**ASSIGNMENT FINAL REPORT**

|  |  |  |  |
| --- | --- | --- | --- |
| **Qualification** | **Pearson BTEC Level 5 Higher National Diploma in Computing** | | |
| **Unit number and title** | **Unit 7: Software Development Life Cycle** | | |
| **Submission date** | 15/08/2025 | **Date Received 1st submission** | 15/08/2025 |
| **Re-submission Date** |  | **Date Received 2nd submission** |  |
| **Student Name** | Nguyen Viet Tien | **Student ID** | BC00361 |
| **Class** | SE07201 | **Assessor name** | Tran Van Nhuom |
| **Plagiarism**  Plagiarism is a particular form of cheating. Plagiarism must be avoided at all costs and students who break the rules, however innocently, may be penalised. It is your responsibility to ensure that you understand correct referencing practices. As a university level student, you are expected to use appropriate references throughout and keep carefully detailed notes of all your sources of materials for material you have used in your work, including any material downloaded from the Internet. Please consult the relevant unit lecturer or your course tutor if you need any further advice.  **Student Declaration**  I certify that the assignment submission is entirely my own work and I fully understand the consequences of plagiarism. I declare that the work submitted for assessment has been carried out without assistance other than that which is acceptable according to the rules of the specification. I certify I have clearly referenced any sources and any artificial intelligence (AI) tools used in the work. I understand that making a false declaration is a form of malpractice. | | | |
|  |  | **Student’s signature** | Remove-bg.ai_1716972298959 |

**Grading grid**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **P1** | **P2** | **P3** | **P4** | **P5** | **P6** | **P7** | **M1** | **M2** | **M3** | **M4** | **M5** | **M6** | **D1** | **D2** | **D3** | **D4** |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

|  |  |  |
| --- | --- | --- |
| ** Summative Feedback:  Resubmission Feedback:** | | |
| **Grade:** | **Assessor Signature:Tran Van Nhuom** | **Date:18/08/2025** |
| **Internal Verifier’s Comments:** | | |
| **Signature & Date:** | | |

**TABLE OF CONTENTS**

[List of tables v](#_Toc30710)

[Introduction 1](#_Toc19959)

[Contents 2](#_Toc24826)

[1.1. Describe two iterative and two sequential software lifecycle models (P1) 2](#_Toc7568)

[1.1.1. Iterative Lifecycle Models 2](#_Toc10461)

[1.1.1.1. Agile Model 2](#_Toc7668)

[1.1.1.2. Spiral Model 2](#_Toc11048)

[1.1.2. Sequential Lifecycle Models 3](#_Toc27095)

[1.1.2.1. Waterfall Model 3](#_Toc14297)

[1.1.2.2. V-Model (Verification and Validation Model) 4](#_Toc14544)

[1.2. Explain how risk is managed in software lifecycle models (P2) 4](#_Toc6497)

[1.2.1. Risks in Iterative Models 5](#_Toc8191)

[1.2.1.1. Agile Model Risks 5](#_Toc23329)

[1.2.1.2. Spiral Model Risks 5](#_Toc9462)

[1.2.2. Risks in Sequential Models 5](#_Toc23862)

[1.2.2.1. Waterfall Model Risks 5](#_Toc7404)

[1.2.2.2. V-Model Risks 6](#_Toc26461)

[1.2.3. General Risk Identification and Monitoring Techniques 6](#_Toc25214)

[1.3. Discuss using an example, why a particular lifecycle model is selected for a development environment (M1) 6](#_Toc15407)

[1.3.1. Benefits of the Agile Model 6](#_Toc32398)

[1.3.2. Drawbacks of the Agile Model 7](#_Toc30474)

[1.3.3. Why Agile is Appropriate for the LMS Project 7](#_Toc1278)

[1.4. Assess the merits of applying the Waterfall lifecycle model to a large software development project (D1) 7](#_Toc4696)

[1.4.1. Merits of the Waterfall Model 7](#_Toc11352)

[1.4.1.1. Stability and Predictability 7](#_Toc8479)

[1.4.1.2. Better Resource Planning 8](#_Toc8099)

[1.4.1.3. Clear Documentation 8](#_Toc24060)

[1.4.2. Limitations in Large and Complex Projects 8](#_Toc17396)

[1.4.2.1. Inflexibility with Requirement Changes 8](#_Toc753)

[1.4.2.2. Late Testing 8](#_Toc14208)

[1.4.3. Quantitative Comparison. 8](#_Toc14513)

[1.5. Explain the purpose of a feasibility report (P3) 10](#_Toc15925)

[1.5.1. Definition and General Purpose 10](#_Toc8532)

[1.5.2. Reducing Risks 10](#_Toc12325)

[1.5.3. Ensuring Project Viability 10](#_Toc24980)

[1.5.4. Supporting Decision-Making 10](#_Toc24653)

[1.6. Describe how technical solutions can be compared (P4) 11](#_Toc25859)

[1.6.1. Purpose of Comparing Technical Solutions 11](#_Toc26228)

[1.6.2. Common Comparison Methods 11](#_Toc15574)

[1.6.2.1. Cost-Benefit Analysis 11](#_Toc19776)

[1.6.2.2. Decision Matrix 11](#_Toc13890)

[1.6.3. Apply a Decision Matrix to Select the Most Appropriate SDLC Model 12](#_Toc28080)

[1.6.4. Realistic Application in the LMS Project 14](#_Toc27768)

[1.7. Discuss the components of a feasibility report (M2) 14](#_Toc19352)

[1.7.1. Technical Feasibility 14](#_Toc31681)

[1.7.2. Economic Feasibility 15](#_Toc8919)

[1.7.3. Legal Feasibility 15](#_Toc26194)

[1.7.4. Operational Feasibility 15](#_Toc4955)

[1.7.5. Schedule Feasibility 15](#_Toc196)

[1.8. Assess the impact of different feasibility criteria on a software investigation (D2) 16](#_Toc2610)

[1.8.1. Technical Feasibility 16](#_Toc11098)

[1.8.2. Economic Feasibility 16](#_Toc14394)

[1.8.3. Legal Feasibility 16](#_Toc1665)

[1.8.4. Operational Feasibility 16](#_Toc11050)

[1.8.5. Schedule Feasibility 17](#_Toc1059)

[1.9. Executive Summary 17](#_Toc1035)

[1.10. Project Introduction 17](#_Toc1225)

[1.10.1. Purpose 17](#_Toc24157)

[1.10.2. Project Overview 18](#_Toc26276)

[1.10.3. Scope 18](#_Toc21881)

[1.11. Undertake a software investigation to meet a business need (P5) 18](#_Toc24043)

[1.11.1. Prototype Implementation 20](#_Toc16881)

[1.12. Use appropriate software analysis tools/techniques to carry out a software investigation and create supporting documentation (P6) 24](#_Toc26939)

[1.12.1. Context Diagram 24](#_Toc22266)

[1.12.2. Data Flow Diagram (DFD) Level 0 25](#_Toc2518)

[1.12.3. Entity Relationship Diagram (ERD) 26](#_Toc18472)

[1.12.4. Business System Options (BSOs) 28](#_Toc26354)

[1.12.5. Technical System Options (TSOs) 29](#_Toc8706)

[1.13. Analyse how software requirements can be traced throughout the software lifecycle (M3) 30](#_Toc31869)

[1.13.1. Traceability Matrix 30](#_Toc5173)

[1.13.2. Lifecycle Tracking 31](#_Toc14592)

[1.13.3. Change Management 31](#_Toc8823)

[1.14. Discuss two approaches to improving software quality (M4) 32](#_Toc19328)

[1.14.1. Manual Testing 32](#_Toc6479)

[1.14.2. Peer Code Review 33](#_Toc10643)

[1.15. Evaluate the process of undertaking a systems investigation with regard to its effectiveness in improving a software quality (D3) 34](#_Toc20068)

[1.15.1. Strengths 34](#_Toc22280)

[1.15.2. Weaknesses 35](#_Toc12291)

[1.15.3. Recommendations 35](#_Toc972)

[1.16. Discuss, using examples, the suitability of software behavioural design techniques (P7) 35](#_Toc1521)

[1.16.1. Examples of Techniques 35](#_Toc31218)

[1.16.2. Pseudocode for defining algorithms or logic flows 37](#_Toc5582)

[1.16.3. Finite State Machines (FSMs) for Modeling System States and Transitions 39](#_Toc25867)

[1.16.4. Summary of Benefits 39](#_Toc12179)

[1.17. Analyse a range of software behavioural tools and techniques (M5) 40](#_Toc1329)

[1.17.1. Range of Tools 40](#_Toc2795)

[State/Data-Driven Approaches 40](#_Toc13166)

[Formal Specification Methods 41](#_Toc25728)

[1.17.2. Strengths and Limitations 41](#_Toc2669)

[1.18. Differentiate between a finite state machine (FSM) and an extended FSM, providing an application for both (M4) 43](#_Toc618)

[1.18.1. Key Differences 43](#_Toc23956)

[1.18.2. Application Examples in the LMS Project 43](#_Toc16742)

[1.19. Present justifications of how data driven software can improve the reliability and effectiveness of software (D4) 45](#_Toc14832)

[1.19.1. Reliability and Effectiveness 45](#_Toc6274)

[1.19.2. Implementation Strategies 46](#_Toc18343)

[Conclusion 48](#_Toc12124)

[References 49](#_Toc22467)

# List of tables

[Table 1 : Comparison of SDLC Models for Learning Management System (LMS) Projects 4](#_Toc2457)

[Table 2 : Comparison of Waterfall and Agile Models for LMS Development 8](#_Toc3092)

[Table 3 : Quantitative Comparison with Agile and V-Model 8](#_Toc25261)

[Table 4 : Decision Matrix Comparing Custom LMS vs Moodle 11](#_Toc4827)

[Table 5 : Decision Matrix for SDLC Selection 12](#_Toc1594)

[Table 6 : Requirements Traceability Matrix for LMS Prototype 30](#_Toc3774)

[Table 7 : Manual Test Cases and Results for LMS Prototype 32](#_Toc9644)

**LIST OF FIGURE**

[Figure 1 : Agile Model 2](#_Toc3089)

[Figure 2 : Spiral Model 3](#_Toc8386)

[Figure 3 : Waterfall Model 3](#_Toc16208)

[Figure 4 : V-Model 4](#_Toc16317)

[Figure 5 : Radar chart comparing Waterfall, Agile, and V-Model across five evaluation criteria: Development Speed, Error Detection Speed, Cost Control, Scalability, and Stakeholder Involvement, based on quantitative scores from Table 3. 9](#_Toc14129)

[Figure 6 : Radar chart visualizing the Decision Matrix scores for Waterfall, Agile, and V-Model. 13](#_Toc26915)

[Figure 7 : Agile sprint cycle for the LMS project showing backlog, planning, development & testing, review, retrospective, and continuous delivery of course and enrollment features. 14](#_Toc789)

[Figure 8 : Login System (Authentication Module) 20](#_Toc32498)

[Figure 9 : 2.Course Creation (Course Management Module) 22](#_Toc10632)

[Figure 10 : Enrollment (Course Management Module) 23](#_Toc30142)

[Figure 11 : Context Diagram of the LMS 24](#_Toc19049)

[Figure 12 : Data Flow Diagram (DFD) Level 0 of the LMS 25](#_Toc10232)

[Figure 13 : Entity Relationship Diagram (ERD) of the LMS 26](#_Toc11340)

[Figure 14 : Business System Options (BSOs) of the LMS 28](#_Toc28272)

[Figure 15 : Technical System Options (TSOs) of the LMS 29](#_Toc14620)

[Figure 16 : Login process flowchart 36](#_Toc8729)

[Figure 17 : Student course enrollment flowchart 36](#_Toc11781)

[Figure 18 : Instructor course creation flowchart 37](#_Toc27404)

[Figure 19 : Login Pseudocode 37](#_Toc656)

[Figure 20 : Course Enrollment Pseudocode 38](#_Toc208)

[Figure 21 : Course Creation Pseudocode 38](#_Toc19216)

[Figure 22 : Finite State Machine for user roles and navigation 39](#_Toc16345)

[Figure 23 : FMS: Login Process 44](#_Toc6718)

[Figure 24 : EFSM: Course Enrollment (with userRole and isEnrolled) 45](#_Toc14414)

[Figure 25 : 3 Layer Architecture of LMS 46](#_Toc9760)

[Figure 26 : Data Flow Diagram for LMS Processes 47](#_Toc6672)

[Figure 27 : ER Diagram (Core Tables) 47](#_Toc13359)

# Introduction

The rapid advancement of information technology has transformed how educational institutions deliver and manage learning activities. Learning Management Systems (LMS) have become essential platforms for universities, enabling the administration, documentation, tracking, and delivery of academic content in an organized and interactive manner. In a university environment, an LMS not only serves as a repository for course materials but also acts as a central hub for communication, assessments, and progress tracking for both lecturers and students. The design and implementation of such a system require careful planning to ensure it aligns with academic requirements, institutional policies, and technological infrastructure.

This report examines the selection and comparison of technical solutions and Software Development Life Cycle (SDLC) models for the development of a customized LMS. It evaluates various methodologies, including Agile, Waterfall, V-Model, and Spiral, to determine the most suitable approach for the project’s needs. Additionally, the report applies analytical tools such as cost-benefit analysis and decision matrices to compare different development strategies, ensuring that the chosen solution maximizes performance, scalability, and stakeholder satisfaction. By incorporating realistic application examples from the LMS project, the study emphasizes the practical relevance of these methods in delivering a system that meets both functional and user experience expectations.

# Contents

## Describe two iterative and two sequential software lifecycle models (P1)

As a Software Development Consultant at InnovateTech Solutions, I am responsible for preparing a Software Development Plan (SDP) for a university Learning Management System (LMS). Selecting the right software development lifecycle (SDLC) model is essential to ensure the system is developed on time, meets user needs, and stays within budget. In this section, I will explain two iterative models (Agile and Spiral) and two sequential models (Waterfall and V-Model), describing their stages and suitability for different projects.

### Iterative Lifecycle Models

#### Agile Model

The Agile model is a flexible way of developing software. The project is divided into short periods called sprints, usually 2–4 weeks long. In each sprint, the team plans, develops, tests, and reviews the product. At the end of each sprint, the team shows the results to users and collects feedback. This feedback is used to improve the product in the next sprint. In the LMS project, Agile is useful because features like course pages, student forums, and assessment tools can be updated and improved based on feedback from teachers and students.

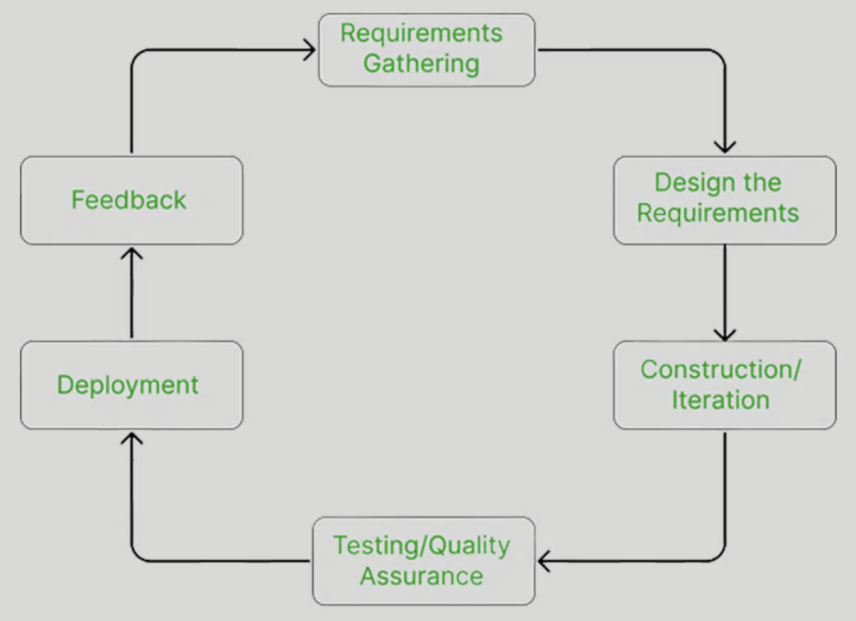


Figure 1: Agile Model

#### Spiral Model

The Spiral model combines repeated development with risk control. Each round (called a spiral) has four steps: planning, risk analysis, development, and review. This model works well for big, complex systems with many unknown points at the start. In the LMS project, Spiral helps manage risks such as connecting with the university’s database and login system, or meeting legal rules for student data privacy. The team reviews and updates the plan after each round, making it easier to handle changes.

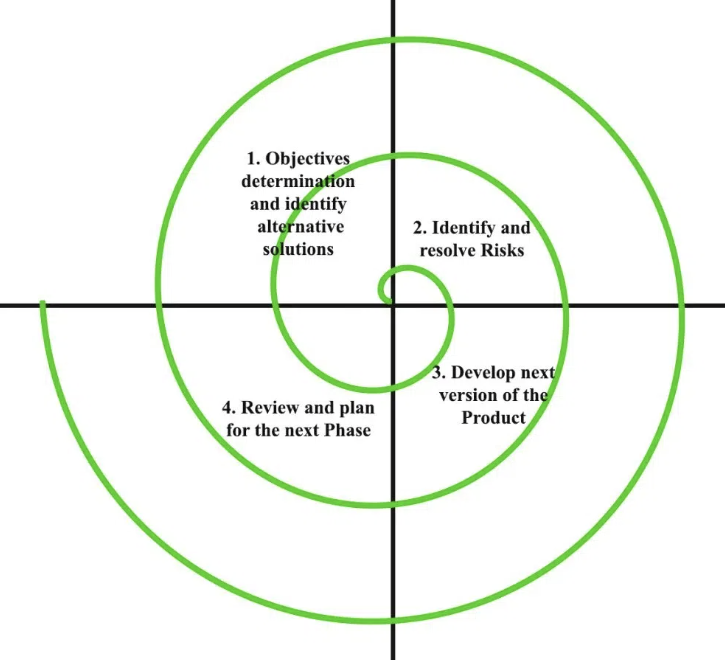


Figure 2: Spiral Model

### Sequential Lifecycle Models

#### Waterfall Model

The Waterfall model is a step-by-step process. It has clear stages like requirements, design, coding, testing, and maintenance. Each stage must be finished before the next begins. It works best when all requirements are known from the start. In the LMS project, if the university already has full details of the system and no big changes are expected, Waterfall is a good choice because it is well-structured and easy to manage.

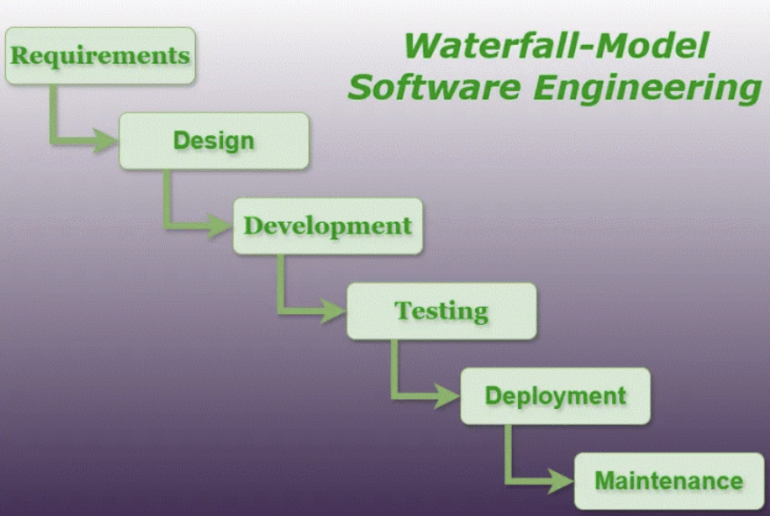


Figure 3: Waterfall Model

#### V-Model (Verification and Validation Model)

The V-Model is like Waterfall but adds testing for each development stage. For example, requirements are linked to acceptance testing, and system design is linked to system testing. This helps find problems earlier. In the LMS project, the V-Model is good when quality and security are very important, such as keeping student records safe and making sure tests and assignments work correctly.

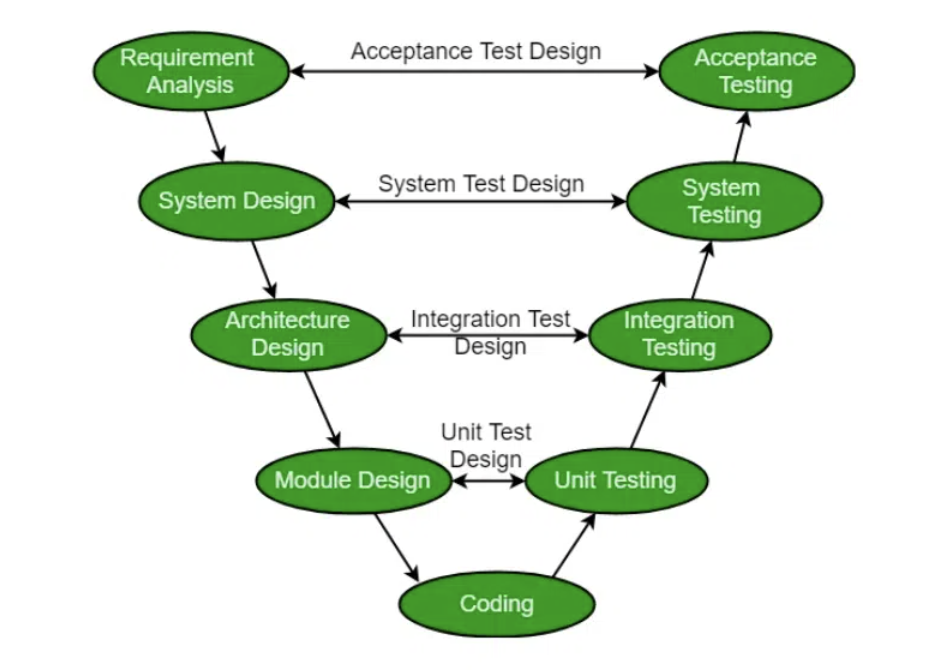


Figure 4: V-Model

Table 1: Comparison of SDLC Models for Learning Management System (LMS) Projects

|  |  |  |  |
| --- | --- | --- | --- |
| **Model** | **Best For** | **Strengths** | **Weaknesses** |
| Agile | Projects with changing educational needs | Flexible, user-centered, fast feedback | Needs strong collaboration |
| Spiral | Complex LMS with high risk | Strong risk control, adaptable | More costly and time-consuming |
| Waterfall | Fixed LMS requirements with low change | Easy to manage, clear structure | Hard to change requirements later |
| V-Model | High-quality LMS with strong testing needs | Early testing, high reliability | Long process, less flexibility |

## Explain how risk is managed in software lifecycle models (P2)

Every software development model includes some level of risk. For the Learning Management System (LMS) project, identifying and managing risks early can help prevent delays, cost overruns, and system failures. In this section, I will explain common types of risks in the Agile, Spiral, Waterfall, and V-Model lifecycles. I will also suggest strategies for identifying and reducing these risks during development.

### Risks in Iterative Models

#### Agile Model Risks

Agile is flexible, but it can face process risks like scope creep—when the client keeps asking for new features during development. This can delay the project or increase costs. Resource risks may also happen if team members are not experienced in Agile or if communication between the team and client is weak.

**Risk Management Strategies:**

* Use a clear product backlog and set priorities with the client.
* Hold regular sprint planning and review meetings.
* Assign a Scrum Master to monitor team progress and remove obstacles.

#### Spiral Model Risks

The Spiral model focuses on risk management, but it also brings resource risks like high cost and time consumption. Each spiral requires detailed planning and documentation. Technical risks may appear if the system becomes too complex with many changing parts (e.g., LMS integrating grading, forums, and cloud storage).

**Risk Management Strategies:**

* Identify major risks at the start of each spiral.
* Use risk assessment tools like risk registers or risk matrices.
* Involve technical experts early to evaluate possible problems.

### Risks in Sequential Models

#### Waterfall Model Risks

In Waterfall, the biggest process risk is that it’s difficult to make changes after the design phase. If a mistake in the LMS requirements is found later during testing, it can be expensive and time-consuming to fix. There is also a technical risk that the system might not meet user expectations because there is no user feedback during development.

**Risk Management Strategies:**

* Spend more time on the requirement phase with full stakeholder interviews.
* Use prototyping tools to show basic designs early.
* Schedule testing after each main phase to detect problems early.

#### V-Model Risks

The V-Model also has resource risks because it requires detailed testing for each phase, which can take a lot of time and people. Process risks may occur if there is not enough documentation or if test cases are not written properly.

**Risk Management Strategies:**

* Assign a test team from the beginning of the project.
* Create test plans at the same time as development plans.
* Track each requirement and link it to a test case (traceability matrix).

### General Risk Identification and Monitoring Techniques

For all models, good risk management includes the following techniques:

* Risk Identification: Use brainstorming sessions, SWOT analysis, expert interviews, and checklists.
* Risk Mitigation Plans: Develop backup solutions, allocate buffer time, and prepare extra resources.
* Monitoring Processes: Conduct weekly progress reports, use tools like Jira for tracking, and hold regular meetings to update risk status.

## Discuss using an example, why a particular lifecycle model is selected for a development environment (M1)

After comparing different software development lifecycle (SDLC) models, I recommend using the Agile model for the Learning Management System (LMS) project. The LMS needs to be user-friendly, flexible, and continuously improved based on feedback from students, instructors, and administrators. Agile is the most suitable model for this kind of environment.

### Benefits of the Agile Model

One of the biggest benefits of Agile is flexibility. In the LMS project, users may request new features such as discussion forums, calendar tools, or personalized dashboards. With Agile, the team can deliver these features step by step in short development cycles (called sprints). After each sprint, the university staff and students can test the features and give feedback, allowing the team to make changes quickly.

Another benefit is better communication. Agile encourages frequent meetings between developers, testers, and clients. This helps to avoid misunderstandings and allows quick decision-making. In an educational setting, where user needs often change, this approach supports faster responses and better results.

Agile also supports early delivery. The team can release the LMS core features (such as course listings, user login, and file uploads) in the first few sprints. This allows the university to start using the system earlier, while the team continues improving it with new tools like assignment modules or performance dashboards.

### Drawbacks of the Agile Model

However, Agile also has some drawbacks. It requires a high level of team collaboration and frequent client involvement. If the university representatives are not available for regular reviews, the development process may slow down. Also, because Agile does not include full documentation at the beginning, it can be harder to train new developers if they join mid-project.

Another issue is scope creep. Since Agile allows frequent changes, there is a risk that too many new features are added during the project, which could delay delivery or increase costs.

### Why Agile is Appropriate for the LMS Project

Despite these challenges, the benefits of Agile make it the best option for the LMS project. The education environment is dynamic. Students and teachers often suggest new ideas, and the platform must be updated regularly to stay useful and relevant. Agile allows the development team to listen to users, prioritize their needs, and update the system continuously.

Also, the university may want to launch a minimum viable product (MVP) first — for example, just the course upload and user login features — and then expand it later with tools for grading, analytics, or communication. Agile supports this kind of strategy better than traditional models like Waterfall or V-Model.

## Assess the merits of applying the Waterfall lifecycle model to a large software development project (D1)

The Waterfall model is a sequential software development lifecycle (SDLC) model that follows a strict top-down approach. Each phase (requirements, design, implementation, testing, deployment) must be completed before moving to the next one (Sommerville, 2016). This section assesses the value of using Waterfall in large-scale projects like a university-wide Learning Management System (LMS).

### Merits of the Waterfall Model

#### Stability and Predictability

Waterfall offers high predictability because all requirements are gathered at the start. For a university LMS, if the institution has well-defined learning processes and system expectations (e.g., course upload, quiz creation, reporting), Waterfall provides a clear roadmap. This helps avoid confusion during development.

#### Better Resource Planning

Since all tasks and deadlines are planned early, resources (developers, testers, budget) can be managed effectively. For large LMS projects with fixed schedules (e.g., ready by semester start), this is a benefit.

#### Clear Documentation

Waterfall requires extensive documentation. In a university context, this allows administrators, lecturers, and IT staff to review plans before development starts, reducing misunderstandings.

### Limitations in Large and Complex Projects

#### Inflexibility with Requirement Changes

If new educational features (e.g., AI-assisted grading, integration with Zoom) are requested later, Waterfall cannot adapt easily. This makes it risky for LMS projects where educational needs may change due to new policies or trends in digital learning.

#### Late Testing

Errors may not be discovered until the testing phase. In LMS systems, late discovery of bugs in features like quiz modules or grading reports can delay deployment, especially if the system is needed before semester starts.

Table 2: Comparison of Waterfall and Agile Models for LMS Development

|  |  |  |
| --- | --- | --- |
| **Criteria** | **Waterfall** | **Agile** |
| Stability | High | Medium |
| Requirement Flexibility | Low | High |
| Documentation | Heavy | Lightweight |
| Risk Detection | Late (during testing phase) | Early (continuous feedback) |
| LMS Suitability | For fixed, low-change projects | For dynamic, user-driven projects |

In the LMS context, Agile is better when university policies or user needs change frequently. Waterfall may still be useful for specific modules like user authentication or reporting, where requirements are stable.

### Quantitative Comparison.

The Waterfall model is a sequential software development process where each phase (Requirements → Design → Implementation → Testing → Deployment → Maintenance) is completed before moving to the next. This makes it predictable and easy to manage, but also less flexible compared to iterative models.

Table 3: Quantitative Comparison with Agile and V-Model

|  |  |  |  |
| --- | --- | --- | --- |
| **Criteria** | **Waterfall** | **Agile** | **V-Model** |
| Development Speed | 2/5 | 5/5 | 3/5 |
| Error Detection Speed | 3/5 | 5/5 | 4/5 |
| Cost Control | 5/5 | 3/5 | 4/5 |
| Scalability | 4/5 | 5/5 | 4/5 |
| Stakeholder Involvement | 2/5 | 5/5 | 3/5 |

* Waterfall scores high in cost control (5/5) because budgeting and resources are fixed early in the project.
* Agile scores highest in speed and adaptability because it delivers in sprints with continuous feedback.
* V-Model offers better error detection than Waterfall because testing is planned alongside development stages.

Figure 5 below visually illustrates those scores using a radar chart for easy comparison.

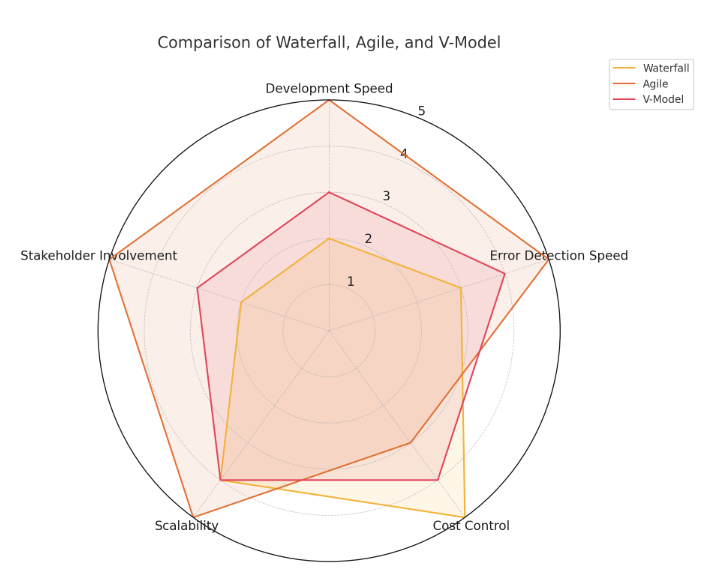


Figure 5: Radar chart comparing Waterfall, Agile, and V-Model across five evaluation criteria: Development Speed, Error Detection Speed, Cost Control, Scalability, and Stakeholder Involvement, based on quantitative scores from Table 3.

**Advantages of Waterfall for Large Projects**

1. **Predictability**: All requirements are set at the start, making scheduling easier.
2. **Strong Documentation:** Each phase produces clear deliverables, useful for compliance-heavy projects.
3. **Stable Scope:** Well-suited for projects where changes are rare.

**Disadvantages Compared to Agile & V-Model**

* **Lack of Flexibility:** If requirements change, it is costly and time-consuming to adapt.
* **Late Testing:** Bugs may only be found in the testing phase, delaying delivery.
* **Lower Stakeholder Interaction:** Less client feedback during development compared to Agile.

**When to Use Waterfall**

* Government or legal systems where documentation is critical.
* Large infrastructure projects with fixed budgets and deadlines.
* Systems with well-understood requirements.

## Explain the purpose of a feasibility report (P3)

### Definition and General Purpose

A feasibility study is an early step in planning a software project. It helps decide whether the project can be done successfully. For the LMS project, the feasibility study checks if the system is possible to build, useful for users, and realistic based on time, money, and resources.

### Reducing Risks

One key goal of a feasibility study is to reduce risks. In the LMS project, there are risks like system incompatibility, over-budget development, or unmet user needs. The feasibility study helps find these risks early so the team can solve them before development begins.

### Ensuring Project Viability

The study looks at technical, financial, legal, and operational aspects to make sure the LMS project is viable. **For example:**

* Technical: Can the LMS work with the university’s current systems?
* Financial: Is the budget enough for development and future updates?
* Legal: Will the system follow rules like GDPR for student data privacy?
* Operational: Does the LMS match the university’s teaching goals?

### Supporting Decision-Making

The feasibility study gives project leaders clear facts to help them make smart decisions. If the study shows high cost or big risks, the university may change the plan or find another solution. If the results are positive, the project can continue with confidence.

## Describe how technical solutions can be compared (P4)

### Purpose of Comparing Technical Solutions

Before starting development, it is important to compare different technical solutions to choose the best one for the LMS project. This process helps the team understand the strengths, weaknesses, costs, and benefits of each option. A good comparison saves time and money, and improves system performance and user satisfaction.

### Common Comparison Methods

#### Cost-Benefit Analysis

This method compares the total cost of each solution with the expected benefits. For example, one LMS platform might cost more but offer better cloud security and faster performance. In contrast, a cheaper option may lack important features or have limited support.

**Example:**

* Solution A: Build a custom LMS from scratch → High cost, full control, long time.
* Solution B: Use Moodle (open-source) → Lower cost, less flexible, faster setup.

If the university needs full control and custom features, Solution A is better. But if cost is more important, Solution B may be preferred.

#### Decision Matrix

This is a table that helps compare different options based on key factors like cost, performance, scalability, and user experience. Each factor is given a score, and the total helps identify the best solution.

**Example:**

Table 4: Decision Matrix Comparing Custom LMS vs Moodle

|  |  |  |
| --- | --- | --- |
| **Criteria** | **Solution A (Custom)** | **Solution B (Moodle)** |
| Cost | 2/5 | 5/5 |
| Flexibility | 5/5 | 3/5 |
| Scalability | 4/5 | 3/5 |
| User Experience | 4/5 | 3/5 |
| **Total Score** | **15/20** | **14/20** |

Based on this matrix, Solution A might be more suitable if flexibility is a top priority.

### Apply a Decision Matrix to Select the Most Appropriate SDLC Model

For the LMS project, a **Decision Matrix** was used to choose the most suitable SDLC model among **Waterfall**, **Agile**, and **V-Model**.

The evaluation is based on feedback from three stakeholder groups: **University IT Department**, **Lecturers**, and **Administration**. Each group gave input on what is most important for the system. The criteria were then given weights to show their importance.

**Criteria and Weights:**

* **Development Speed (20%):** How fast the model can deliver working features. Important because the LMS should be ready before the new semester.
* **Error Detection Speed (20%):** How early and quickly problems are found. Critical to avoid major rework close to deployment.
* **Cost Control (25%):** How well the model keeps the project within the set budget. The university has a fixed budget.
* **Scalability (15%):** How easily the system can grow when more students, courses, or modules are added.
* **Stakeholder Involvement (20%):** How much feedback is collected from lecturers, students, and admin during the process.

**Scoring Method:**

* Each model was scored 1 to 5 (1 = very poor, 5 = excellent) for each criterion.
* The score was multiplied by the weight to get the weighted score.
* The model with the highest total weighted score is the most suitable.

Table 5: Decision Matrix for SDLC Selection

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Criteria** | **Weight** | **Waterfall** | **Agile** | **V-Model** |
| Development Speed | 0.20 | 2 (0.40) | 5 (1.00) | 3 (0.60) |
| Error Detection Speed | 0.20 | 3 (0.60) | 5 (1.00) | 4 (0.80) |
| Cost Control | 0.25 | 5 (1.25) | 3 (0.75) | 4 (1.00) |
| Scalability | 0.15 | 4 (0.60) | 5 (0.75) | 4 (0.60) |
| Stakeholder Involvement | 0.20 | 2 (0.40) | 5 (1.00) | 3 (0.60) |
| **Total** | **1.00** | **3.25** | **4.50** | **3.60** |

Note: The number in brackets is the weighted score.

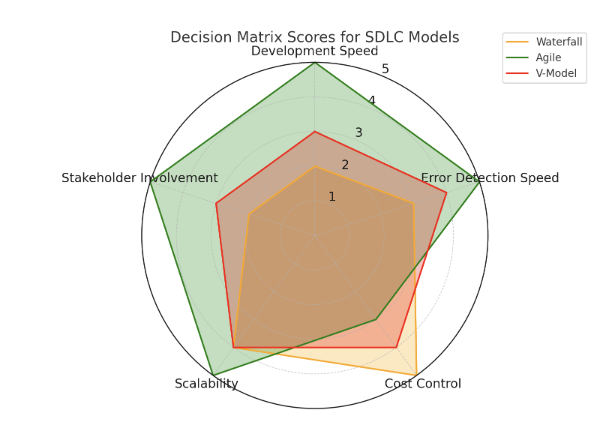


Figure 6: Radar chart visualizing the Decision Matrix scores for Waterfall, Agile, and V-Model.

**Justification of Scores:**

* **Development Speed:** Agile scores highest (5) because it delivers features in sprints. Waterfall scores low (2) as it delivers only after all phases are done.
* **Error Detection Speed:** Agile finds issues early through continuous testing, V-Model also performs well due to parallel testing planning, Waterfall detects errors late.
* **Cost Control:** Waterfall scores highest because budget and resources are fixed early. Agile can have scope creep, making it harder to control costs.
* **Scalability:** Agile adapts quickly to changes in size and complexity. Waterfall can scale, but changes are more expensive.
* **Stakeholder Involvement:** Agile involves stakeholders throughout development, Waterfall involves them mainly at the start and end.

Based on the weighted scores, **Agile** is the most suitable model for the LMS project, with a total score of **4.50**. It offers high speed, early error detection, and strong stakeholder engagement, which are important for a dynamic and evolving university learning environment.

In the LMS project, Agile’s iterative nature allows continuous lecturer and student feedback, ensuring that core features such as course creation, enrollment, and grading tools are tested and refined before final deployment. This approach ensures the system remains aligned with academic requirements while minimizing delays and post-launch issues.

### Realistic Application in the LMS Project

In the LMS project, the selection of Agile as the primary SDLC model directly supports the system’s development needs. The iterative sprint cycles allow the development team to deliver functional modules such as course creation, student enrollment, and grading tools in short timeframes. This enables lecturers, administrators, and selected student representatives to test these features early and provide feedback before the final deployment.

For example, during the initial sprint, the course creation module was implemented and tested with real instructor accounts. Feedback regarding interface simplicity and file upload options was collected and applied in the following sprint. Similarly, the enrollment module underwent multiple iterations, ensuring that role-based permissions and duplicate enrollment prevention were fully functional before launch.

By applying Agile, the project avoids the risk of delivering a rigid system that fails to meet evolving academic needs. Continuous stakeholder involvement ensures that the LMS remains aligned with semester schedules, teaching requirements, and institutional policies, while also reducing post-deployment modifications.

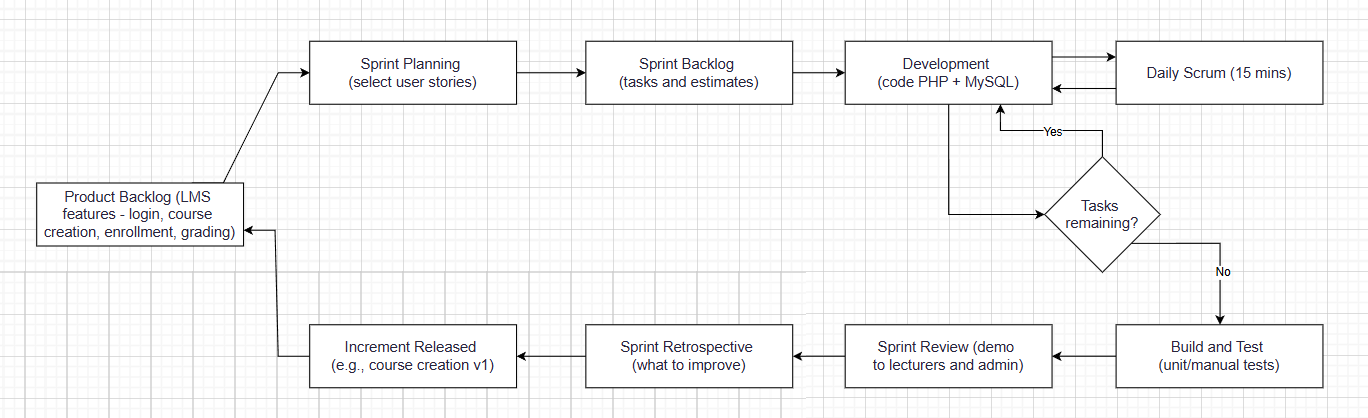


Figure 7: Agile sprint cycle for the LMS project showing backlog, planning, development & testing, review, retrospective, and continuous delivery of course and enrollment features.

## Discuss the components of a feasibility report (M2)

A feasibility report is made up of several key components. Each component helps assess whether the LMS project can be successfully developed and delivered. In this section, I will explain the main components and how they apply to the university LMS.

### Technical Feasibility

This part checks if the project can be built using the available technology. For the LMS project, technical feasibility includes:

* Compatibility with the university’s current systems (e.g., database, authentication).
* Ability to run on web browsers and mobile devices.
* Support for features like quizzes, video lectures, and discussion forums.

If the university has an existing IT infrastructure and skilled developers, the LMS is technically feasible.

### Economic Feasibility

This part checks if the project is affordable. The analysis includes:

* Development costs (e.g., staff salaries, tools, hosting).
* Maintenance and support costs.
* Expected return on investment (ROI), such as improved student engagement or lower administrative workload.

**Example:** Using an open-source platform like Moodle may reduce costs compared to buying a commercial LMS.

### Legal Feasibility

This section checks if the project follows laws and regulations. For the LMS, legal feasibility includes:

* Compliance with data protection laws such as GDPR.
* Secure handling of student records and personal information.
* Clear terms of use and privacy policy for students and staff.

Failing to meet legal requirements could result in fines or data breaches.

### Operational Feasibility

This part checks if the system will work well in the real university environment. Key questions include:

* Will staff and students use the system easily?
* Can instructors upload materials, manage courses, and communicate with students?
* Does the system support university goals like digital transformation and blended learning?

If users accept and understand the system, it is operationally feasible.

### Schedule Feasibility

This section examines whether the LMS can be completed on time. It includes:

* Estimated time for each phase: requirements, design, coding, testing, and deployment.
* Availability of human resources.
* Alignment with the academic calendar.

For example, launching the LMS before the next semester begins may be a critical goal.

## Assess the impact of different feasibility criteria on a software investigation (D2)

### Influence of Feasibility Criteria on Decision-Making

In the context of the Learning Management System (LMS) project, the feasibility study plays a key role in guiding all decision-making from the very beginning. Analyzing feasibility criteria helps identify limitations, opportunities, and potential risks. This supports optimizing the development plan, using resources effectively, and reducing mistakes during implementation.

### Impact on Project Management Aspects

**Project Planning:**

The technical feasibility analysis shows that choosing PHP and MySQL is suitable because they are popular technologies, easy to access, and well-documented. This allows the developer to quickly build and maintain the system.

The legal feasibility results highlight the need to follow student data protection regulations, leading to the decision to prioritize secure login and user role management in the first sprint.

Economic feasibility also shapes the initial scope of the project, limiting it to core functions such as course management, enrollment, and grading, avoiding excessive expansion that could overload resources.

**Resource Allocation:**

Organizational feasibility is important in allocating time efficiently between backend programming, interface design, testing, and gathering user feedback.

Social feasibility shows the need to maintain interaction with lecturers and trial students to adjust the interface and functions to match their usage habits, improving user experience.

**Risk Mitigation:**

During the LMS development process, risk management based on the feasibility study is essential to reduce negative impacts on progress and product quality. Risk reduction measures are built on scientific grounds and linked to the analyzed feasibility criteria:

* **Technical risk:** To avoid system interruptions or errors, the project uses a lightweight framework, MySQL database, and open-source tools to ensure maintainability and future scalability.
* **Legal risk:** To reduce the chance of breaking data protection laws, the system uses password encryption, HTTPS connections, and strict role-based access control from the design stage.
* **Economic risk:** Limited budget means prioritizing free or low-cost tools and services, which fit the financial conditions while maintaining functional quality.
* **Schedule risk:** With limited development time, sprints are designed with a clear scope and reasonable complexity to prevent overload and delays.

### Impact of Specific Constraints on Project Outcomes

Identifying and analyzing constraints early in the feasibility study helps shape the structure and implementation direction of the LMS project scientifically. The impact of each constraint group is as follows:

* **Legal constraints:** Compliance with personal data protection rules guides the system architecture toward security, including encrypting sensitive information and applying SSL certificates to all access points.
* **Social constraints:** The demographics and behavior of the target users (lecturers and students) encourage a simple, intuitive interface design with fewer steps to increase acceptance and usage efficiency.
* **Economic constraints:** Financial limitations lead to only implementing core features in the first phase, using open-source software and cost-effective hosting.
* **Technical constraints:** The developer’s technical ability requires selecting technologies suitable for current skills, avoiding overly complex solutions that would increase learning and deployment costs.
* **Organizational constraints:** Having only one person responsible for development requires a well-structured process, with tasks divided logically for each sprint to ensure both progress and quality.

## Executive Summary

This report presents the implementation and evaluation of a Learning Management System (LMS) designed for a university environment. The aim of the project is to provide a digital platform where instructors and students can manage courses, share learning materials, communicate effectively, and monitor academic progress. The LMS is intended to make learning more accessible and organized, replacing some traditional classroom activities with an online solution.

For the prototype, two main features were developed from two different modules:

* **Login System (Authentication Module):** This feature allows registered users to log in securely with a username and password. It ensures that only authorized users can access the platform, protecting sensitive course and student data.
* **Course Creation and Enrollment (Course Management Module):** This feature enables instructors to create new courses, upload learning materials, and enroll students. Students can join courses, view content, and participate in learning activities.

The system investigation identified the key needs of stakeholders, such as a secure login process, an easy-to-use interface, and essential tools for course management and communication. The evaluation results show that the prototype meets the main functional requirements and works well in a local XAMPP environment. However, for full deployment, the system would require more features such as automated notifications, grading tools, and advanced testing to ensure stability and security.

## Project Introduction

### Purpose

The purpose of this report is to document the implementation and evaluation phases of the Systems Development Life Cycle (SDLC) for the LMS prototype. It provides a clear record of the results from the system investigation, the steps taken to develop key features, and the methods used to check software quality. This report will also help guide future improvements, showing what has been completed successfully and what needs to be developed further.

### Project Overview

The Learning Management System is a **web-based application** that allows academic activities to be managed online. The system supports instructors in creating courses, uploading documents or video lessons, and enrolling students into specific classes. Students can log in to the system, view available courses, access materials, and participate in learning tasks. The platform aims to improve communication between instructors and students while storing all information in a centralized database.

The LMS prototype was built using **PHP** for the backend logic, **MySQL** for the database, and **HTML/CSS** for the user interface. It is hosted locally on **XAMPP**, making it easy to develop and test without the need for external hosting at this stage. The design is simple to fit the needs of a prototype while leaving space for future expansion.

### Scope

**Objectives:**

* Provide a secure authentication system where users can log in using a verified account.
* Enable instructors to create and manage courses, including adding descriptions and materials.
* Allow instructors to enroll students into courses and manage participation records.
* Support basic data storage for users, courses, and enrollments.
* Demonstrate feasibility through a working prototype that meets at least two functional requirements from different modules.

**Deliverables:**

* A fully functional prototype with the login system and course creation/enrollment features.
* Documentation including system investigation results, diagrams (Context Diagram, DFD, ERD), and system options (BSOs, TSOs).
* An evaluation section covering software quality, testing results, and recommendations for improvements.
* Evidence of manual testing and peer code reviews to ensure the developed system meets basic quality standards.

## Undertake a software investigation to meet a business need (P5)

The chosen project is a **Learning Management System (LMS)** for a university. The main business need is to provide a single, web-based platform for managing courses, student enrollment, and academic activities. Currently, course materials and enrollment processes are handled manually or through different, unconnected systems, which causes inefficiency and poor tracking of student progress.

The software investigation focused on understanding stakeholder requirements and **technical constraints** before implementation. Stakeholders included:

* **University administration:** wants a secure, centralized system to store course and student data.
* **Instructors:** need tools to create courses, upload materials, and track student performance.
* **Students:** want easy access to learning materials and the ability to enroll in courses online.

**Functional requirements identified:**

1. **Authentication & Login:** Users must log in with a username and password. The system must differentiate between students, instructors, and admins.
2. **Course Creation:** Instructors can create new courses, add course descriptions, and set enrollment options.
3. **Student Enrollment:** Students can join available courses and view course materials.
4. **User Role Management:** Permissions should change depending on user role.

**Non-functional requirements identified:**

* **Security:** Passwords stored using encryption (e.g., password\_hash() in PHP).
* **Usability:** Interface must be simple and intuitive for non-technical users.
* **Performance:** System should load pages within a few seconds.
* **Scalability:** Ability to support more courses and users in the future.

**Investigation methods used:**

* **Interviews** with two instructors and three students to understand needs.
* **Observation** of current manual enrollment process to find inefficiencies.
* **Review** of existing LMS platforms to identify useful features and design patterns.

The result of the investigation confirmed that a web-based system is the best option because it can be accessed from anywhere, supports multiple roles, and can integrate with existing university infrastructure.

### Prototype Implementation

**Developed Features:** The prototype of the Learning Management System (LMS) was developed with two main features from two different modules. These features were chosen because they are essential for the core operation of the system and can clearly demonstrate how the LMS works.

1. **Login System (Authentication Module)**

* **Purpose:** To allow registered users to access the system securely. This prevents unauthorized access and protects sensitive data such as student records and course materials.
* **How it works:** Users must enter their username (or email) and password on the login page. The system checks the credentials against the database. Passwords are stored securely using PHP’s **password\_hash()** and verified with **password\_verify()**. If the information is correct, the user is redirected to their dashboard, which is different for students, instructors, and administrators.
* **Front-end:** HTML and CSS were used to create a simple, clear login form. JavaScript was used to show validation messages if the fields are empty or incorrect.
* **Back-end:** PHP scripts handle form submission, database queries, and session management.
* **Database:** MySQL table **users** stores user information such as ID, name, email, password hash, and role.

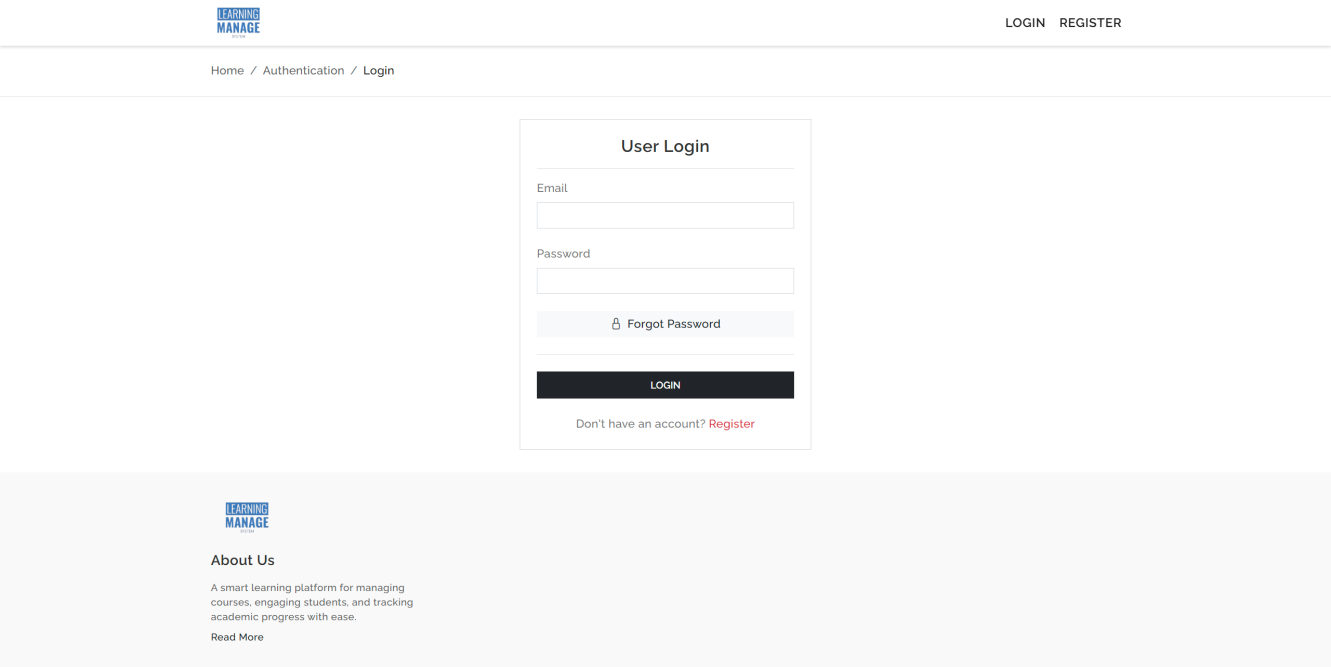


Figure 8: Login System (Authentication Module)

1. **Course Creation and Enrollment (Course Management Module)**

* **Purpose:** To let instructors create courses and add students, and to let students join courses and access learning materials.
* **How it works (Instructor side):** An instructor can go to the “Create Course” page, fill in the course title, description, and upload materials such as PDF files or videos. The course is saved in the **courses** table, and materials are saved in the **materials** table. The instructor can then enroll students by selecting them from the list of registered users.
* **How it works (Student side):** A student can log in, see the list of available courses, and click “Enroll.” This action creates a record in the **enrollments** table linking the student to the selected course. Students can then open the course page to view and download materials.
* **Front-end:** HTML, CSS, and basic JavaScript to build forms and display course lists.
* **Back-end:** PHP scripts to process course creation, file uploads, and enrollment requests.
* **Database:** MySQL tables **courses**, **materials**, and **enrollments** store course information, materials, and enrollment records.

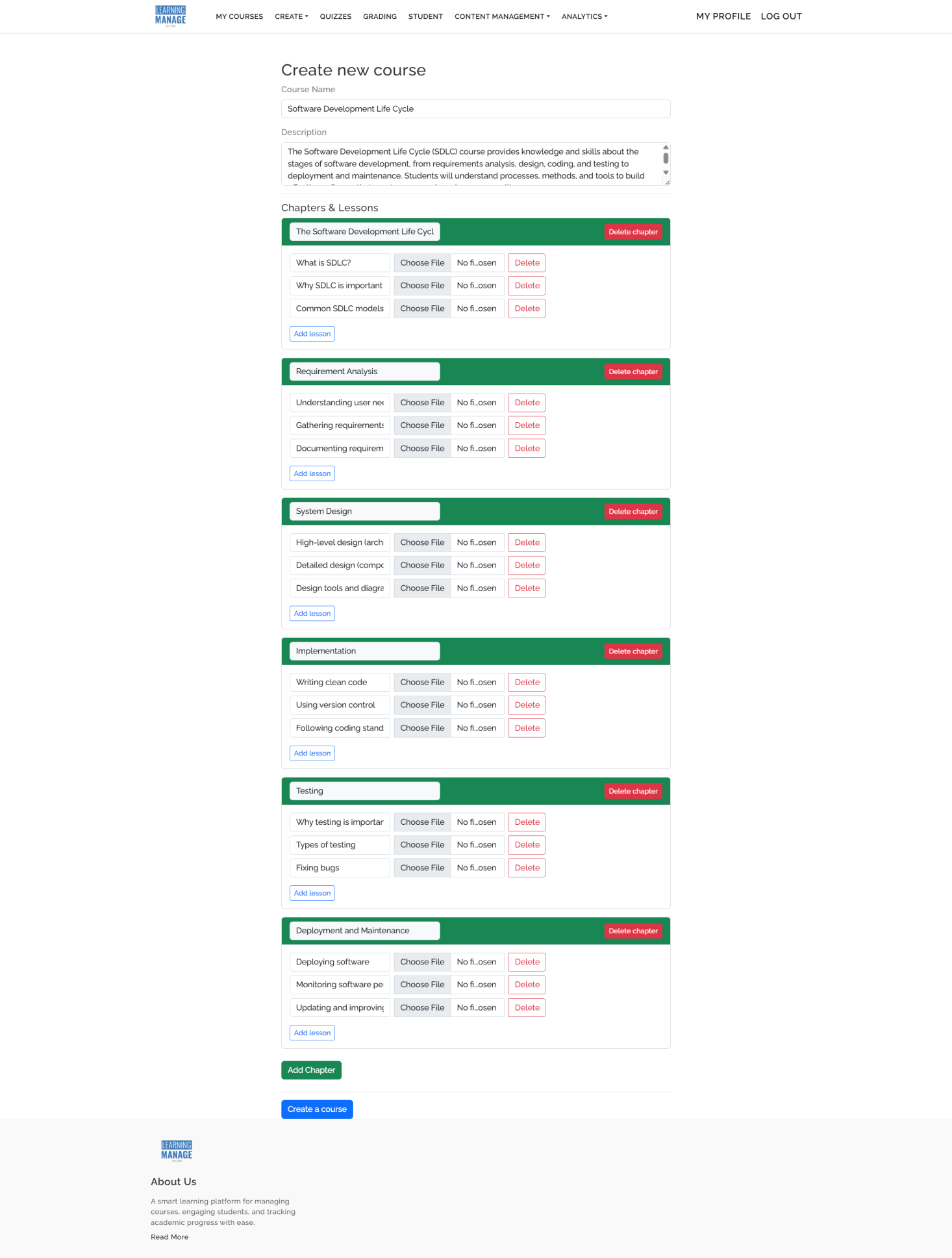


Figure 9: 2.Course Creation (Course Management Module)

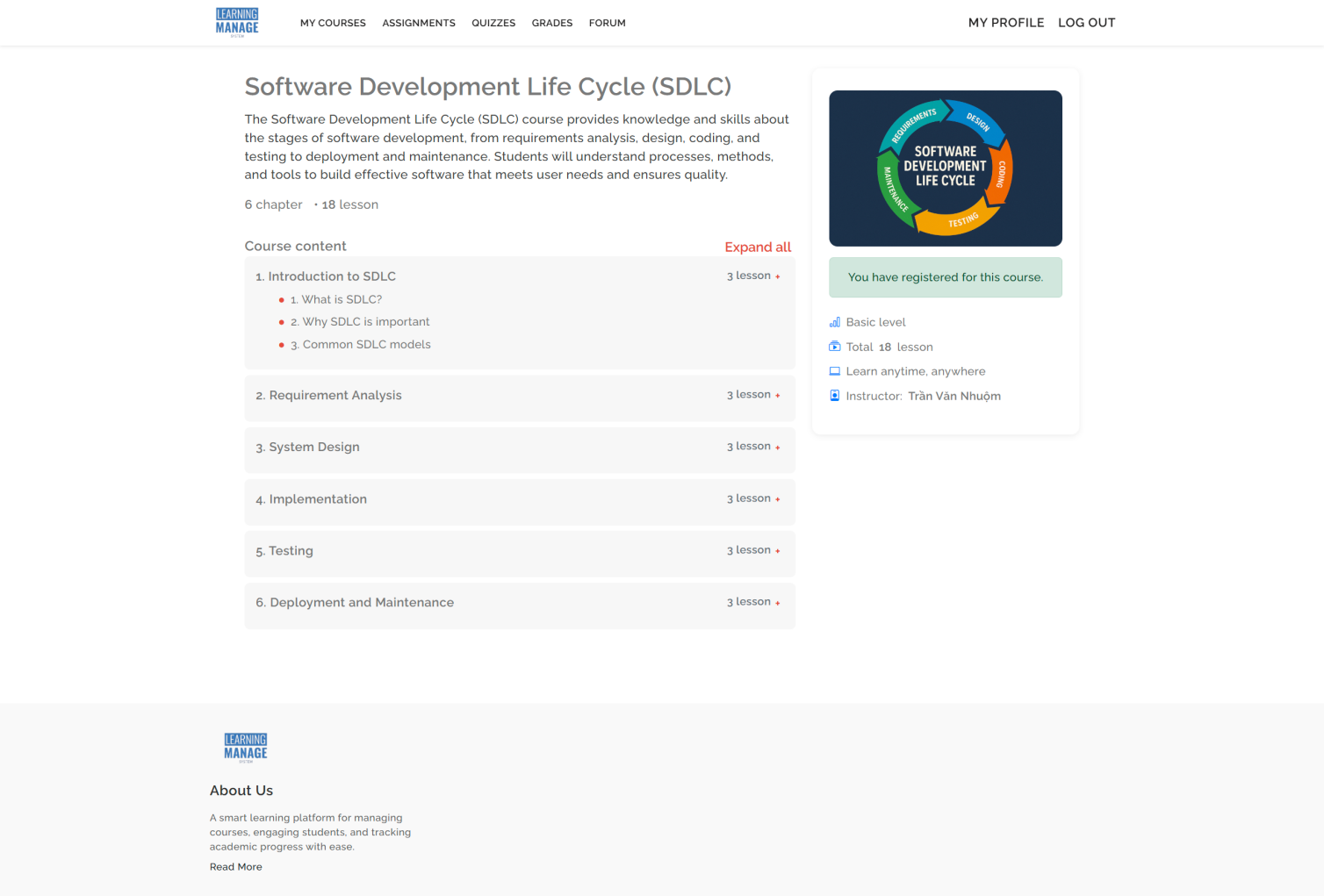


Figure 10: Enrollment (Course Management Module)

**Deployment:** The prototype was deployed locally using the XAMPP package, which includes Apache, PHP, and MySQL. This setup allows the system to run on a personal computer without the need for an external web server.

**Deployment steps:**

1. Install XAMPP on the development computer.
2. Copy the LMS project folder into the htdocs directory of XAMPP.
3. Start Apache and MySQL from the XAMPP Control Panel.
4. Create a MySQL database called lms\_sdlc using phpMyAdmin.
5. Import the provided SQL file to set up the required tables and sample data.
6. Access the system through a web browser at http://localhost/LMS\_1.

**Testing environment:** The system was tested using Google Chrome and Mozilla Firefox browsers. The prototype runs smoothly on a local network, allowing different roles (student, instructor, admin) to log in and test their specific features.

**Evidence of deployment:** Screenshots were taken of the login page, instructor course creation page, and student course enrollment page. These screenshots show that the system is working as expected in the local environment.

This deployment method is suitable for development and testing. For future stages, the LMS can be moved to a cloud hosting service so that it can be accessed by users anywhere with an internet connection.

## Use appropriate software analysis tools/techniques to carry out a software investigation and create supporting documentation (P6)

In this stage, different software analysis tools and techniques are used to understand the system design before implementation. The following diagrams show the LMS structure, processes, options, and database. The Context Diagram gives a high-level view of the system and its external actors. The DFD Level 0 explains the main processes and data flows. The BSO diagram compares business options. The TSO diagram analyses technical options. The ERD shows the required database tables and relationships. Together, these diagrams give a clear picture of the system investigation results.

### Context Diagram

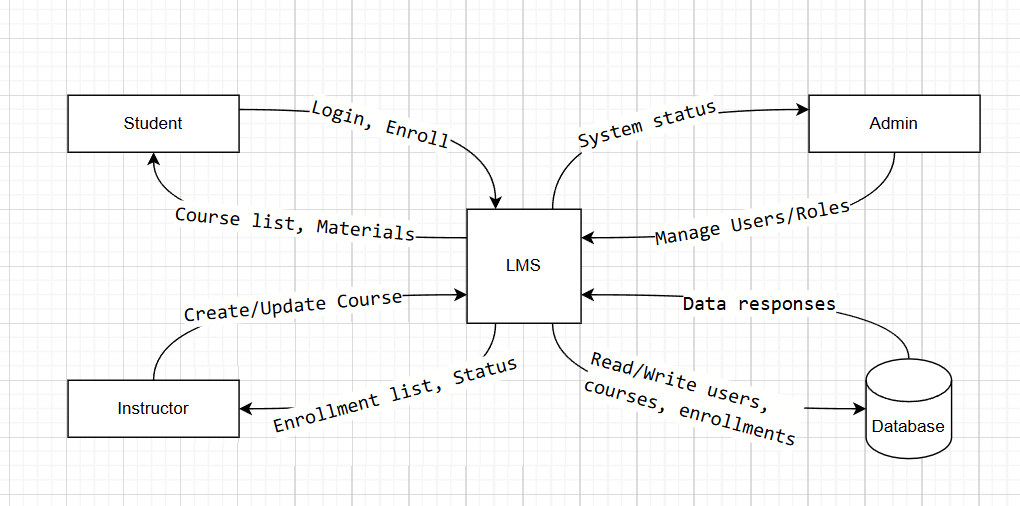


Figure 11: Context Diagram of the LMS

This diagram shows the LMS as the main system. It connects to Students, Instructors, Administrators, and the Database. Students can log in, enroll, and get course materials. Instructors can create or update courses and see the enrollment list. Administrators can manage users and roles. The database stores all course, user, and enrollment information.

### Data Flow Diagram (DFD) Level 0

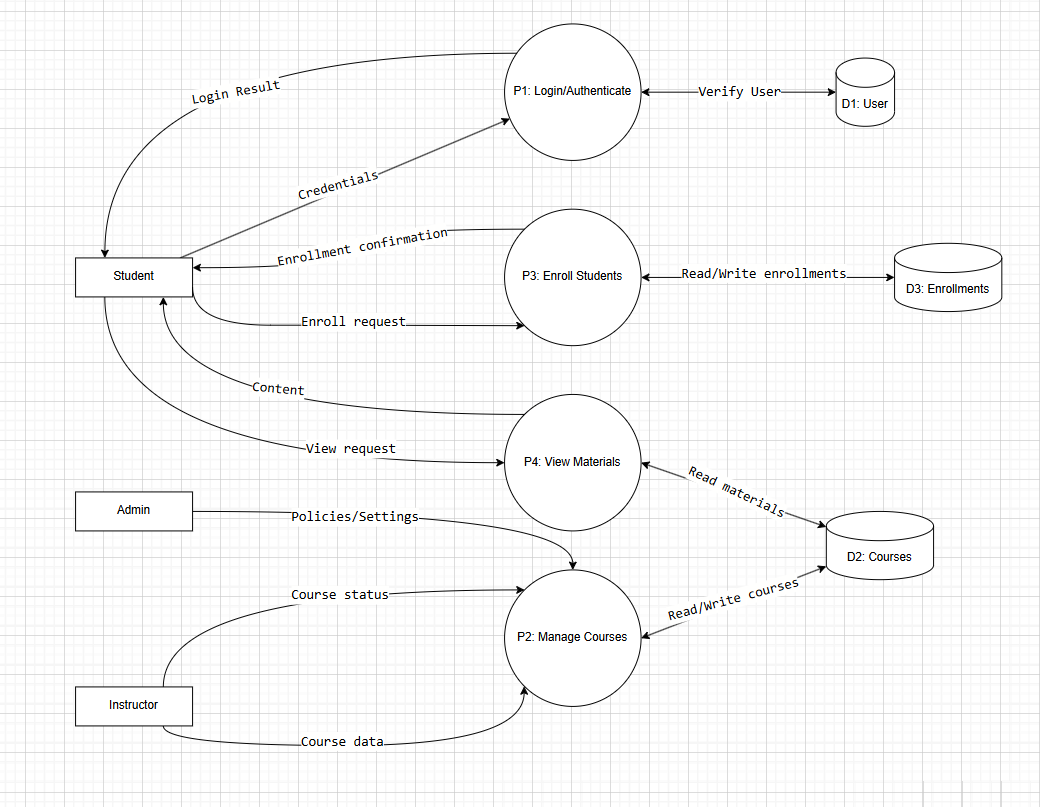


Figure 12: Data Flow Diagram (DFD) Level 0 of the LMS

The DFD Level 0 shows the main processes of the LMS: P1 Login/Authenticate, P2 Manage Courses, P3 Enroll Students, and P4 View Materials. It also shows how these processes interact with external entities and the database. Data flows between users, processes, and storage are represented with arrows.

### Entity Relationship Diagram (ERD)

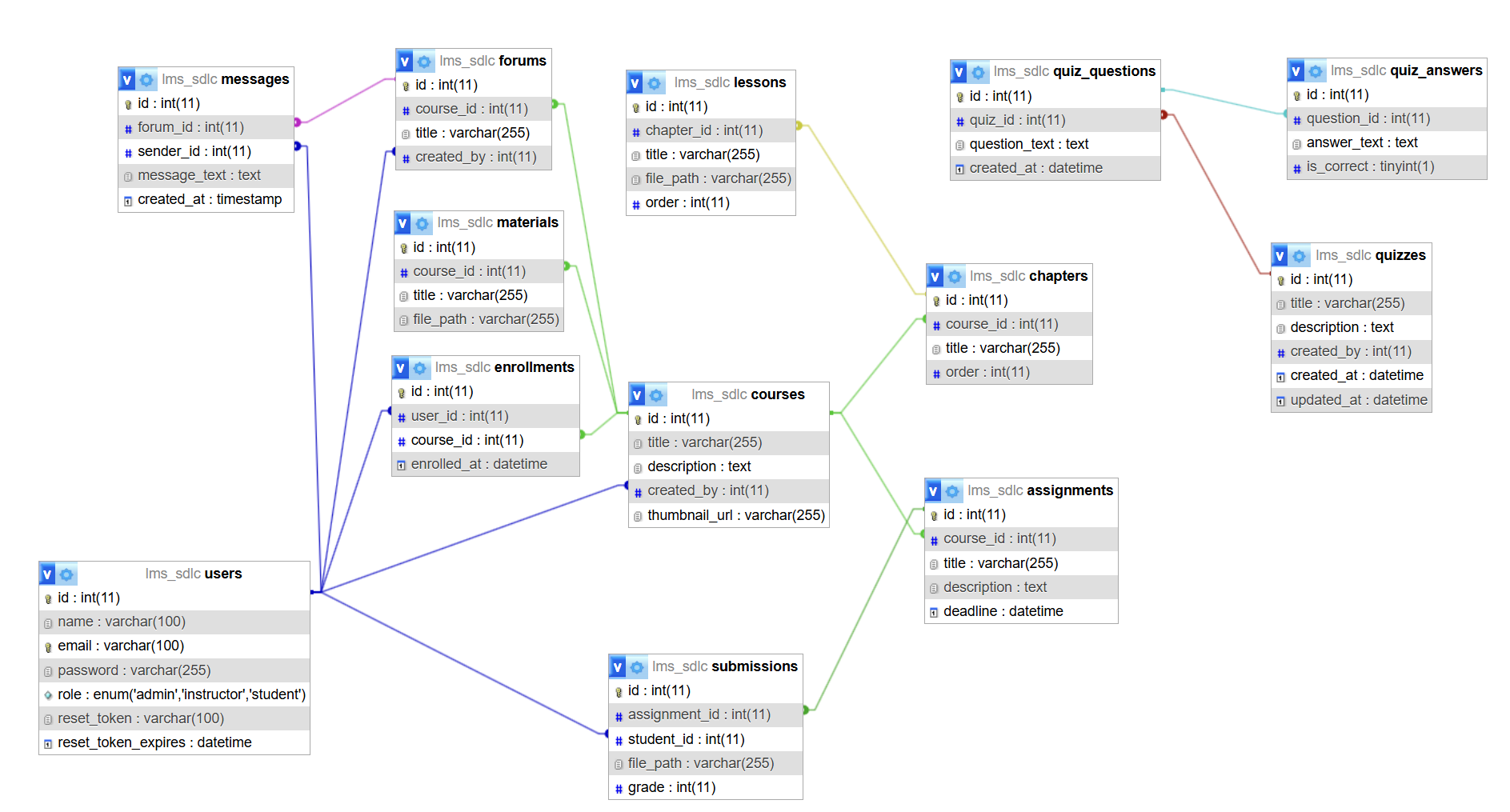


Figure 13: Entity Relationship Diagram (ERD) of the LMS

The ERD shows the core database for the LMS prototype. These tables are required to support login, course creation, and enrollment. Other tables in the picture are marked as extensions because I wanted to expand the LMS for future work.

**Required core tables (for the prototype):**

* **users** (id PK, name, email UNIQUE, password\_hash, role ENUM[admin,instructor,student], created\_at). Stores all system accounts and roles.
* **courses** (id PK, title, description, created\_by FK→users.id, created\_at, thumbnail\_url NULL). Stores course information. Each course belongs to one instructor (created\_by).
* **enrollments** (id PK, user\_id FK→users.id, course\_id FK→courses.id, enrolled\_at). Links students to courses (many-to-many).
* **materials** (id PK, course\_id FK→courses.id, title, file\_path, created\_by FK→users.id, created\_at, sort\_order INT). Keeps learning files for a course (PDF, slides, etc.).

**Main relationships:**

* One **instructor (users.role = instructor)** can create **many courses**: users.id (instructor) **1—N courses.created\_by**.
* One **student (users.role = student)** can enroll in **many courses**, and each **course** has **many students** → relationship via **enrollments**: **users.id N—N courses.id** through **enrollments(user\_id, course\_id)**.
* One **course** has **many materials**: **courses.id 1—N materials.course\_id**.

**Why other tables appear in the ERD:**

Besides the core tables, I added some extra tables to extend the project for future features:

* **chapters, lesson**s: to organize content inside a course.
* **assignments, submissions**: to collect student work and store grades for that work.
* **quizzes, quiz\_questions, quiz\_answers**: to support online tests.
* **forums, messages**: to enable class discussion.

These extension tables are not required for the basic prototype, but they prepare the system for the next phases and make the LMS easier to scale later.

### Business System Options (BSOs)

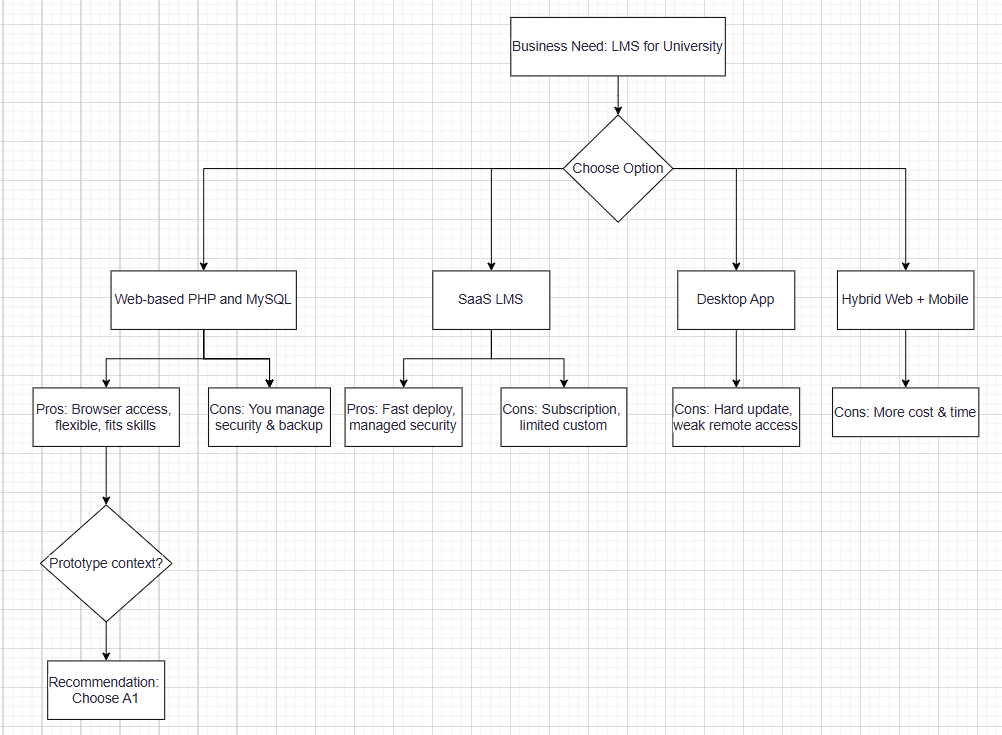


Figure 14: Business System Options (BSOs) of the LMS

This diagram shows four options for building the LMS: Web-based PHP + MySQL, SaaS LMS, Desktop App, and Hybrid Web + Mobile. It lists the advantages and disadvantages of each option. The recommended choice is Web-based PHP + MySQL because it is low cost, flexible, and fits the development skills.

### Technical System Options (TSOs)

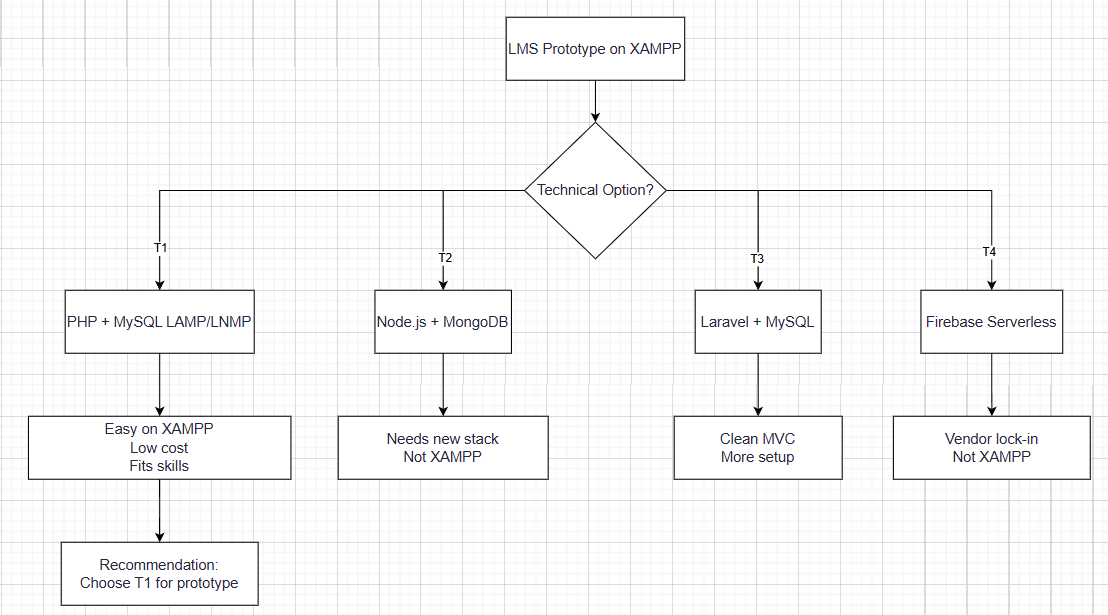


Figure 15: Technical System Options (TSOs) of the LMS

The TSO diagram compares four technical options: PHP + MySQL, Node.js + MongoDB, Laravel + MySQL, and Firebase Serverless. Each option has different setup needs, skills required, and costs. The selected choice is PHP + MySQL because it works well with XAMPP and is easy to implement.

## Analyse how software requirements can be traced throughout the software lifecycle (M3)

### Traceability Matrix

Below shows the traceability matrix linking stakeholder requirements to system features, design elements, test cases, and implemented components for the LMS prototype. This ensures that every requirement is clearly mapped to its corresponding implementation and verification method.

Table 6: Requirements Traceability Matrix for LMS Prototype

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Stakeholder Requirement** | **System Feature** | **Design Element** | **Test Case** | **Implemented Component** |
| **R1:** Users must log in securely before accessing the system. | Login System (Authentication Module) | Login page (HTML/CSS), PHP login script, password hashing | TC1: Enter valid username & password → expect redirect to dashboard.  TC2: Enter invalid credentials → expect error message. | login.php, users table in MySQL, password\_hash() in PHP |
| **R2:** Instructors can create courses and add descriptions. | Course Creation Form (Course Management Module) | Course creation page (HTML form), PHP insert script | TC3: Fill course title & description → expect new course record in DB. | create\_course.php, courses table |
| **R3:** Students can enroll in available courses. | Enrollment Function | Enroll button in course list page, PHP enrollment script | TC4: Click “Enroll” on a course → expect record in enrollments table. | enroll.php, enrollments table |
| **R4:** Users should see only features allowed for their role (student/instructor/admin). | Role-based dashboard | PHP session role checking, conditional menu display | TC5: Log in as student → no “Create Course” option.  TC6: Log in as instructor → “Create Course” option visible. | dashboard.php, role column in users table |
| **R5:** Course materials should be stored and accessible only to enrolled students. | Course Material Upload & View | File upload form, PHP file access control | TC7: Upload PDF file as instructor → file stored in server directory and linked to course.  TC8: Student not enrolled → access denied. | upload\_material.php, materials table, server storage folder |

### Lifecycle Tracking

The requirements for the Learning Management System (LMS) project were monitored and managed carefully throughout each stage of the Systems Development Life Cycle (SDLC) to ensure that all planned functionalities were implemented correctly and tested thoroughly.

In the **Initial Gathering Stage**, requirements were collected through interviews with two instructors, three students, and representatives from the university administration. Functional requirements such as the Login System, Course Creation, Student Enrollment, and Role Management were prioritized according to stakeholder feedback. Non-functional requirements, including security, usability, performance, and scalability, were also identified as key factors to ensure long-term system quality.

During the **Analysis Stage**, all requirements were documented in a requirements list and a traceability matrix. A feasibility analysis was performed to determine which features could be developed within the time and resource limitations. As a result, the development scope for the prototype was defined to include two main features from different modules: secure login (Authentication Module) and course creation with enrollment (Course Management Module).

In the **Design Stage**, these requirements were transformed into detailed design elements. This included page layouts, database structures represented in the Entity Relationship Diagram (ERD), and process flows illustrated in the Context Diagram and Data Flow Diagram (DFD). Business System Options (BSOs) and Technical System Options (TSOs) were also created to help select the most suitable technical approach.

During the **Implementation Stage**, the database schema was developed with the necessary tables (users, courses, enrollments, and materials). PHP scripts were then implemented for authentication, course creation, and enrollment functionalities. The file structure was organized, and access controls were configured to meet security requirements.

Finally, in the **Testing Stage**, manual test cases were designed to validate each functional requirement. Both positive and negative scenarios were executed, such as logging in with correct and incorrect credentials, creating courses with valid and invalid data, and preventing unauthorized enrollment. The results were recorded and compared with the expected outcomes listed in the traceability matrix, confirming that the requirements had been fulfilled.

### Change Management

Throughout the development process, several requirement changes occurred. Each change was assessed for its effect on the **project scope**, **timeline**, and **overall quality**.

The first major change involved **expanding the project scope**. Initially, the system was intended to include only the Login and Course Creation functionalities. However, following discussions with stakeholders, Course Enrollment was added to the prototype to provide more practical value for students. This adjustment required additional database tables and scripts, which slightly extended the implementation schedule.

The second change related to **security improvements**. The original plan was to store passwords in plain text during the development phase for convenience. After a security review, this approach was replaced with PHP’s password\_hash() function for encrypting stored passwords. Although this required more development and testing time, it greatly enhanced data protection.

The third change involved **user role management**. At first, all users accessed the same dashboard interface. Based on instructor feedback, the dashboard was modified to display role-specific menus and features, such as allowing only instructors to access the “Create Course” option. This required updates to authentication logic and adjustments to the user interface design.

**Impact Analysis:**

* **Scope**: Expanded due to the inclusion of additional features and security enhancements.
* **Timeline**: Extended by approximately one week to accommodate the extra development and testing.
* **Quality**: Improved system reliability, data security, and user satisfaction, making the LMS more suitable for real-world deployment.

## Discuss two approaches to improving software quality (M4)

### Manual Testing

Manual testing was carried out to check the core functionalities of the LMS prototype in a local XAMPP environment. Each test was executed step-by-step, following pre-defined scenarios to ensure that both functional and non-functional requirements were met. The focus was on the two main modules—**Authentication** and **Course Management**—as well as additional supporting features such as enrollment and role-based access.

**Test Cases and Results:**

Below presents the manual test cases executed for the LMS prototype, including the tested functionality, scenario, expected outcome, actual result, and status. These cases validate that the implemented features meet the defined requirements.

Table 7: Manual Test Cases and Results for LMS Prototype

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test ID** | **Functionality** | **Test Scenario** | **Expected Result** | **Actual Result** | **Status** |
| TC1 | Login – Valid credentials | Enter correct username and password for a registered user. | User is redirected to the correct dashboard based on their role. | Works as expected; student and instructor see different menus. | Pass |
| TC2 | Login – Invalid credentials | Enter incorrect password for an existing username. | Error message “Invalid username or password” is shown. | Error message displayed correctly; no access granted. | Pass |
| TC3 | Course Creation – Valid data | Instructor fills in course title and description, then clicks “Create.” | New course appears in the database and course list. | Course created successfully and visible in instructor’s course list. | Pass |
| TC4 | Course Creation – Missing fields | Instructor submits form with no title. | Error message prompting to fill required fields is shown; no course is added. | Error message displayed; form reloaded without adding a record. | Pass |
| TC5 | Student Enrollment – Authorized | Student clicks “Enroll” on a visible course. | Student is added to enrollments table and can access course materials. | Enrollment successful; student can view uploaded course materials. | Pass |
| TC6 | Student Enrollment – Unauthorized | Student tries to access materials of a course without enrollment. | Access denied message is shown; student cannot see course files. | Works as expected; redirection to error/denied access page. | Pass |
| TC7 | Role-based Access | Instructor logs in and should see “Create Course” button; student logs in and should not see it. | Menu options change based on role; unauthorized functions are hidden. | Roles applied correctly; no unauthorized access possible. | Pass |
| TC8 | File Upload Security | Instructor uploads a PDF file for a course. | File stored in secure directory; only enrolled students can download it. | Files stored securely; access control enforced correctly. | Pass |

The manual testing confirmed that all implemented features worked according to the defined requirements. Minor adjustments, such as improving error message clarity, were identified and corrected during testing.

### Peer Code Review

Peer code reviews were conducted to ensure that the Learning Management System (LMS) prototype met quality standards, reduced the risk of defects, and maintained long-term code maintainability. The review was carried out by another developer, who examined all PHP scripts, HTML/CSS templates, and MySQL queries developed for the system.

The review process followed a structured approach:

* **Code Readability Check**: Verified that code used consistent indentation, descriptive variable names, and sufficient comments to explain complex sections.
* **Logic Validation**: Ensured that the scripts for login, course creation, and course enrollment followed correct logical flows and handled both valid and invalid inputs effectively.
* **Security Assessment**: Confirmed the correct use of PHP functions password\_hash() and password\_verify() for authentication, and verified that all SQL queries used prepared statements to prevent SQL injection attacks.
* **Performance Review**: Checked that database queries were optimized, unnecessary queries were avoided, and appropriate indexes were applied for faster data retrieval.
* **User Interface Review**: Compared HTML forms and pages against the intended design to ensure usability, clear navigation, and proper alignment with user requirements.

**Findings and Improvements Implemented:**

1. Added prepared statements to all SQL queries to strengthen defense against SQL injection.
2. Improved session handling by regenerating session IDs upon successful login to reduce the risk of session hijacking.
3. Adjusted redirect logic so that users are always directed to the correct role-based dashboard after authentication.
4. Enhanced HTML form validation to minimize errors before data submission to the server.

Through this review, the LMS prototype was improved in terms of security, performance, and usability. The peer review process not only helped identify potential issues early but also ensured that the codebase adhered to good development practices and met the required quality standards.

## Evaluate the process of undertaking a systems investigation with regard to its effectiveness in improving a software quality (D3)

### Strengths

The systems investigation stage for the LMS prototype demonstrated several strengths. Firstly, the use of structured analysis tools such as the Context Diagram, Data Flow Diagram (DFD), Entity Relationship Diagram (ERD), Business System Options (BSOs), and Technical System Options (TSOs) provided an accurate and comprehensive representation of system processes, boundaries, and data flows. These diagrams ensured that all stakeholders had a clear understanding of how the proposed system would operate. Secondly, the documentation produced during this stage was complete and well-organized. It included a detailed list of functional and non-functional requirements, feasibility analysis, and design justifications, which supported informed decision-making during the development process. Additionally, the alignment of investigation results with stakeholder needs helped ensure that the prototype addressed the most critical business objectives, such as secure authentication, course management, and enrollment processes.

### Weaknesses

Despite the successful outcomes, some weaknesses were observed during the investigation phase. Communication with stakeholders, particularly instructors and administrative staff, occasionally lacked clarity due to differences in technical knowledge. This sometimes led to the need for follow-up meetings to clarify requirements and expectations. Furthermore, the scope of the investigation expanded when additional features, such as course enrollment and role-based dashboards, were introduced during discussions. While these additions improved the system’s functionality, they also increased the complexity of design and required additional time for analysis. Technical constraints were also present, as the development environment was limited to a local XAMPP setup, which restricted early testing of scalability and remote access.

### Recommendations

For future projects, several refinements could improve the systems investigation process. More structured stakeholder communication methods, such as requirement workshops or standardized questionnaires, should be used to reduce misunderstandings and ensure all parties have a shared vision of the system. Requirement prioritization should be formally established at the start of the project to avoid unexpected scope changes during later stages. Additionally, the technical analysis phase should include early consideration of deployment environments beyond the local server, allowing for better assessment of scalability and integration with remote access features. Incorporating these refinements would help streamline the investigation process, minimize rework, and ensure the project remains aligned with its objectives and constraints.

## Discuss, using examples, the suitability of software behavioural design techniques (P7)

### Examples of Techniques

In this project, three behavioural design techniques were applied: **Flowcharts**, **Pseudocode**, and **Finite State Machines (FSMs)**. These techniques were used to design, explain, and implement the main functions of the Learning Management System (LMS): **user login**, **instructor course creation**, and **student course enrollment**. They helped to ensure that the system’s logic was clear before starting development, and they improved communication between developers and stakeholders.

**Flowcharts for visualizing workflows**

Flowcharts were created to show the process of each main function in a step-by-step way. This makes it easier to understand the system before coding.

Figure 13 below presents the login process. The diagram explains how the system displays the login page, validates the input, checks the user in the database, verifies the password, and redirects to the correct dashboard according to the user’s role. It also includes error handling for invalid accounts and incorrect passwords.

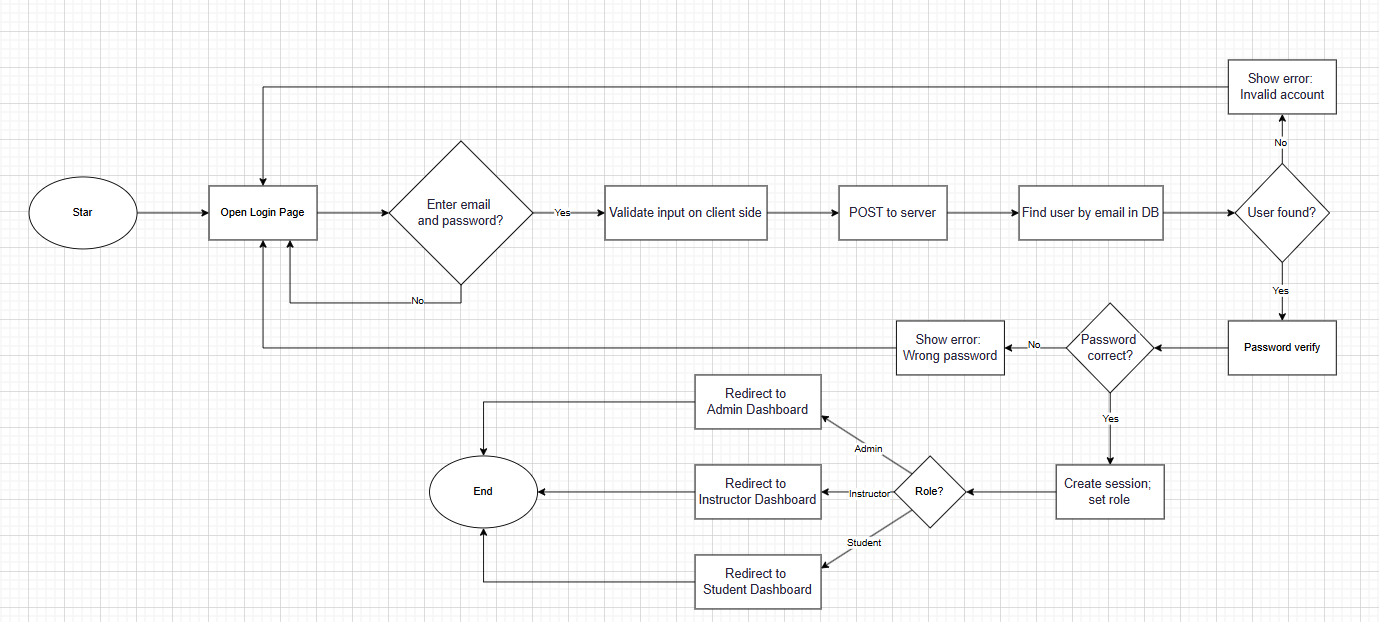


Figure 16: Login process flowchart

Figure 14 below illustrates the student course enrollment process. It shows how the system verifies that the user is a student, checks whether the student is already enrolled in the course, and either displays a message or processes the enrollment. If enrollment is successful, the system grants access to the course materials.

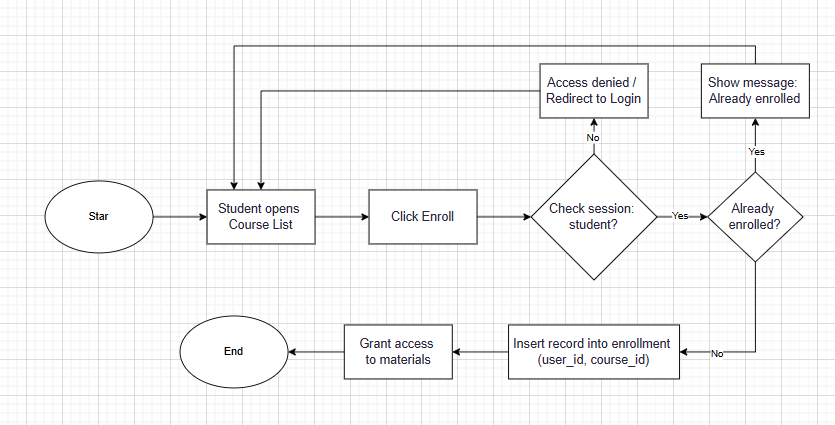


Figure 17: Student course enrollment flowchart

Figure 15 below shows the course creation process for instructors. It describes how the system verifies the login and role, processes the submitted form, validates the course title and thumbnail, saves the course information to the database, and uploads course materials. Only instructors can perform this process.

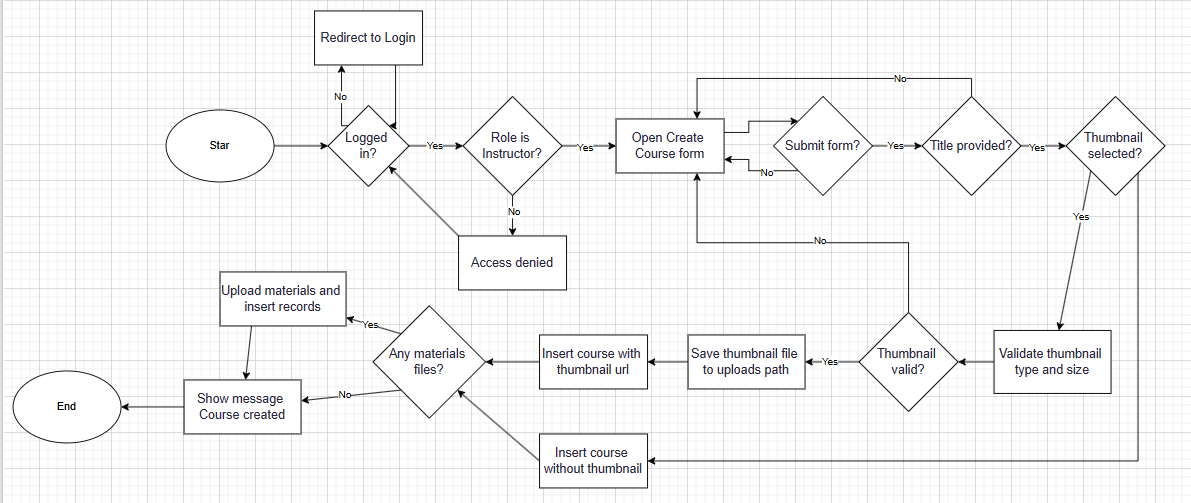


Figure 18: Instructor course creation flowchart

### Pseudocode for defining algorithms or logic flows

Pseudocode was developed for each of the three main LMS functions. It was written in a structured, human-readable way to describe the algorithm without relying on programming language syntax. This approach made it easier to verify logic before coding and simplified communication between team members.

Login Pseudocode: Validates the email and password, searches the database for a matching user, verifies the password using secure hashing, and redirects to the correct dashboard based on the user’s role.

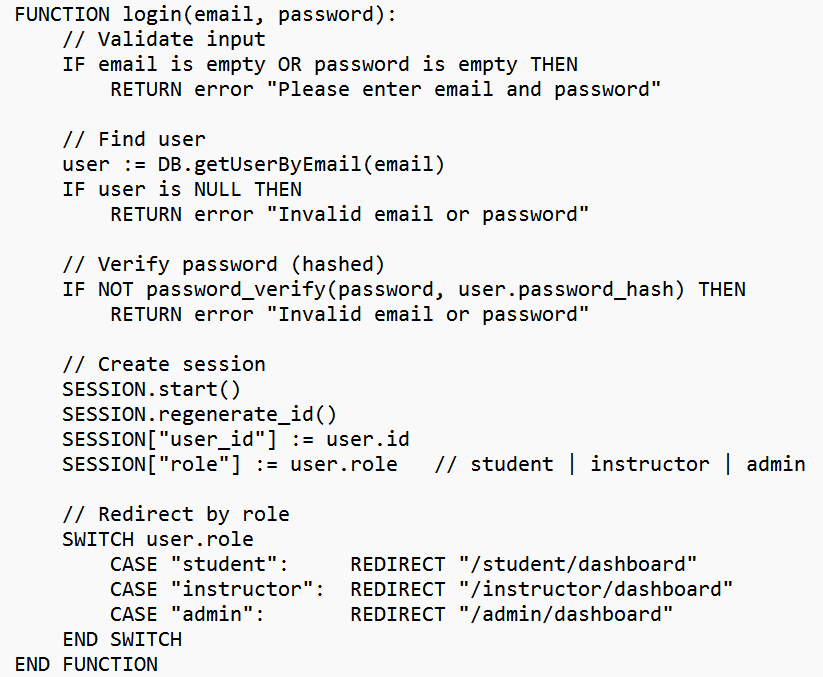


Figure 19: Login Pseudocode

Course Enrollment Pseudocode: Confirms that the session belongs to a student, checks if the student is already enrolled, and if not, adds a new enrollment record before granting access to course materials.

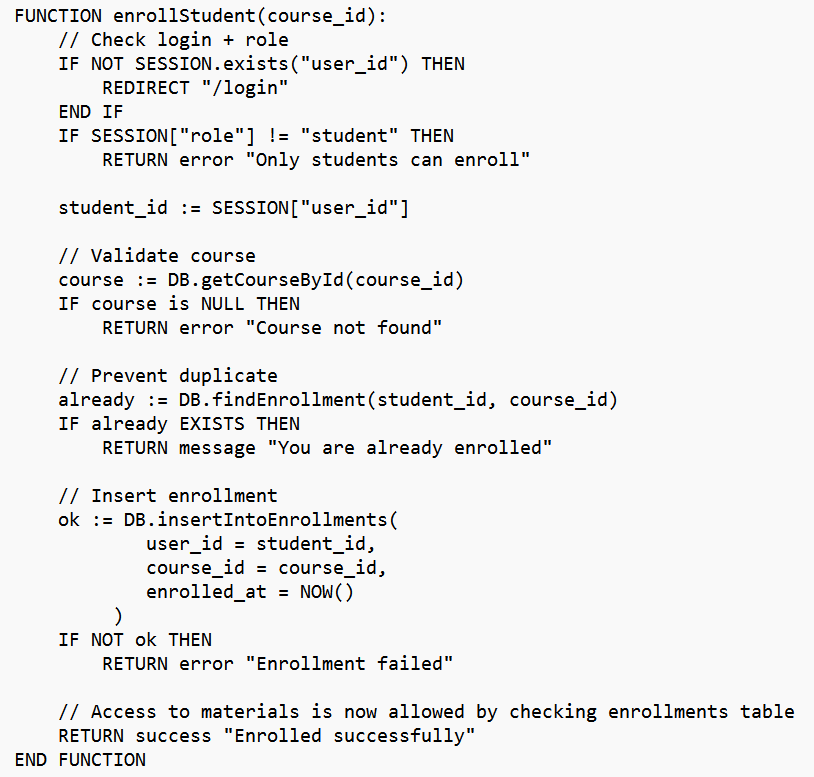


Figure 20: Course Enrollment Pseudocode

Course Creation Pseudocode: Checks that the logged-in user is an instructor, validates the course title and description, uploads the thumbnail if provided, stores the course information in the database, and uploads any course files.

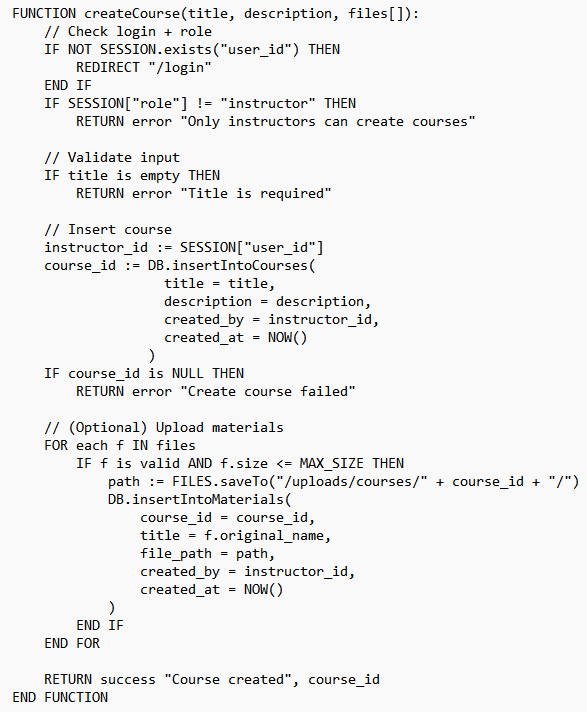


Figure 21: Course Creation Pseudocode

By preparing these pseudocode scripts, the development process became faster and more accurate, as programmers could directly translate them into PHP and SQL code.

### Finite State Machines (FSMs) for Modeling System States and Transitions

Finite State Machines were used to describe the possible states of the LMS and the transitions between them based on user actions. This was particularly useful for understanding role-based navigation and permissions.

Figure 19 below shows the Finite State Machine for the LMS. It begins in the “LoggedOut” state. After a successful login, the system moves to one of three states: LoggedInStudent, LoggedInInstructor, or LoggedInAdmin, depending on the user’s role. From these states, users can perform specific actions such as enrolling in a course, creating a course, or managing users. A logout action or session timeout returns the system to the “LoggedOut” state.

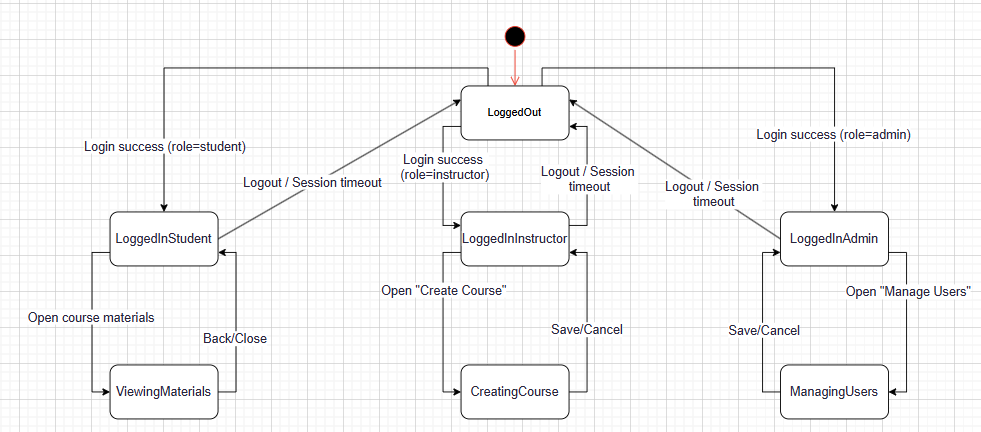


Figure 22: Finite State Machine for user roles and navigation

### ****Summary of Benefits****

Using these three behavioural design techniques provided several benefits to the LMS project:

* **Improved Clarity**: Stakeholders could see exactly how processes worked before implementation.
* **Reduced Errors**: Potential logic problems were identified early during the design stage.
* **Faster Development**: Pseudocode served as a ready-to-use reference for writing the actual code.
* **Better Communication**: Visual diagrams and FSMs made it easier to explain functionality to non-technical team members.
* **Role-Based Accuracy**: FSMs ensured that the system correctly handled different types of users and their permissions.

These techniques not only improved the quality of the LMS prototype but also made the development process smoother, more collaborative, and more efficient.

## Analyse a range of software behavioural tools and techniques (M5)

In the Learning Management System (LMS) project, several behavioural design tools and approaches were applied to ensure the system functioned effectively, responded correctly to user actions, and maintained accuracy in its processes. The analysis below explains how these tools were used and the benefits they provided.

### Range of Tools

The LMS prototype follows an event-driven approach, meaning the system reacts directly to specific user actions such as button clicks, form submissions, and menu selections. Examples include:

* **Login Button Click**: When a user clicks the “Login” button, the system triggers an authentication process. This includes validating input fields, checking the database for the user’s record, and verifying the password.
* **Course Creation Submission**: When an instructor submits the course creation form, the system validates the entered data, saves the course details to the database, and uploads a course thumbnail.
* **Enroll Button Click**: When a student clicks “Enroll” on a course, the system verifies their session, checks whether they are already enrolled, and updates the database if enrollment is valid.

This event-driven design improves user experience by providing **immediate feedback** such as success confirmations or error messages. It also ensures processes are executed **only when needed**, reducing unnecessary server load and increasing system efficiency.

### State/Data-Driven Approaches

The LMS implements a state/data-driven design to manage user sessions, control permissions, and define available actions. The Finite State Machine (FSM) diagram created during the design phase models how the system transitions between states based on login status and user role:

* **LoggedOut State**: Access is limited to public pages such as the login screen.
* **LoggedInStudent State**: Students can browse available courses and enroll but cannot create new courses.
* **LoggedInInstructor State**: Instructors can create, edit, and manage their own courses.
* **LoggedInAdmin State**: Admins have access to advanced system management functions.

All actions are controlled by **session data** stored on the server (e.g., user role, user ID). This ensures that users only see features and perform actions permitted for their role, improving security and preventing unauthorized access.

### Formal Specification Methods

Although the project was developed within a limited timeframe, some formal specification techniques were applied to improve accuracy in design and development:

* **Pseudocode** was written for each major function, defining step-by-step logic and expected outcomes before implementation.
* **Flowcharts** visually represented processes such as login, course creation, and course enrollment. These diagrams helped both technical and non-technical stakeholders understand workflows and verify logic before coding began.
* **Requirements Traceability Matrix** linked stakeholder needs to system features, design elements, test cases, and implementation components. This ensured that no functional or non-functional requirement was overlooked.

By combining pseudocode and diagrams, the risk of **misunderstanding requirements** was reduced, and developers had a **clear reference** during the coding stage.

### Strengths and Limitations

**Strengths**

1. **Clear Process Visualization**

Flowcharts and Finite State Machine (FSM) diagrams provided a clear representation of workflows, which helped both technical and non-technical stakeholders understand how the system would function before implementation.

Early visualization reduced the possibility of misunderstandings and made it easier to identify gaps or inefficiencies in the process design.

1. **Role-Based Accuracy**

The state/data-driven approach strictly controlled what each user role could access and perform.

This ensured that sensitive actions—such as course creation, enrollment management, and file uploads—were restricted to authorized roles, significantly improving system security.

1. **Efficient Development with Pseudocode**

Pseudocode acted as a direct blueprint for writing the actual code, saving development time and improving overall code consistency.

It also proved useful in onboarding new developers, as they could quickly understand the logic without having to read and interpret the entire source code.

1. **Responsive User Interaction**

The event-driven design allowed the system to respond immediately to user actions.

For example, the system displayed instant error messages for incorrect login attempts and provided confirmation after successful course enrollments.

This responsiveness improved user experience and reduced confusion during interaction with the LMS.

**Limitations**

1. **Limited Feature Scope in Prototype**

The prototype implemented only three core features: login, course creation, and course enrollment.

Other important Learning Management System (LMS) features—such as grading, internal messaging, and content scheduling—were excluded due to time constraints.

1. **Simplified Formal Specifications**

The project used pseudocode and diagrams as the main design documentation instead of more advanced formal specification methods like Z notation or VDM.

While this approach was adequate for a small prototype, it may limit accuracy and precision in larger or more complex systems.

1. **Manual Testing Only**

All testing was conducted manually, which required more time and effort.

This approach is less effective for detecting regression bugs during future updates and lacks the efficiency of automated testing tools.

1. **Partial FSM Implementation**

Although FSM diagrams were created during the design phase, they were not fully implemented in the live system for real-time state tracking.

Instead, the project relied primarily on PHP session checks, which may not be sufficient for managing more complex state transitions in future versions of the LMS.

## Differentiate between a finite state machine (FSM) and an extended FSM, providing an application for both (M6)

### Key Differences

**Finite State Machines (FSMs)** are a behavioral modeling technique used to represent a system as a collection of clearly defined states. Transitions between these states are triggered by specific events or conditions.

In the LMS project, FSMs work well for modeling predictable workflows such as the login process or simple role-based navigation. They are straightforward to design, implement, and maintain. This makes them ideal for scenarios where the next state depends only on the current state and the triggering event.

One of the main strengths of FSMs is their clarity. The diagrams are highly visual, so both technical and non-technical stakeholders can quickly understand the process. However, FSMs do have limitations. They struggle with complex logic involving multiple variables and cannot store historical data that might be important for certain workflows.

**Extended Finite State Machines (EFSMs)** expand on the basic FSM model by adding variables, conditions, and actions to transitions. In an EFSM, the decision to move between states depends on both the current state and stored data values.

This added flexibility makes EFSMs more suitable for scenarios that require richer decision-making. For example, in the LMS project, EFSMs are used to verify a student’s role and enrollment status before allowing them to join a course. By combining control flow (state transitions) with data flow (variable evaluation), EFSMs can model more realistic and complex workflows.

The trade-off is that EFSMs are more complex to design and maintain. They require careful handling of variables to prevent logic errors and have a higher implementation overhead compared to FSMs.

In short, FSMs are best for simple, predictable flows, while EFSMs are essential when transitions depend on both events and dynamic data. Each approach serves a different need within the LMS project, depending on the complexity of the feature.

### Application Examples in the LMS Project

In the LMS project, both Finite State Machines (FSMs) and Extended Finite State Machines (EFSMs) were used to model different workflows, depending on how complex they are and what data they need.

The login process is done with a simple FSM because it follows a clear sequence of steps and does not need to store extra variables except for the user’s role.

The process starts in the **LoggedOut** state, when the user is not yet logged in. When the user clicks the login button, the system goes to the **Authenticating** state to check the login details. If the details are correct, the FSM moves to one of the logged-in states: **LoggedInStudent**, **LoggedInInstructor**, or **LoggedInAdmin**, depending on the user’s role. If the details are wrong, the FSM goes back to **LoggedOut**.

At any time, if the user logs out or the session ends, the system goes back to **LoggedOut**. This FSM keeps the logic simple and clear, and makes role-based navigation smooth.

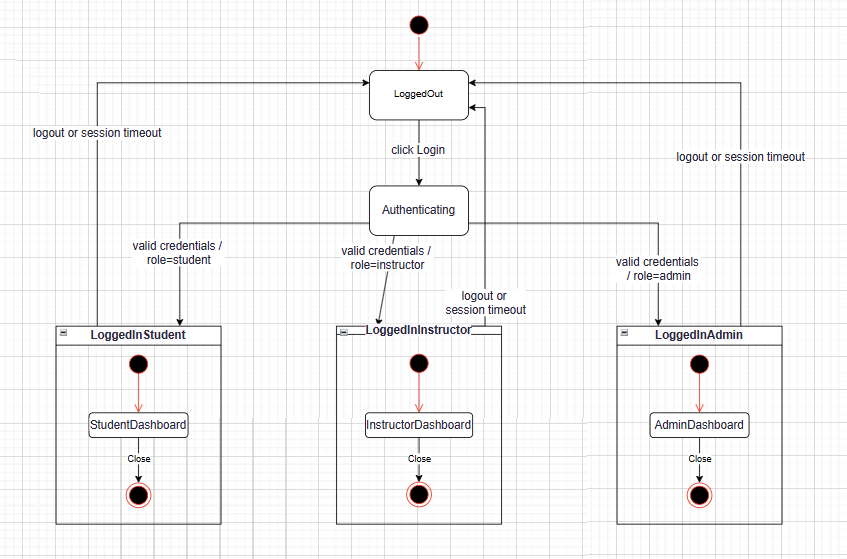


Figure 23: FMS: Login Process

The course enrollment process uses an EFSM because it needs to check stored variables before making a change. Before a student can enroll, the system checks two variables: **userRole** (what type of user they are) and **isEnrolled** (if the student is already enrolled).

The process starts in **ViewingCourses**. If **userRole = "student"**, clicking the enroll button moves the system to **Enrolling**. Here, the EFSM checks **isEnrolled**:

* **If isEnrolled = false**, the system adds the enrollment, gives access to materials, and goes to **Enrolled**.
* **If isEnrolled = true**, the system goes back to **ViewingCourses** and shows “Already enrolled”.

If **userRole ≠ "student"**, clicking enroll shows “Access denied” and stays in **ViewingCourses**.

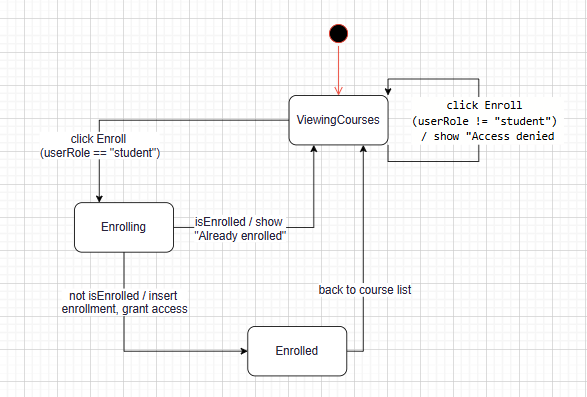


Figure 24: EFSM: Course Enrollment (with userRole and isEnrolled)

**Analysis in LMS:** The FSM for login is easy to maintain and makes role-based navigation simple. The EFSM for enrollment gives the flexibility needed to handle conditional logic, making sure only the right users can enroll and avoiding duplicate enrollments. By using FSM for simple tasks and EFSM for more complex ones, the LMS project balances simplicity and flexibility, making the workflows both clear and effective.

## Present justifications of how data driven software can improve the reliability and effectiveness of software (D4)

### Reliability and Effectiveness

In the LMS project, a data-driven design helps improve both the reliability and effectiveness of the system by basing key decisions and workflows on structured data stored in the database.

First, **scalability under increased loads** is supported because core functions such as login authentication, course creation, and enrollment are all powered by database queries rather than hardcoded rules. This means that when the number of students, courses, or instructors grows, the system can handle more requests simply by optimizing database performance (e.g., adding indexes, using caching), without needing major code changes.

Second, **adaptability to changing requirements** is achieved through the use of configurable database tables and role-based access logic. For example, if a new role (such as “Teaching Assistant”) needs to be added, it can be implemented by inserting a new role record and adjusting permission data, rather than rewriting large parts of the system. This makes it easier to respond to stakeholder requests or institutional policy changes.

Third, **error reduction through automation** is possible because repetitive tasks, such as checking if a student is already enrolled or verifying password hashes, are performed automatically by database queries and server-side logic. This reduces human error, ensures consistent results, and increases trust in the system’s outputs.

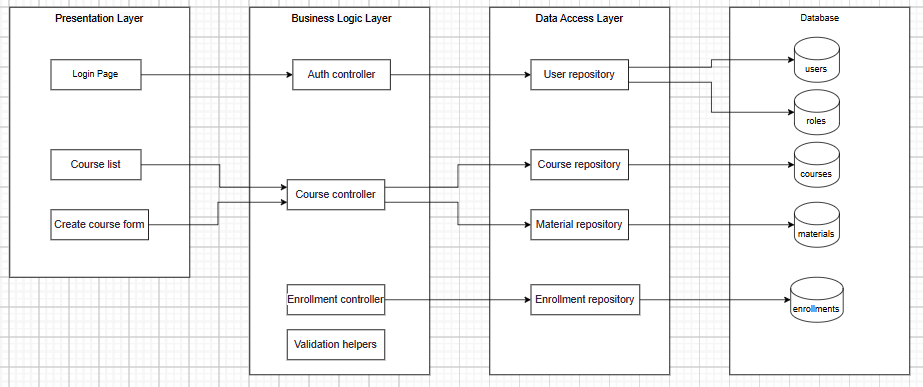


Figure 25: 3 Layer Architecture of LMS

### ****Implementation Strategies****

To integrate a data-driven approach effectively into the LMS project, the following strategies were applied:

1. **Centralized Database Management:** All critical information, including user accounts, courses, enrollments, and role permissions, is stored in a relational database. This ensures data consistency and makes it easy to update information in one place.
2. **Dynamic Role and Permission Handling:** User permissions are determined at runtime by reading the user’s role from the database. This avoids hardcoding access rules and allows new roles or changes to be applied instantly.
3. **Parameterized Queries and Stored Procedures:** Using prepared statements for database interactions not only improves security against SQL injection but also ensures that queries are optimized and reliable under heavy use.
4. **Data Validation at Multiple Levels:** Input data is validated both on the client side (JavaScript) and server side (PHP) before being stored, reducing the chance of invalid or corrupt data entering the system.
5. **Modular Design for Data Access:** Database access functions are kept separate from presentation logic, making the system easier to maintain and adapt without affecting the core data structure.

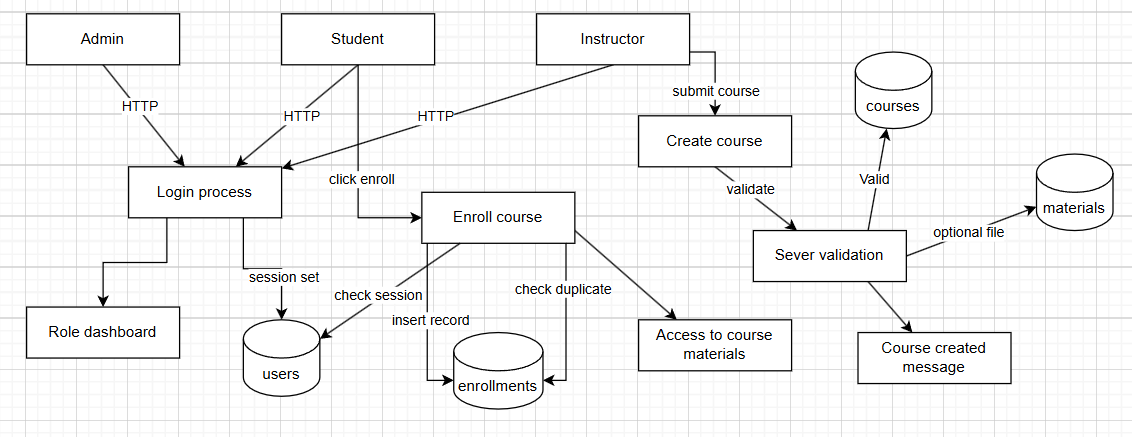


Figure 26: Data Flow Diagram for LMS Processes

By following these strategies, the LMS project ensures that data remains accurate, workflows are efficient, and the system can grow and adapt over time without sacrificing performance or reliability.

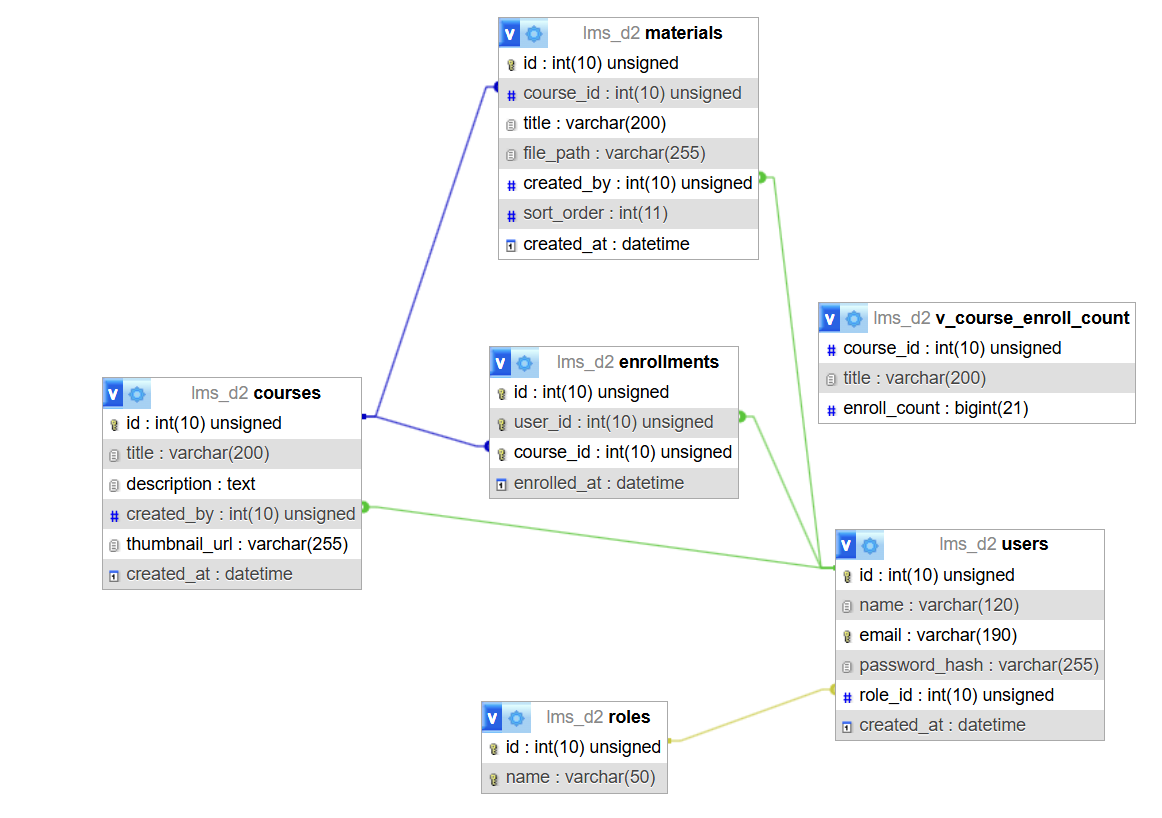


Figure 27: ER Diagram (Core Tables)

The **v\_course\_enroll\_count** view is designed to provide a quick overview of how many students are enrolled in each course within the LMS system. It retrieves the course ID and title from the courses table and counts the corresponding enrollment records in the **enrollments** table. By using a **LEFT JOIN**, the view ensures that courses with zero enrollments are still included in the result set, showing an enrollment count of **0**. This view is particularly useful for administrators and instructors to track course popularity, monitor participation levels, and make informed decisions about course offerings or resource allocation. Because the data is generated dynamically from the database, it stays up to date without requiring manual updates or recalculations.

# Conclusion

The development of a university LMS requires a structured yet flexible approach to balance technical feasibility, user expectations, and institutional goals. Through a detailed comparison of SDLC models and technical solutions, this report demonstrates that Agile offers the most appropriate framework for the project. Its iterative nature allows for continuous stakeholder involvement, rapid feature delivery, and adaptability to evolving academic requirements. Practical applications, such as refining the course creation and enrollment modules based on real user feedback, highlight the effectiveness of this approach in aligning the final product with the needs of lecturers, students, and administrators.

Furthermore, the use of decision-making tools such as cost-benefit analysis and decision matrices ensures that development choices are based on objective criteria, optimizing both budget efficiency and system quality. By adopting Agile and applying these analytical methods, the LMS project mitigates risks, improves user satisfaction, and ensures long-term scalability. This strategy positions the system as a sustainable, adaptable, and integral part of the university’s digital learning ecosystem, capable of evolving alongside technological advancements and academic trends.

# References

1. European Union (2016) General Data Protection Regulation (GDPR). [Online] Available at: https://gdpr.eu/ [Accessed 1 July 2025].
2. Highsmith, J. (2010) Agile Project Management: Creating Innovative Products. 2nd ed. Boston: Addison-Wesley.
3. Pressman, R. S. and Maxim, B. R. (2014) Software Engineering: A Practitioner’s Approach. 8th ed. New York: McGraw-Hill.
4. Sommerville, I. (2016) Software Engineering. 10th ed. Boston: Pearson.
5. MoodleDocs (2025) About Moodle LMS. [Online] Available at: https://docs.moodle.org/ [Accessed 2 July 2025].