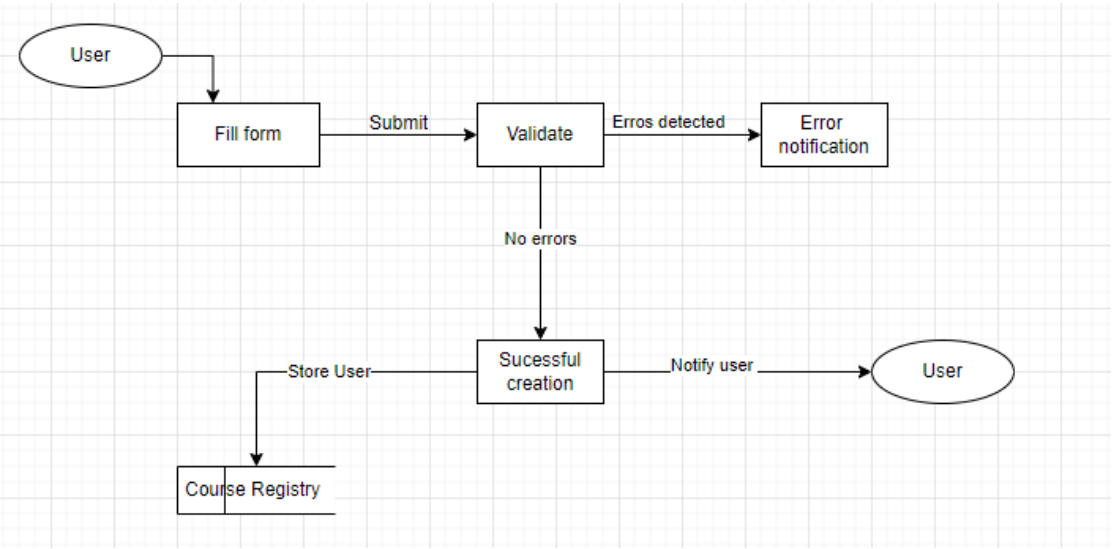
5.4.2: Diagram Zero



5.4.3: Diagram 1 User Registration



5.4.4: Diagram 1 Course Registration



5.5 Data Structure

A data structure is a framework for organizing, storing, and managing data. Data structures are usually described using algebraic notation. Any data structure is designed to organize data to suit a specific purpose so that it can be accessed and worked on appropriately.

I: User = Username* + Account type* + Email* + Password

II: Course = title + description + image + banner + status

II: Module = course ID + title + description + video link + time

Key * - Mandatory field

5.6 Data Dictionary

The data dictionary provides a detailed description of all tables found within the designer-created database. Thus, the data dictionary contains at least all of the attribute names and characteristics for

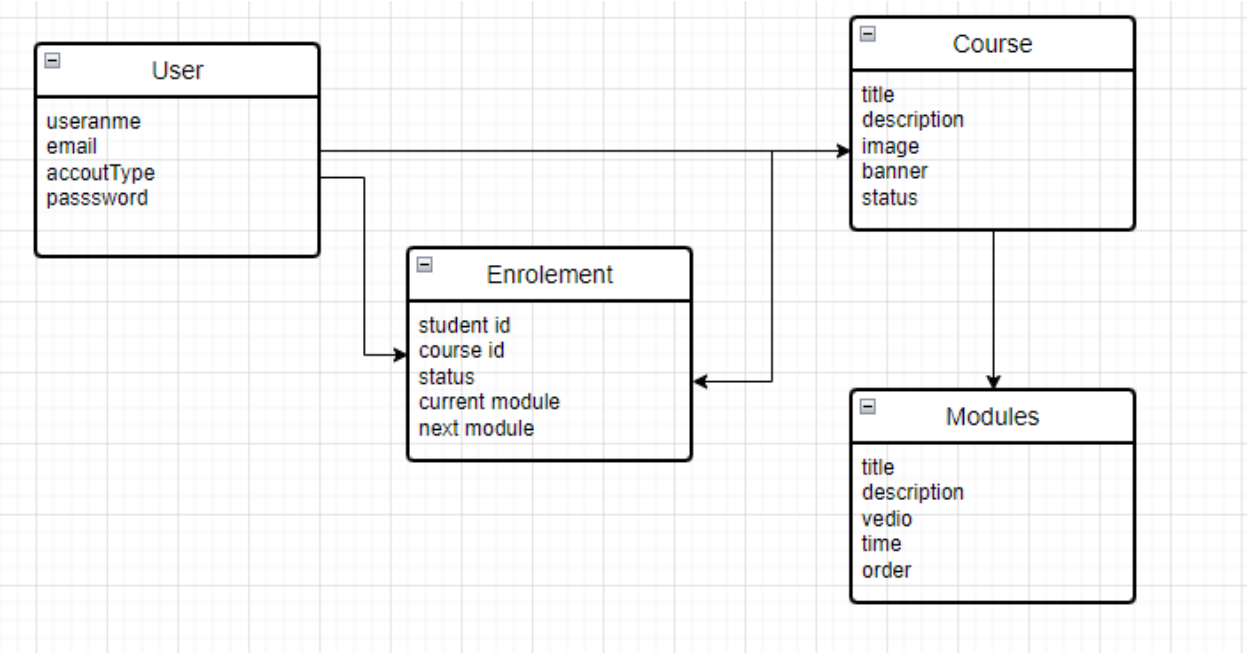
each table in the system. In short, the data dictionary contains metadata- data about data. (Rob & Coronel, 2009)

5.7 Database Design

Database Design is a collection of processes that facilitate the designing, development, implementation and maintenance of enterprise data management systems. The main objectives of database designing are to produce logical and physical designs models of the proposed database system. The methodology for designing database consists of three phases that is conceptual design, logical design, and physical design. These three phases are explained as follows:-

1. Logical Design

The logical model concentrates on the data requirements and the data to be stored independent of physical considerations. It does not concern itself with how the data will be stored or where it will be stored physically

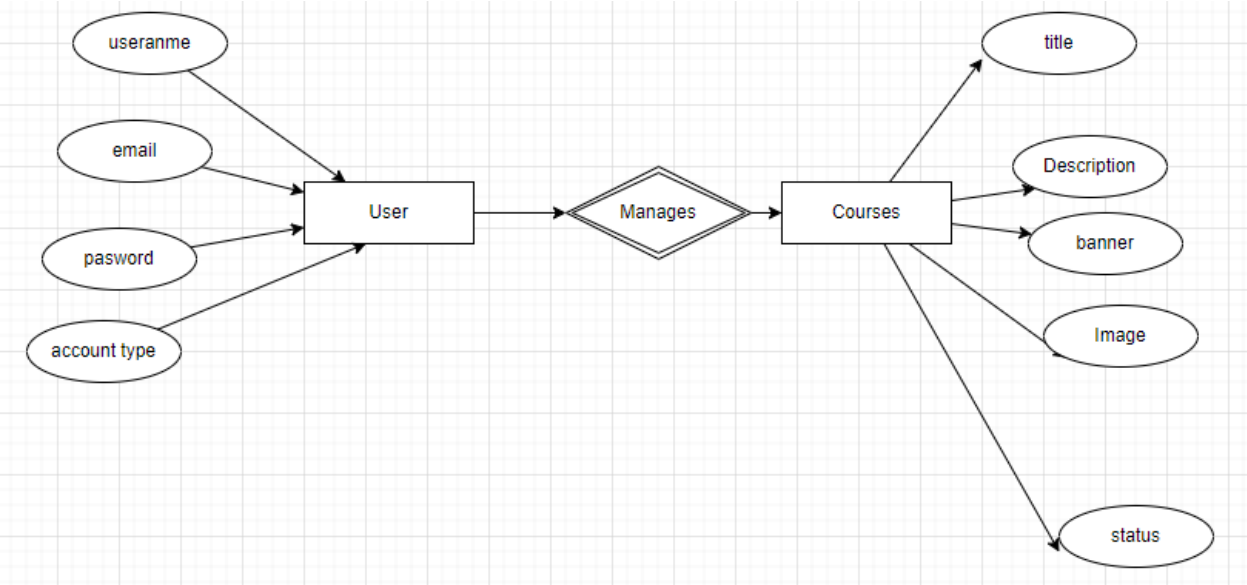


2. Physical Design

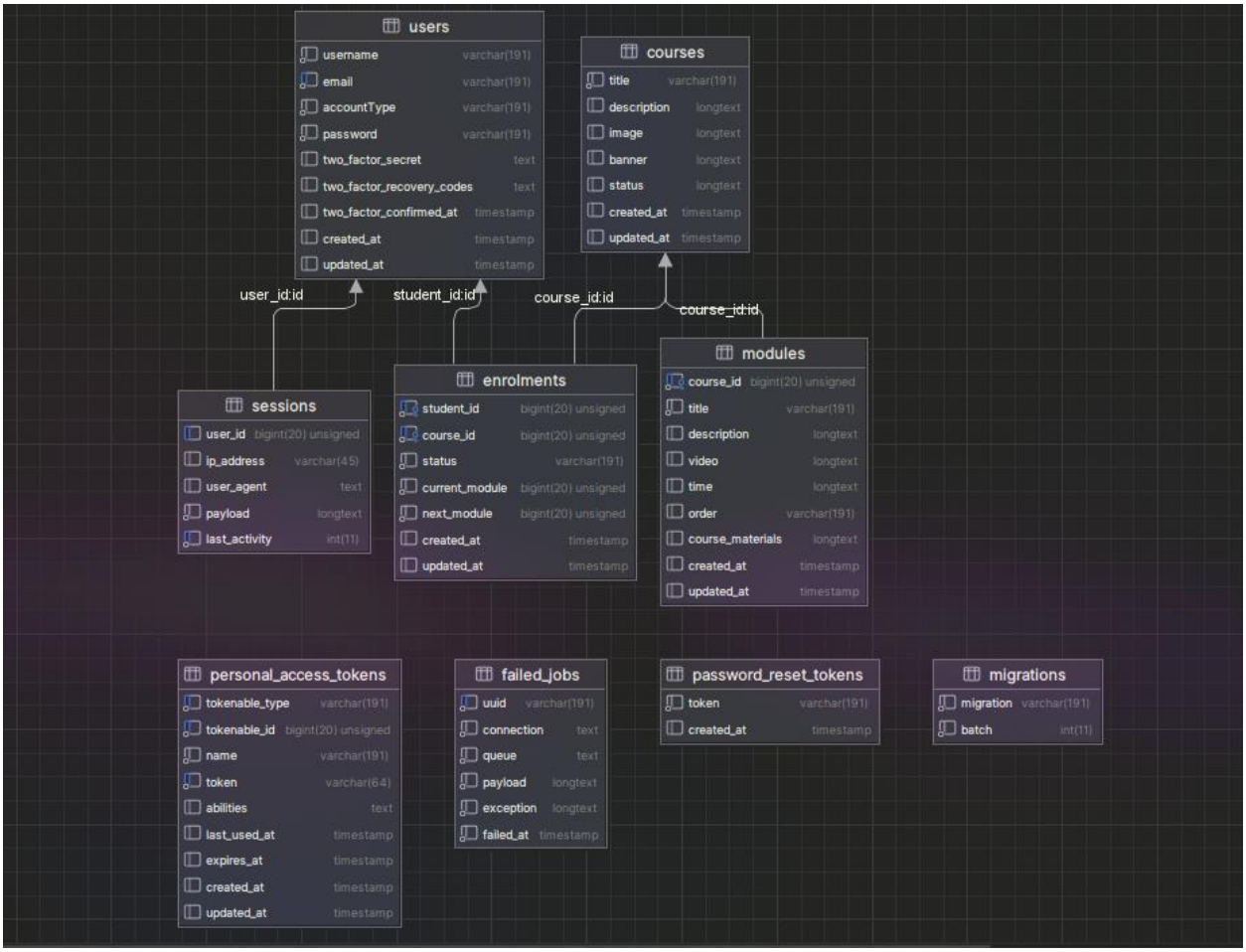
The physical data design model involves translating the logical design of the database into physical media using hardware resources and software systems such as database management systems (DBMS)

3. Conceptual Design

Conceptual database design includes the identification of the entities, relationship between entities and defining the attributes.



5.8 Entity Relationship Diagram



CHAPTER SIX

SYSTEM IMPLEMENTATION

6.0 Introduction

This chapter elaborates the Implementation issues of how the system design was implemented in terms of database implementation; application implementation for presentation or the user interface. The purpose of implementation was to transform the system design and the functional requirements into the actual system being developed. Planning for it begins early in the design stage. It covers the following areas:

a)Training of employees

b)Programming

c)System testing

d)Changeover procedure

e)Review and maintenance

6.1 Database Implementation

For the database implementation of our LMS system, MariaDB was selected as the backend solution. MariaDB, a robust and open-source relational database management system, was chosen for its compatibility with the AWS infrastructure and its proven performance in handling large datasets efficiently.

The implementation of MariaDB involved several key steps to ensure the seamless integration of the database with the overall system architecture. These steps included:

Database Schema Design: The design of the database schema was carefully crafted to accommodate the various data entities and relationships required by the LMS, including user profiles, course materials and enrollment records This schema design aimed to optimize data storage and retrieval while maintaining data integrity and consistency.

Data Migration: Existing data, if any, was migrated to the MariaDB database using appropriate data migration tools and techniques. This ensured that historical data from previous versions of the LMS or external sources could be seamlessly integrated into the new system.

Performance Tuning: Performance tuning measures were applied to optimize the database performance, including indexing key columns, optimizing query execution plans, and configuring MariaDB parameters for optimal resource utilization. These efforts aimed to enhance the responsiveness and scalability of the database, particularly under high load conditions.

Data Backup and Recovery: Robust data backup and recovery procedures were implemented to safeguard against data loss and ensure business continuity. Automated backup schedules were configured to regularly backup the database to secure storage locations within the AWS environment, providing redundancy and resilience against hardware failures or disasters.

By leveraging MariaDB for database implementation, our LMS system benefits from a reliable and scalable data management solution, enabling efficient storage, retrieval, and manipulation of educational content and user data.

6.2 Interface Implementation

Vue.js was chosen as the frontend framework for implementing the user interface of our LMS system. Vue.js, renowned for its simplicity, flexibility, and performance, proved to be an ideal choice for creating dynamic and responsive interfaces that enhance the user experience.

The interface implementation process involved several key components and considerations:

Component Design: The user interface was designed as a collection of reusable components, each encapsulating specific functionality and presentation logic. This modular approach facilitated code organization, maintenance, and extensibility, allowing developers to build complex interfaces by composing simple, self-contained components.

State Management: Vue.js's built-in state management solution, Vuex, was utilized to manage the application's state in a centralized manner. Vuex facilitated the sharing of data between components and ensured consistent behavior across the application, enabling seamless interaction and synchronization between different parts of the user interface.

Data Binding: Vue.js's declarative and reactive data binding capabilities were leveraged to establish dynamic connections between the application's data model and the user interface elements. This enabled real-time updates and synchronization of data between the frontend and backend components, providing users with a responsive and interactive experience.

Routing: Vue Router, the official routing library for Vue.js, was employed to implement client-side routing and navigation within the LMS application. Vue Router facilitated the creation of single-page applications (SPAs) with multiple views and nested routes, enabling smooth transitions between different sections of the interface without full page reloads.

Styling: The user interface was styled using a combination of CSS frameworks, such as Bootstrap or Tailwind CSS, and custom stylesheets to achieve a visually appealing and consistent design across different screens and devices. Vue.js's support for scoped CSS and CSS preprocessors allowed for encapsulation and modularization of styles, preventing style conflicts and enhancing maintainability.

By adopting Vue.js for interface implementation, our LMS system benefits from a modern and efficient frontend framework that empowers developers to build rich and interactive user experiences. Vue.js's extensive ecosystem of plugins and tools further enhances productivity and flexibility, enabling rapid development and iteration of new features to meet the evolving needs of our users."

6.2.1 User Interface Design

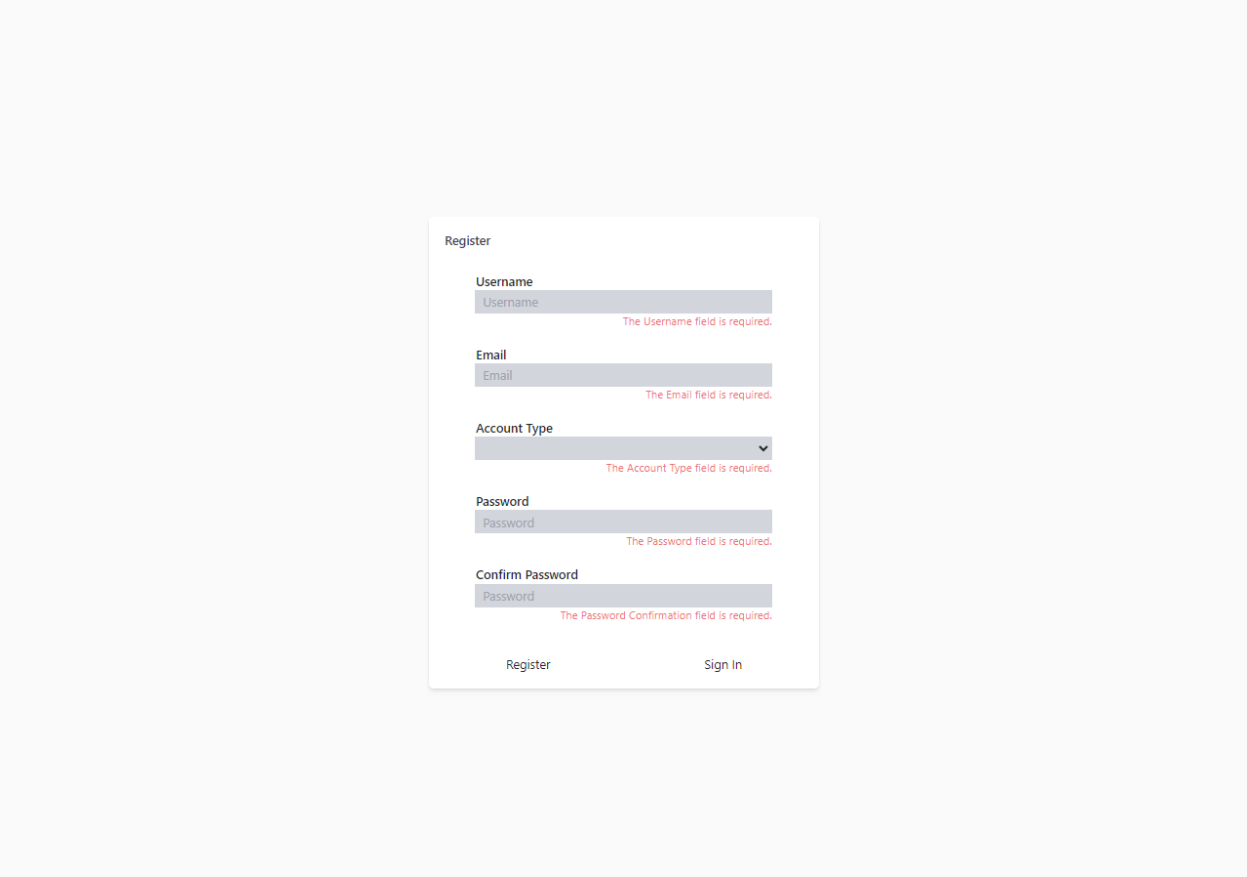
The User Interface design is very important not just from user point of view but from the overall success of the project. User Interfaces should be designed by keeping the following designing issues in mind;

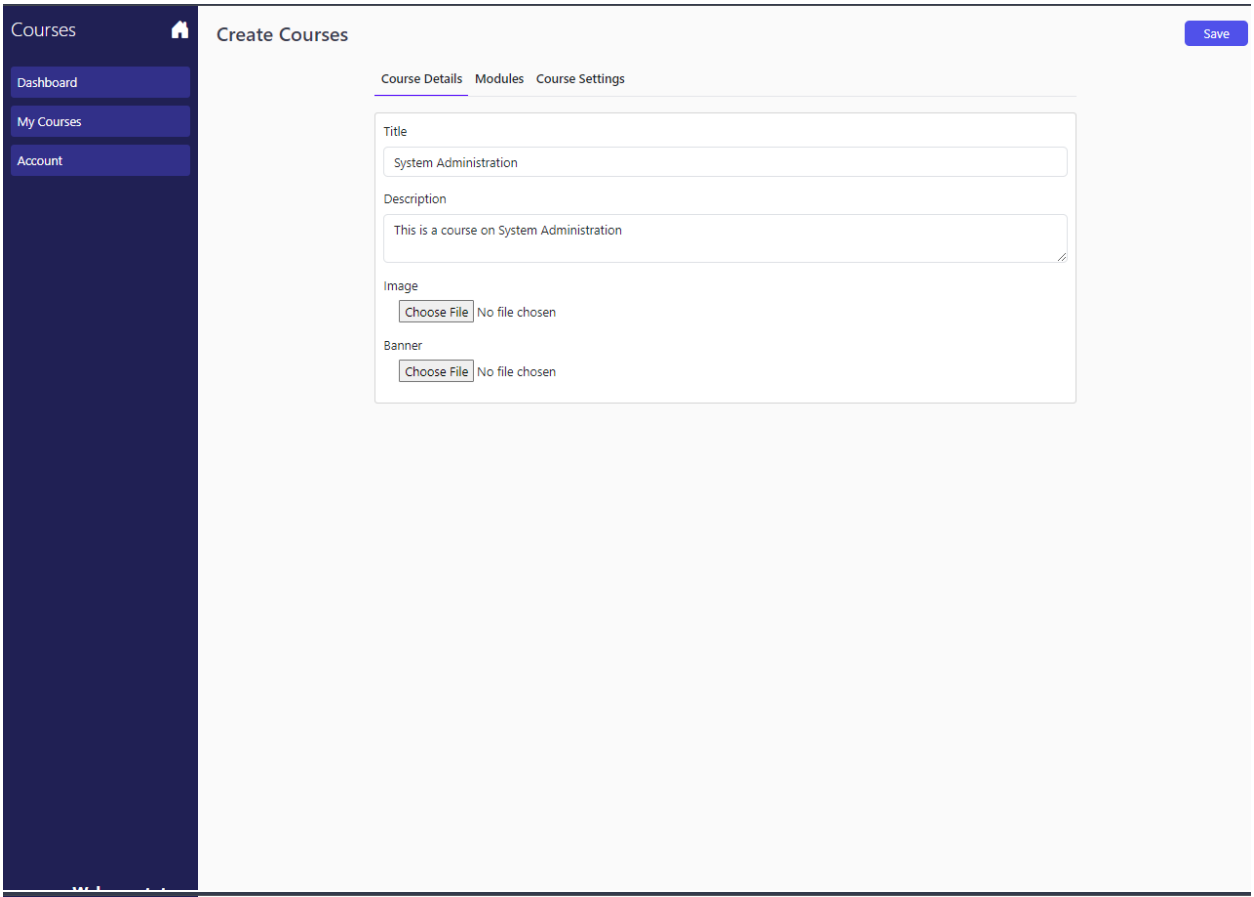
a)Consistency

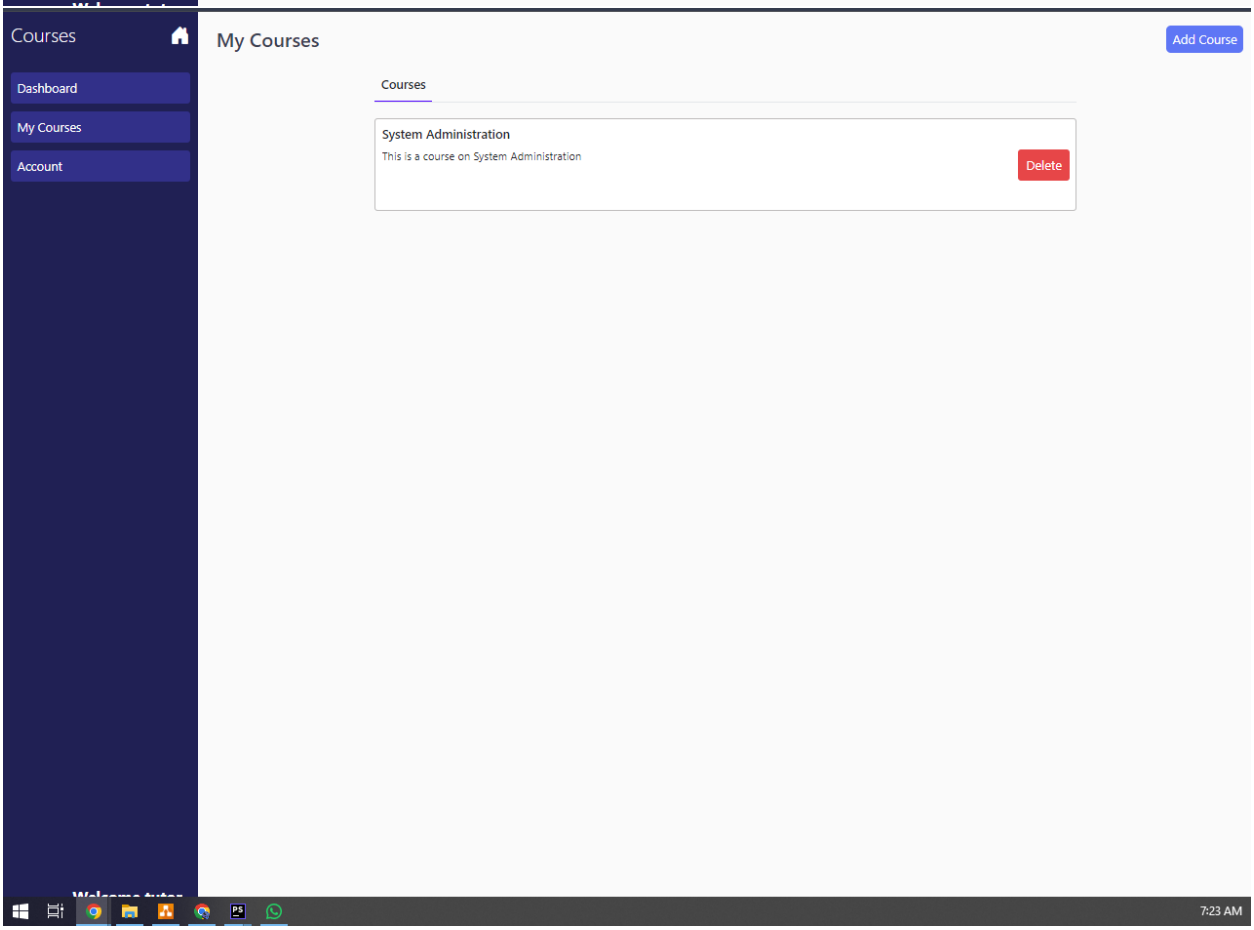
User Interface should be consistent throughout the system. The design layouts of different forms should be consistent over the whole application.

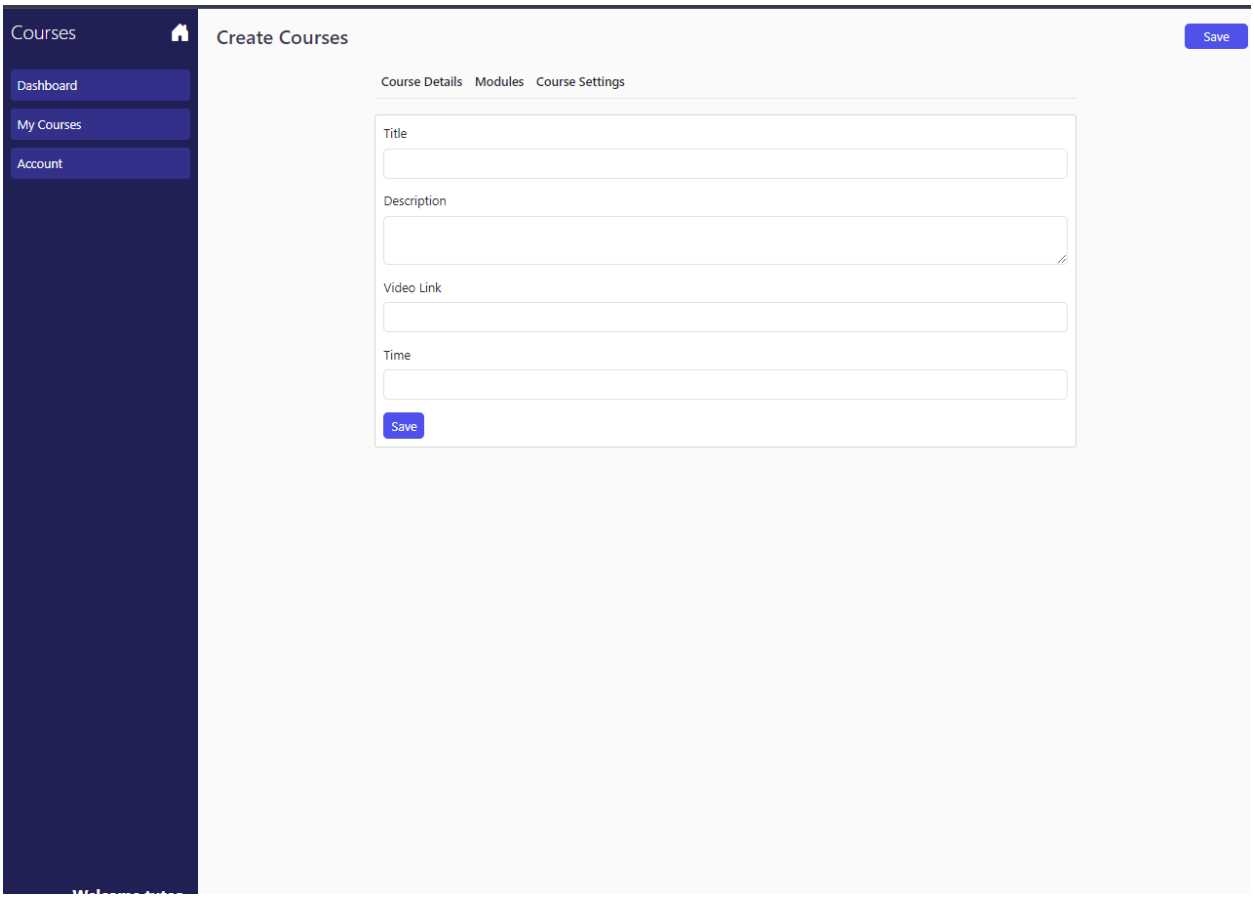
b)Prevent errors.

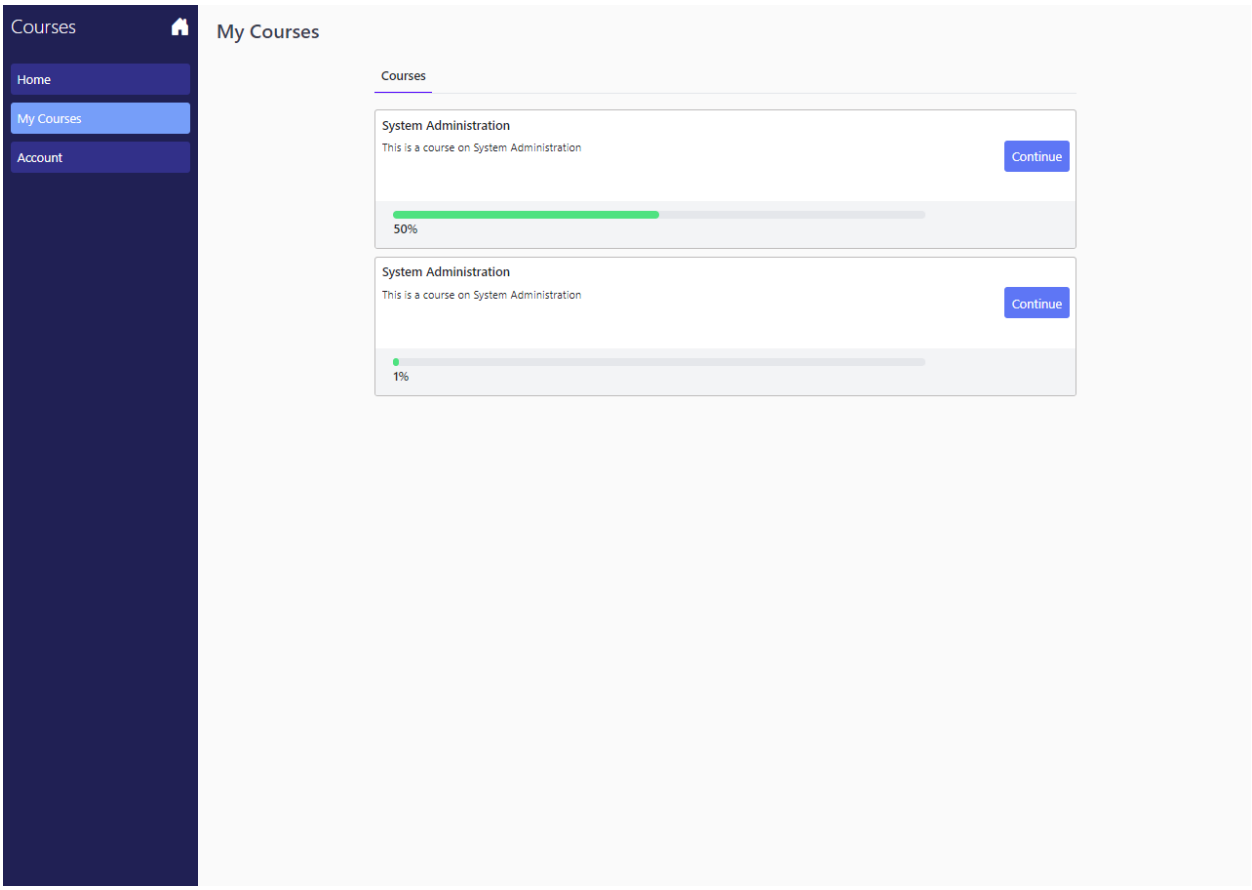
User interface should be designed in such way that the user cannot make any serious errors. Maximum consideration should be given to place checks for invalid inputs at the user interface




Courses
Dashboard
My Courses
Account
Create Courses
Save
Course Details
Modules
Course Settings
Title
System Administration
Description
This is a course on System Administration
Image
Choose File
No file chosen
Banner
Choose File
No file chosen


Courses
Dashboard
My Courses
Account
My Courses
Add Course
Courses
System Administration
This is a course on System Administration
Delete
7:23 AM


Courses
Dashboard
My Courses
Account
Create Courses
Save
Course Details
Modules
Course Settings
Title
Description
Video Link
Time
Save


Courses
Home
My Courses
Account
My Courses
Courses
System Administration
This is a course on System Administration
Continue
50%
System Administration
This is a course on System Administration
Continue
1%

6.3 Tools Used

The implementation of our LMS system adhered closely to the requirements outlined in the system specification, utilizing two key development tools: Visual Studio Code (VS Code) for frontend coding and MariaDB for database development.

Visual Studio Code (VS Code):

As the primary Integrated Development Environment (IDE) for frontend development, VS Code provided a versatile and efficient platform for coding user interface components, implementing business logic, and managing project files. Its rich feature set, including syntax highlighting, code completion, and integrated debugging, empowered developers to write clean, maintainable code that aligned with the system requirements. Additionally, VS Code's extensive ecosystem of extensions facilitated seamless integration with frontend frameworks such as Vue.js, enabling rapid development and iteration of user interface features.

MariaDB:

For database development, MariaDB was chosen as the backend solution for storing and managing the LMS data. Leveraging the power of MariaDB's relational database management system, developers designed and implemented the database schema in accordance with the system requirements. This involved creating tables, defining relationships, and optimizing queries to ensure efficient data storage and retrieval. With its support for SQL queries and transactions, MariaDB provided a robust foundation for the LMS's data management needs, enabling secure and scalable storage of user profiles, course materials, enrollment records, and other essential data.

By leveraging Visual Studio Code for frontend coding and MariaDB for database development, our development team effectively translated the system specification into a functional LMS system that met the needs of our users and stakeholders. These tools facilitated collaborative development, streamlined workflows, and ensured the timely delivery of high-quality software that met the specified requirements."

Programming Language

PHP: As the core programming language, PHP provided the foundation for implementing the backend logic and functionality of our LMS system. Leveraging PHP's versatility and robustness, developers coded features such as user authentication, course management, and content delivery, in accordance with the system requirements. PHP's extensive ecosystem of libraries and frameworks, combined with its ease of integration with web technologies, facilitated rapid development and customization of the LMS functionality to meet the diverse needs of our users.

Laravel Framework: Building upon PHP, the Laravel framework served as the cornerstone for structuring and organizing the backend codebase of our LMS system. Laravel's elegant syntax, expressive ORM, and built-in features, such as routing, middleware, and authentication, accelerated the development process and promoted code maintainability and scalability. By adhering to Laravel's conventions and best practices, developers ensured consistency and coherence in the codebase, facilitating collaboration and future enhancements.

6.4 Testing

In ensuring the robustness and reliability of our LMS system, rigorous testing was conducted using Pest, a contemporary PHP testing framework, alongside other essential tools. Pest was employed for its simplicity, readability, and expressive syntax, enabling developers to write clear and concise tests that accurately verified the functionality of the system.

Pest: As the primary testing framework, Pest provided a modern and developer-friendly approach to writing and executing tests for our LMS system. With its fluent and intuitive syntax, Pest allowed developers to focus on describing the behavior of the system in a natural language style, enhancing test readability and maintainability. Pest's emphasis on simplicity and convention over configuration streamlined the testing process, enabling developers to write effective tests with minimal overhead.

PHPUnit: Complementing Pest, PHPUnit, a robust and widely-used testing framework for PHP, was utilized for more traditional unit testing scenarios within our LMS system. PHPUnit's extensive feature set, including assertion methods, test fixtures, and data providers, facilitated comprehensive unit testing of individual components and functions, ensuring their correctness and reliability. By leveraging PHPUnit alongside Pest, our development team achieved a balanced testing strategy that covered both high-level behavioral tests and low-level unit tests, mitigating the risk of regressions and defects.

Visual Studio Code (VS Code) with Pest Extension: To streamline the testing workflow, Visual Studio Code (VS Code) was augmented with the Pest extension, which provided seamless integration with the Pest testing framework. The Pest extension for VS Code offered features such as test discovery, code navigation, and inline test results, empowering developers to write, run, and debug tests directly within their IDE. This tight integration facilitated a more efficient and productive testing experience, enabling developers to iterate quickly and confidently validate the behavior of the LMS system.

6.4.1 Testing Techniques

Testing techniques play a critical role in ensuring the reliability, functionality, and performance of our LMS system. The testing process encompasses various methodologies, including unit testing, integration testing, validation testing, and system testing, each serving distinct purposes in validating different aspects of the system's behavior.

Unit Testing:

Unit testing involves testing individual units or components of the system in isolation to verify their correctness and functionality. In the context of our LMS system, unit tests are used to validate the behavior of specific functions, methods, or classes within the codebase. Unit tests typically focus on testing small, atomic units of code, such as individual functions or methods, using mock objects or stubs to isolate dependencies and ensure deterministic test outcomes. By isolating units of code and testing them independently, unit testing helps identify bugs and defects early in the development process, enabling developers to fix issues promptly and maintain code quality.

Integration Testing:

Integration testing verifies the interactions and integration points between different components or modules of the system to ensure they function correctly together as a cohesive unit. In the context of our LMS system, integration tests are used to validate the interactions between various backend components, such as controllers, services, and data access layers, as well as the integration of frontend components with backend APIs. Integration tests typically involve testing the end-to-end flow of data

and functionality through the system, simulating real-world scenarios and identifying potential issues with component interactions or data exchange. By validating the integration of system components, integration testing helps uncover integration-related bugs and ensures the overall stability and reliability of the system.

Validation Testing:

Validation testing focuses on verifying that the system meets the specified requirements and fulfills the needs of its stakeholders. In the context of our LMS system, validation tests are used to validate the correctness, completeness, and usability of the system from the perspective of end users and administrators. Validation tests may include functional testing, usability testing, and acceptance testing, which assess the system's functionality, user interface, and conformance to user expectations, respectively. By validating the system against predefined criteria and user expectations, validation testing helps ensure that the LMS system meets the needs of its intended users and stakeholders, delivering a high-quality and user-friendly educational platform.

System Testing:

System testing evaluates the behavior and performance of the entire system as a whole, including all integrated components and subsystems, under various conditions and scenarios. In the context of our LMS system, system tests are used to validate the overall functionality, performance, and reliability of the system across different use cases and environments. System tests may include end-to-end testing, performance testing, security testing, and scalability testing, which assess the system's behavior under normal and stress conditions, as well as its compliance with security and performance requirements. By evaluating the system as a whole, system testing helps identify any discrepancies or issues that may arise from the interaction of individual components or subsystems, ensuring the overall quality and integrity of the LMS system.

By employing a combination of unit testing, integration testing, validation testing, and system testing techniques, our development team ensures the robustness, reliability, and functionality of our LMS system, delivering a high-quality educational platform that meets the needs of our users and stakeholders."

6.4.2 Training and Handover

Training and handover are essential aspects of the testing process, ensuring that stakeholders are equipped with the necessary knowledge and resources to effectively use and maintain the LMS system. This phase involves two key components: training and handover.

Training:

Training involves providing stakeholders, including administrators, educators, and users, with comprehensive instruction and guidance on how to use the LMS system effectively. Training sessions may cover a wide range of topics, including system navigation, user management, content creation, course enrollment, assessment creation, and reporting functionalities. Training sessions can be conducted through various mediums, such as in-person workshops, online tutorials, documentation guides, and video tutorials, tailored to the specific needs and preferences of the target audience. The goal of training is to empower stakeholders with the knowledge and skills necessary to leverage the full capabilities of the LMS system, enabling them to perform their roles efficiently and effectively.

Handover:

Handover involves transferring ownership and responsibility for the LMS system from the development team to the designated stakeholders, typically administrators or system administrators. During the handover process, the development team provides stakeholders with all relevant documentation, resources, and support materials, including user manuals, technical specifications, system architecture diagrams, and contact information for technical support. Additionally, any outstanding issues, bugs, or feature requests identified during the testing phase are documented and communicated to the stakeholders for resolution. The handover process may also include a formal sign-off or acceptance procedure, where stakeholders acknowledge receipt of the system and confirm their readiness to assume responsibility for its operation and maintenance.

By conducting thorough training sessions and facilitating a smooth handover process, our development team ensures that stakeholders are well-prepared to leverage the LMS system to its fullest potential. Training and handover activities promote user adoption, system acceptance, and long-term sustainability, laying the groundwork for a successful deployment and ongoing operation of the LMS system within the organization."

CHAPTER 7

CONCLUSION AND RECOMMENDATION

7.0 Introduction

This chapter serves as a culmination of the discussions presented in the preceding chapters, consolidating the conclusions and recommendations derived from the project's objectives and scope. By synthesizing the insights gained from the various chapters, we aim to provide a comprehensive overview of the key findings and actionable insights that guide the development and implementation of the Learning Management System (LMS). This chapter outlines the project's objectives, summarizes the conclusions drawn from the analysis, and presents actionable recommendations for optimizing the LMS system's effectiveness and efficiency in meeting the organization's educational objectives.

7.1 Conclusion

Through the exploration of the project's objectives and scope, coupled with an in-depth analysis of the system's design, implementation, and testing phases, several key conclusions have emerged.

Firstly, the development and implementation of the Learning Management System (LMS) have been successfully aligned with the project's objectives. The system has been designed to facilitate efficient registration and login processes for both employees and tutors, leveraging technologies such as Vue.js for interface implementation and MariaDB for database management. Furthermore, the incorporation of testing techniques, including unit testing, integration testing, validation testing, and system testing, has ensured the reliability, functionality, and usability of the system.

Secondly, the LMS system demonstrates scalability and flexibility, enabling seamless integration of new features and functionalities to accommodate future growth and evolving requirements. By leveraging cloud-based infrastructure and scalable architectures, such as AWS, the system can dynamically scale resources based on demand, optimizing cost-effectiveness and performance.

Lastly, user training and support play a crucial role in maximizing the LMS system's effectiveness and user adoption. Providing comprehensive training materials, onboarding sessions, and dedicated support

channels ensures that users and administrators can effectively utilize the system's features and functionalities, thereby enhancing user satisfaction and usability.

In conclusion, the development and implementation of the LMS system have been guided by a clear understanding of the project's objectives and scope. By incorporating best practices in system design, testing, scalability, and user support, the LMS system is well-positioned to support the organization's educational initiatives and facilitate a seamless learning experience for users.

7.2 Recommendation

Building upon the conclusions drawn from the project's analysis and implementation phases, several recommendations emerge to further enhance the effectiveness and efficiency of the Learning Management System (LMS).

Continuous Improvement: Implement a process for continuous improvement, including regular evaluation of system performance, user feedback analysis, and iteration on features and functionalities. By soliciting and incorporating feedback from users and stakeholders, the LMS system can evolve to better meet the needs and expectations of its users.

Accessibility and Inclusivity: Prioritize accessibility and inclusivity in the design and development of the LMS system to ensure that all users, regardless of ability or background, can access and benefit from its features. Consideration should be given to factors such as screen reader compatibility, keyboard navigation, and support for diverse languages and cultural contexts.

Data Security and Privacy: Strengthen data security and privacy measures to safeguard sensitive user information and ensure compliance with relevant regulations, such as GDPR or CCPA. Implement robust authentication mechanisms, encryption protocols, and access controls to protect data from unauthorized access or breaches.

Performance Optimization: Continuously monitor and optimize the performance of the LMS system to ensure fast response times, minimal downtime, and efficient resource utilization. Employ techniques such as caching, load balancing, and query optimization to enhance system scalability and responsiveness, particularly during peak usage periods.

Stakeholder Engagement: Foster ongoing engagement and collaboration with stakeholders, including educators, administrators, and IT personnel, to ensure alignment with organizational goals and priorities. Regular communication, feedback sessions, and stakeholder meetings can help identify emerging needs and opportunities for improvement.

By implementing these recommendations, the organization can further enhance the value and impact of the LMS system, empowering users to engage in effective learning experiences and achieve their educational goals.

REFERENCES

APPENDICES

APPENDIX I: BUDGET

APPENDIX II: SCHEDULE

APPENDIX III: QUESTIONNAIRE

APPENDIX IV: CODE LISTINGS