VIA University College

Project Title - Project Report

Semester Project 3

Group 3

Students
Guillermo Sanchez Martinez (355442)
Piotr Wiktor Junosz (355502)
Halil Ibrahim Aygun (355770)
Alexandru Savin (354790)
Eduard Fekete (355323)

Supervisors
Joseph Chukwudi Okika (JOOK)
Jakob Trigger Knop (JKNR)

Character Count: 31405
Word Count: 4593

Software Technology Engineering

3rd Semester

December 16, 2025

Contents

1 Abstract

This work focuses on the need for scalable and adaptable digital educational solutions by developing the Learnify system, a distributed software solution with the objective of ensuring seamless content delivery and user assessment. The primary goal is to make a durable multi-server solution that leverages the power of multiple programming languages for enhanced availability, data integrity, and optimized user response. This system was built with a polyglot microservices architecture, using programming languages such as Java and C# to ensure a strict interoperability implementation. Some of the key technical design choices are the use of gRPC for high-speed intra-service communication and HTTP for external client services, ensuring both speed and accessibility. Data storage is achieved using a PostgreSQL database, ensuring that security is taken into account using JWT authentication to overcome the potential dangers of distributed systems. This application development occurred using an iterative approach, centering around User Stories developed both using specialized analysis and interviews with the stakeholders. A functional prototype that can manage multiple user sessions concurrently in a distributed manner was developed. This prototype was verified to meet essential non-functional requirements, confirming the successful interoperability of components across Java and C#. Security compliance was established through the validation of salted password hashing algorithms using Argon2, ensuring robust data protection alongside reliable persistence in the PostgreSQL database. Moreover, the interface was validated for conformance with recognized accessibility guidelines to support complete usability by those with color vision defects. Finally, the project achieved a fully deployable distributed system, validating that the architecture provides a secure, stable, and operational foundation for scalability in the future of education.

2 Introduction

Acquiring new knowledge is essential part of human life and evolution. Living in the population equals communicating within it and that requires some minimum level of knowledge.(REFERENCE) The idea of mandatory education keeps its origin between late 1900s and early 2000s(REFERENCE), nonetheless around 40% of the global population still does not have access to proper education in a language they understand (PTI, 2025).

The aim of this project is on the ability to create a system which would be able to provide learning opportunities with main focus on simplifying the accessibility and exploring the idea of learning processes and its speed and efficiency. The goal is to also ensure security, knowledge correctness and deployment of the system.

With the pursuit of knowledge having been a cornerstone of human development for thousands of years, the incorporation of digital technologies into education is in perpetual evolution (Siemens, 2005). A widely accepted model for learning with digital technologies has not been identified, mainly because exponential increases in computing power and volumes of online information constantly redefine how users approach knowledge acquisition, processing, and retention (Haleem et al., 2022).

The approach of this project is to develop a distributed system implemented using at least 2 different programming languages, utilizing a database for data persistence, and adapting a hybrid communication strategy that includes technologies such as gRPC and HTTP.

3 Main Section

In the main section development experience of the "Learnify" e-learning platform is being documented. It describes the importance of essential phases that took place in the development process from the vision to a working software system. This chapter is organized around the core stages: Analysis, Design, Implementation, and Testing, ensuring that each were necessary while building the system.

This chapter starts by elaborating Analysis phase, which incorporates the functional and non-functional requirements, use case diagrams, use case descriptions activity diagrams, threat model as well as the domain model of the system. The design section then focused on forming each parts of the system's architectures and database design, including the communication methods such as gRPC and HTTP. The Implementation stage converted the blueprints into components in two programming languages - C# and Java - that exhibit the integrated logic of the system. Finally, Testing were performed to ensure the system behaves according to the requirements and also meets the general objectives of the project.

3.1 Analysis

In the beginning of the technical documentation Analysis processes of the the distributed system - "Learnify" - are described. This stage applies the higher-level goals of the project to the real challenges that come with the multi-language distributed system.

For defining the scope of the proposed system with clarity, the initial idea was thoroughly refined using requirements. This involved the identification and definition of the functional requirements of the system as well as non-functional ones, including latency, scalability, and security, which are the most important requirements of the distributed learning system. In addition, several interviews with potential clients were done in order to gather new data which helped creating and updating the requirements and then use cases. This stage of analysis also involved the use of structured modeling. The Use Case and Activity Diagrams were used for the representation of user interaction and system behavior, and the Domain Model defined the conceptual framework for the design of data persistence and service integration.

3.1.1 Requirements

3.1.1.1 Functional requirements

1. As a User, I want to register for an account so that I can access the platform.
2. As a User, I want to log in so that I can access the platform from my account.
3. As a User, I want to view "My Courses" dashboard, so that I can see all the courses I am currently enrolled in.
4. As a User, I want to resume my active course from where I left off, so that I can continue learning from where i left.
5. As a User, I want to browse the "All Courses" catalog, so that I can discover new subjects to learn.
6. As a User, I want to filter courses by category or search query, so that I can find specific content quickly.
7. As a User, I want to unenroll from a course, so that I can remove content I am no longer interested in from my dashboard.
8. As a User, I want to view the Leaderboard, so that I can compare my progress with other learners.
9. As a User, I want to view my Profile, so that I can see my personal account details.
10. As a User, I want to complete Multiple Choice Questions in a learning step, so that I can test my knowledge.
11. As a User, I want to complete Fill-in-the-Blank exercises, so that I can practice recalling information.
12. As a Teacher, I want to create a new course draft, so that I can start building a new curriculum.
13. As a Teacher, I want to edit learning steps directly from the learning page, so that I can correct mistakes or improve content.
14. As an Admin, I want to view the Admin Panel, so that I can access administrative tools and settings.
15. As a Teacher, I want to edit course information, so that I can correct mistakes.
16. As an Admin, I want to view a list of waiting drafts, so that I can see which courses need approval.
17. As an Admin, I want to approve course drafts, so that they become available for students to enroll in.

18. As an Admin, I want to create new course categories, so that new types of courses can be introduced.
19. As an Admin, I want to add new languages, so that Teachers can create courses in other available languages.
20. As an Admin, I want to manage users' roles, so that I decide who is a teacher and admin.
21. As an Admin, I want to disapprove course drafts, so that they become unavailable for students to enroll in.

3.1.1.2 Non-functional requirements

1. The system must utilize at least two different programming languages
2. User passwords must be stored securely, using a strong, salted hashing algorithm before persistence
3. The system must use a robust database for all persistent data storage.
4. The system has to be deployable
5. The system has to be color blind people friendly

The requirements were taken from not only the initial vision of the system but also from interviews with real persons that could actually experience the system in the development stages and give relevant feedback to the team. The functional requirements were structured as "user stories" so that the team could maintain clear user perspective development, ensuring that every feature was directly tied to who is making what interaction with the system and how does the user is going to benefit from performing the action - why would he do it. This provided consistent framework for later analysis phases.

3.1.2 Use case diagram

The use case diagram visible on the Figure X below shows all use cases and their actors, which were made out of functional requirements. The goal of the project was to include security within the system by using authorization and authentication. Consequently, because of that all use cases require an authenticated user. In addition, two internal actors, Teacher and Administrator, were designed to inherit from Learner, while also having extra use cases which are not accessible for Learners.

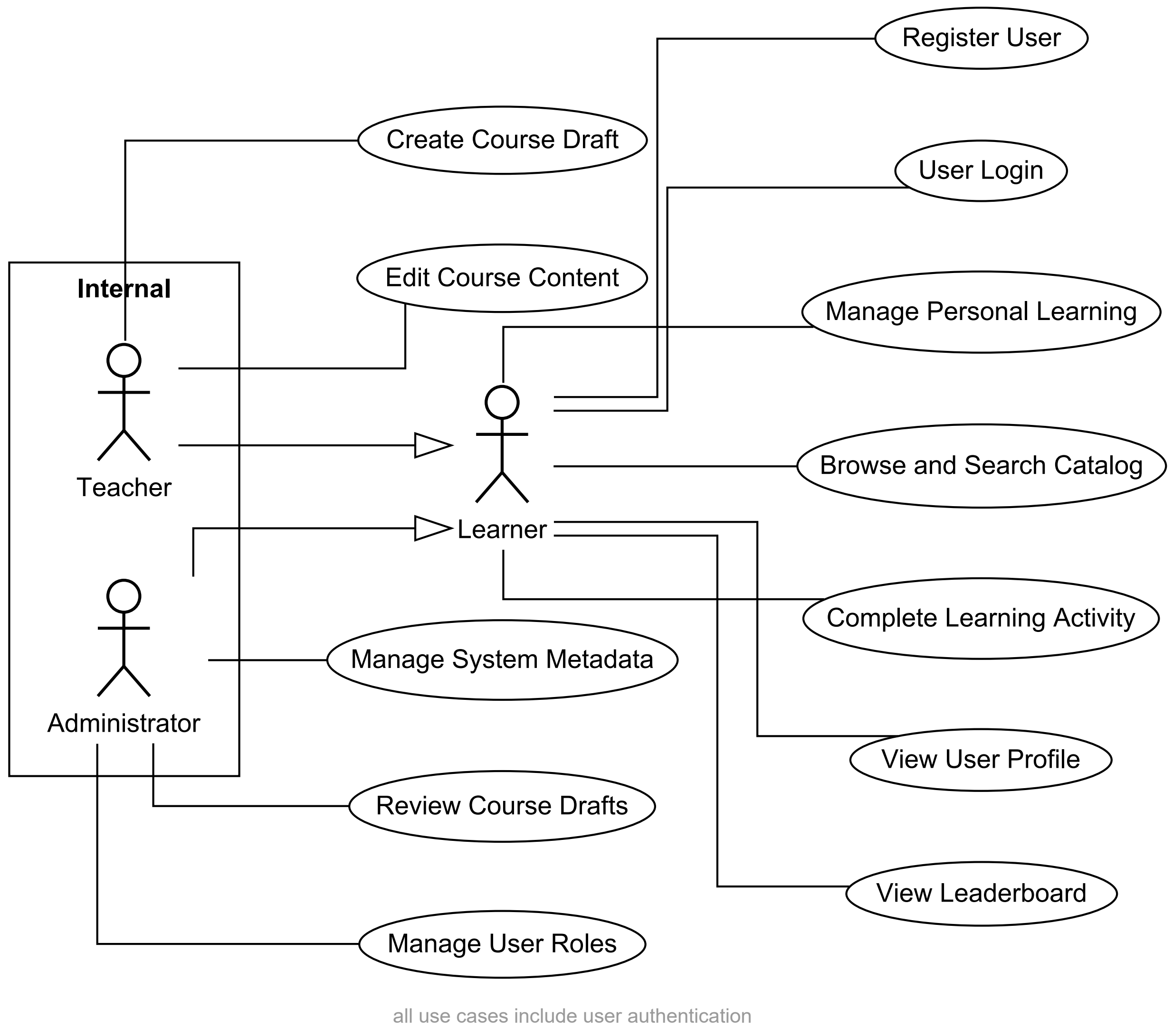
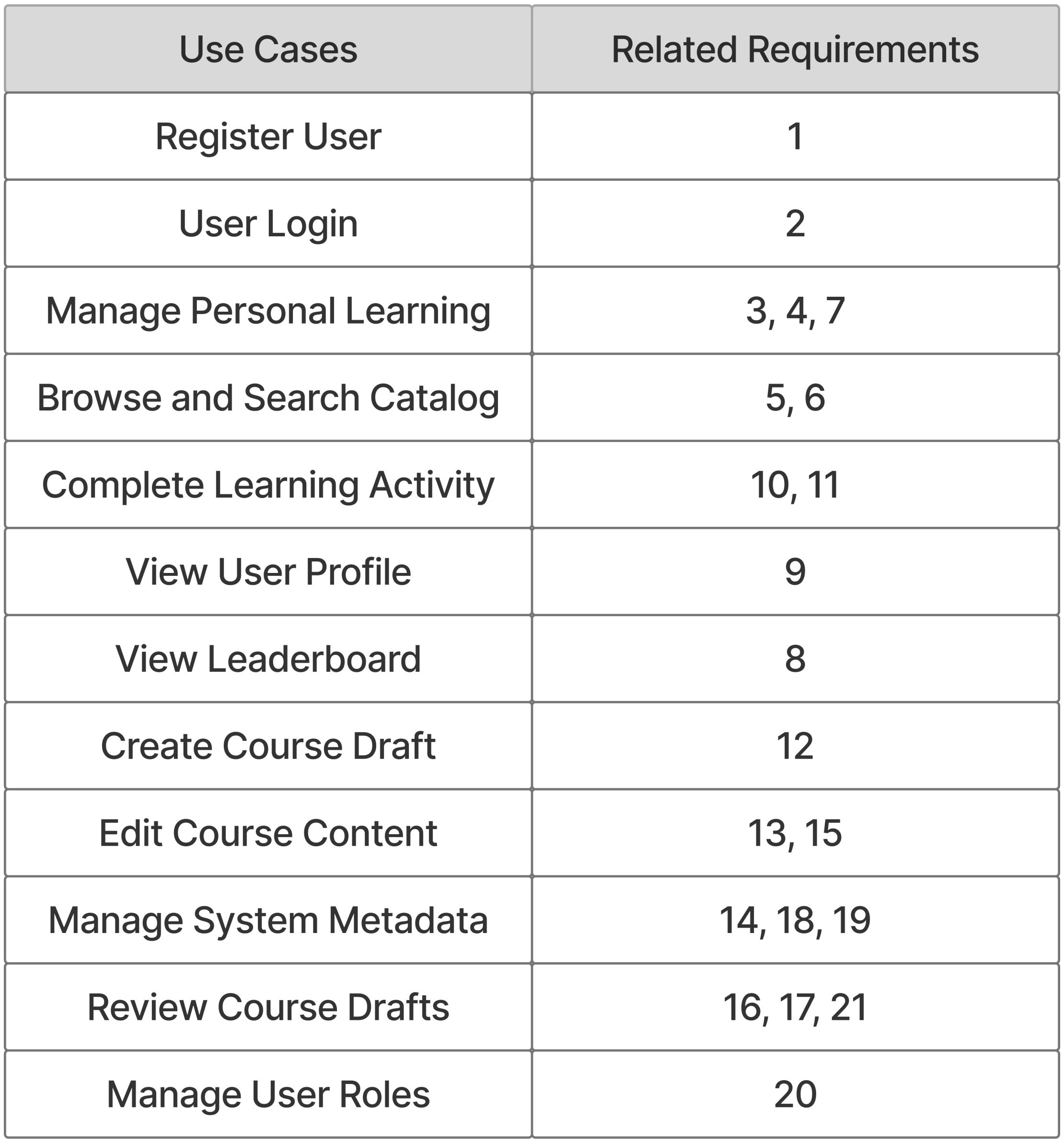


Figure 1: UseCaseDiagram

3.1.3 Use cases and their related requirements

The table on the Figure below shows the connections between use cases and the requirements they cover.

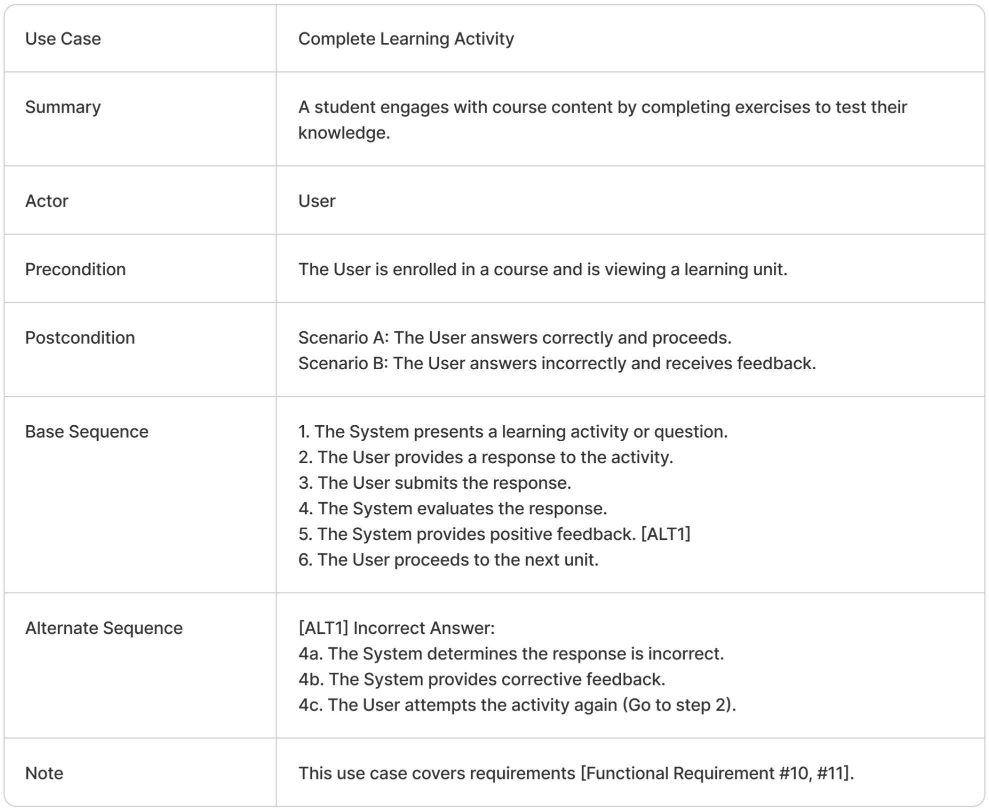


Use Cases	Related Requirements
Register User	1
User Login	2
Manage Personal Learning	3, 4, 7
Browse and Search Catalog	5, 6
Complete Learning Activity	10, 11
View User Profile	9
View Leaderboard	8
Create Course Draft	12
Edit Course Content	13, 15
Manage System Metadata	14, 18, 19
Review Course Drafts	16, 17, 21
Manage User Roles	20

Figure 2: Use Cases with their related requirements

3.1.4 Use case description

Complete Learning Activity Use Case Description shown on the figure presents that use case description were made with precision by focusing on different scenarios and alternate sequences while also sometimes covering more than one requirement.



Use Case	Complete Learning Activity
Summary	A student engages with course content by completing exercises to test their knowledge.
Actor	User
Precondition	The User is enrolled in a course and is viewing a learning unit.
Postcondition	Scenario A: The User answers correctly and proceeds. Scenario B: The User answers incorrectly and receives feedback.
Base Sequence	1. The System presents a learning activity or question. 2. The User provides a response to the activity. 3. The User submits the response. 4. The System evaluates the response. 5. The System provides positive feedback. [ALT1] 6. The User proceeds to the next unit.
Alternate Sequence	[ALT1] Incorrect Answer: 4a. The System determines the response is incorrect. 4b. The System provides corrective feedback. 4c. The User attempts the activity again (Go to step 2).
Note	This use case covers requirements [Functional Requirement #10, #11].

Figure 3: Complete Learning Activity Use Case Description

3.1.5 Activity diagram

The activity diagram below shows the workflow of the given “Complete Learning Activity” use case. The diagram highlights the sequence of operations performed by the user and the system, which begin with the display of a question or an exercise. Validation logic is highlighted in this activity diagram where the system analyzes the response of the user. Based on this, if a wrong solution is provided, a “feedback loop” is initiated where a message prompts the user to try again. As a result, a learning activity will be marked accomplished only when a correct answer is given, which demonstrates a mastery learning technique.

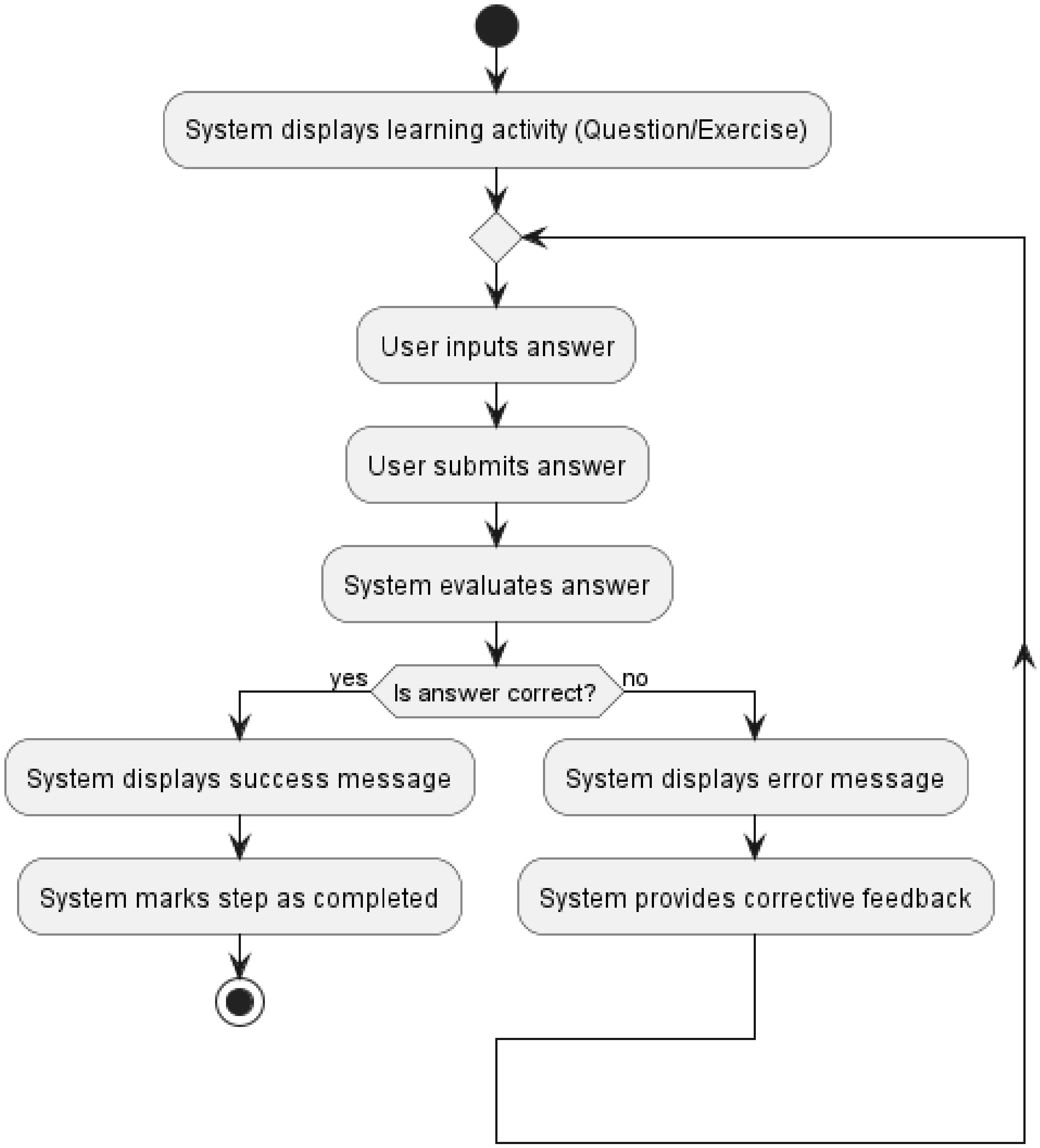

System displays learning activity (Question/Exercise)
User inputs answer
User submits answer
System evaluates answer
Is answer correct?
yes
no
System displays success message
System marks step as completed
System displays error message
System provides corrective feedback

Figure 4: Complete Learning Activity Activity Diagram

3.1.6 Domain model

The domain model was constructed for this project to better understand the problem domain and to aid communication among stakeholders. The crucial aspect of developing the domain model was identifying the relationships between different kinds of users, in particular the roles and responsibilities of Learners, Teachers, and Administrators; which had to be combined with the security aspect of the system as well as had to align with the shared understanding of the stakeholders.

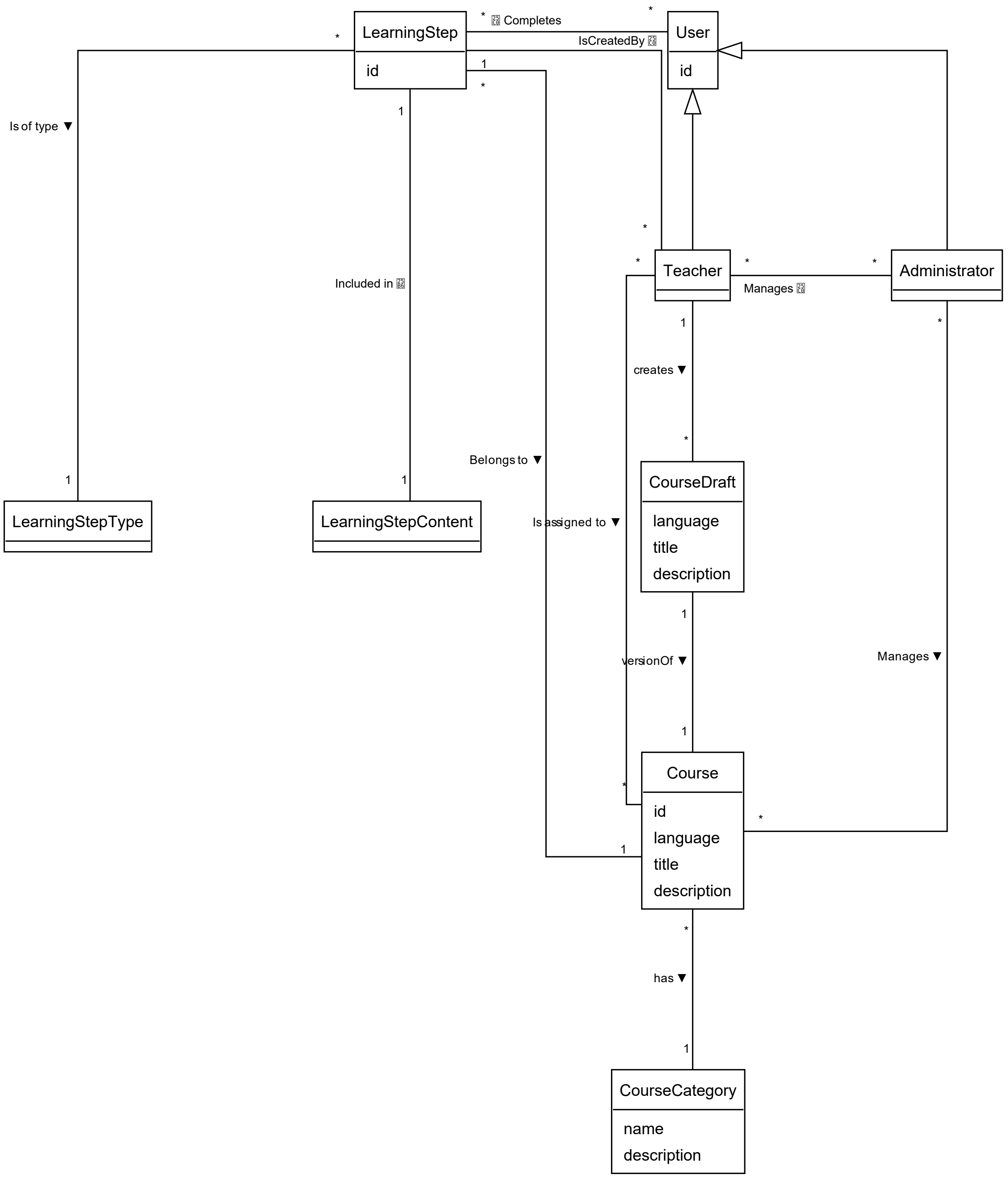


Figure 5: Domain Model

The figure above shows the domain model for Learnify. It can be seen that the crucial aspect of fully describing the different system roles is understanding what entities exist and how they relate to each other.

The inheritance from the User entity reflects the fact that all system users have some common basic attributes, rights and behaviours that can be generalized.

The philosophy behind the attribute selection and abstraction into separate entities was to provide full flexibility for future development of the system, and ensuring that no restrictions are imposed for no reason. For example, Learning Steps are abstracted into three different entities to fully support any kind of idea of a learning step as seen both in the domain and from stakeholder interactions.

At the same time, the domain model was kept in its simplest form possible in terms of more abstract entity concepts. As can be seen on the domain model, the core aspects of the system are:

- Users
- Courses
- Learning Steps

And it could be further argued that Learning Steps only exist as a part of Courses, therefore the Domain Model is centered around the idea of Users learning from Courses, which did not change from the initial vision of the system.

3.1.7 Security Requirements

The security requirements for the system were developed as a part of the threat modelling process. The security objectives were as follows:

- **Confidentiality:** Protect user passwords and personal data from unauthorized disclosure.
- **Integrity:** Ensuring that data can not be altered or tampered with by unauthorized parties.
- **Availability:** Ensure the system remains accessible during high traffic or denial-of-service attempts.
- **Accountability:** Actions must be uniquely traceable to a specific entity.
- **Authenticity:** Verify that data inputs and users are genuine.

These objectives were developed based on the CIA triad and expanded to fit the needs of the system (Appendix A: Threat Model).

Similarly to other parts of the analysis phase, stakeholder interactions shaped the system's security requirements. In particular, even testing the prototype (before having the core system functional) showed concerns about the authority of Administrators and Teachers, and needs for accountability for actions.

3.2 Design

3.2.1 System design

3.2.2 Architectural overview

The architectural overview shown on the picture below presents how the three-tier architecture of the system was looks like including all servers and how they communicate between them. Starting with client layer, which is responsible for running a server in C# Blazor .NET, it can be seen that its job is to host a web application which can be accessible by three types of users (Learners, Teachers and Administrators). Client application communicates with Logic Server, located inside the logic layer, by using HTTP requests and responses. Then from the Logic server information is being sent further into the data server, located inside data tier, which happens by following the gRPC protocol, which is faster than HTTP due to different formatting. Logic server was implemented using C# and Data server using Java. At the end of the architecture chain we have the Postgres database. The data is received through sockets. Although the three-tier overview seems to appear a bit basic, each tier plays their own important role in the system, ensuring that for example data server is not responsible for any feature logic but only performs operations between the database.

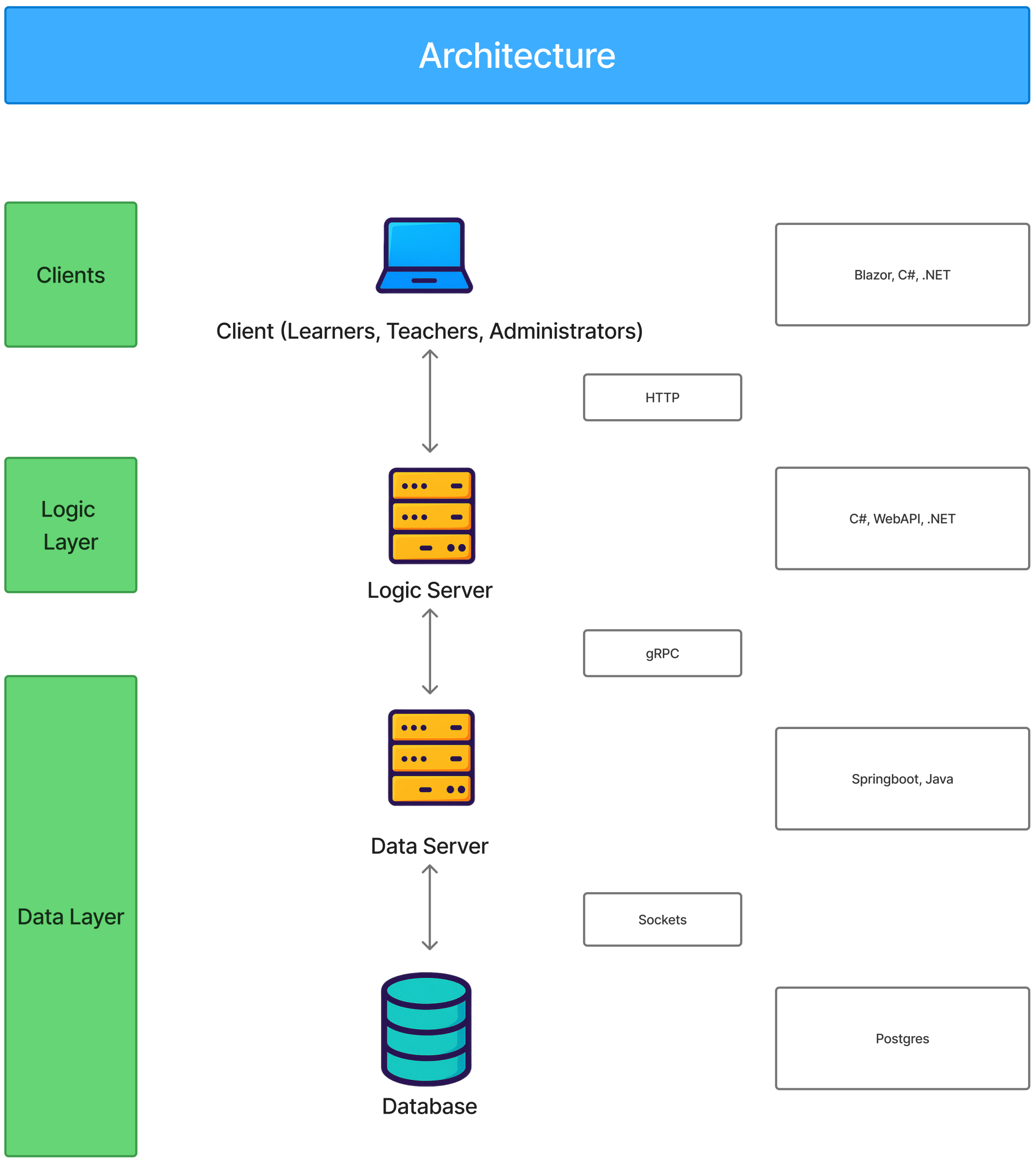


Figure 6: Architectural Overview

3.2.3 Communication protocol design

*Interface Definition (IDL): Show snippets of your .proto files (if using gRPC).

API Specification: Briefly describe the HTTP endpoints (e.g., RESTful routes).

Protocol Choice: Explain why gRPC was used for internal communication vs. HTTP for external (or however you structured it).* ### Database design

3.2.3.1 Enhanced Entity Relationship Diagram As means of bridging the gap from the problem domain in general and the Learnify system in specific, an EER diagram was created as can be seen on the figure below:

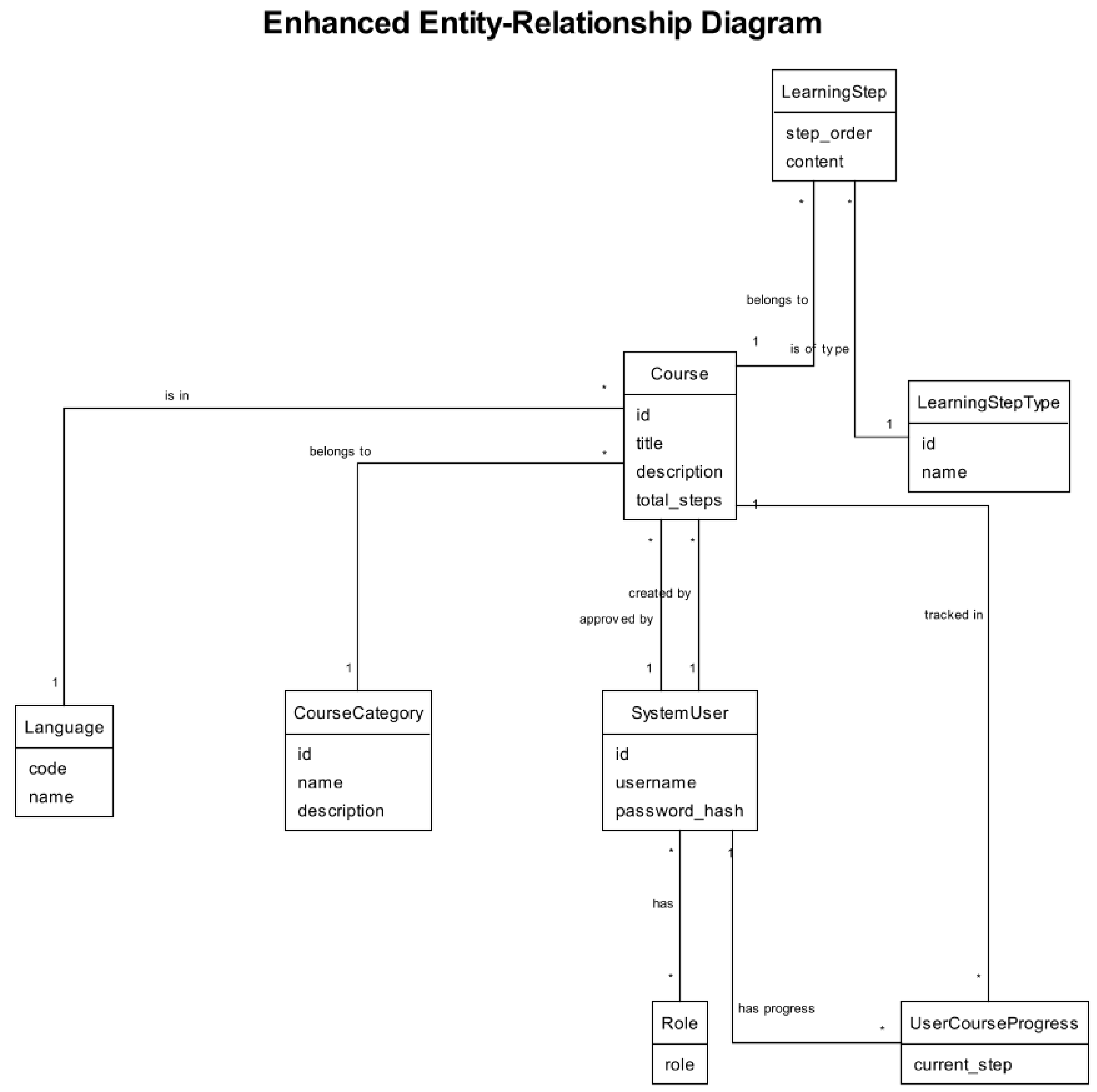
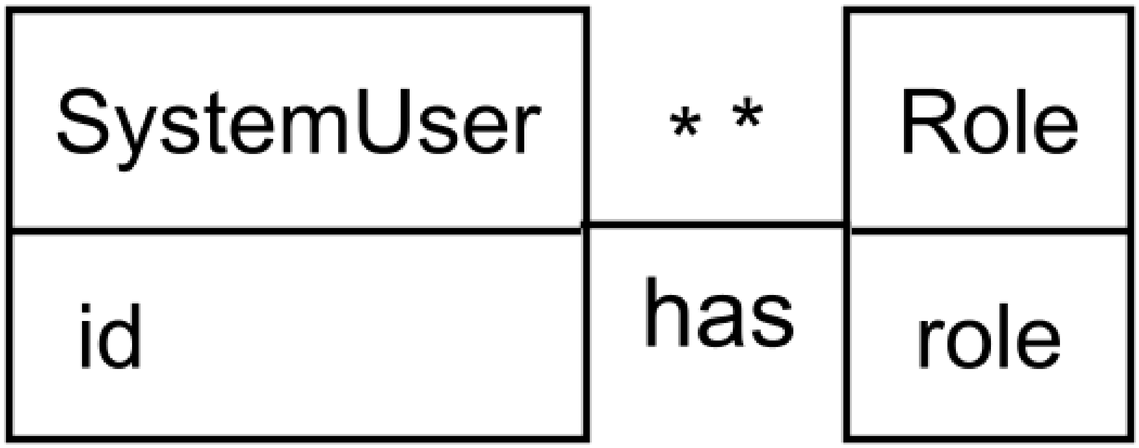


Figure 7: Enhanced Entity Relationship Diagram

The EER developed does not significantly differ from the domain model as both diagrams are conceptual and could in theory be used interchangeably. However, the EER diagram further reflects the decisions made during analysis, which most notably reflected on the way how roles are handled.

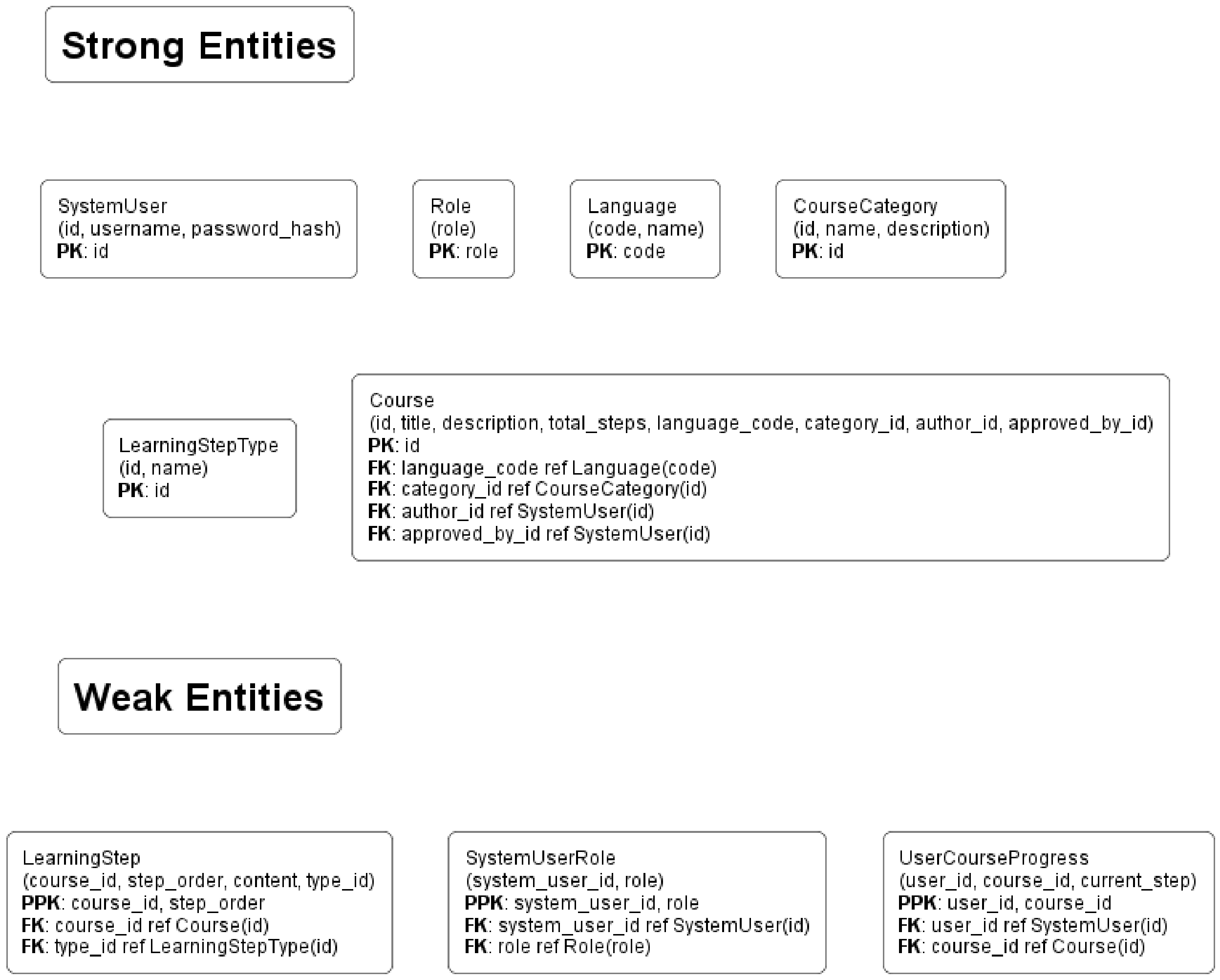


The figure above focuses on the relationship between the User and their Roles. This relationship in contrast to inheritance based models provides a flexible and strict way of handling user roles - their permissions and access to the system. Most importantly it does not hide the complexities of inheritance into a seemingly simple abstraction and prevents the potential issues that could arise from mindless inheritance hierarchies.

3.2.3.2 Relational Schema Contrary to the conceptual modelling, the logical modelling required adherence to the rules and specificities of the relational model. Because of the designed use of standard relational database (PostgreSQL), the mapping of the EER needed to determine all the necessary resolutions of relationships, strong and weak entities, and the establishment of integrity keys.

The mapping of the EER diagram resulted in a relational schema and the to it related global relations diagram.

The mapping resulted in the relation schema as shown on the figure below:



Strong Entities

SystemUser
(id, username, password_hash)
PK: id

Role
(role)
PK: role

Language
(code, name)
PK: code

CourseCategory
(id, name, description)
PK: id

LearningStepType
(id, name)
PK: id

Course
(id, title, description, total_steps, language_code, category_id, author_id, approved_by_id)
PK: id
FK: language_code ref Language(code)
FK: category_id ref CourseCategory(id)
FK: author_id ref SystemUser(id)
FK: approved_by_id ref SystemUser(id)

Weak Entities

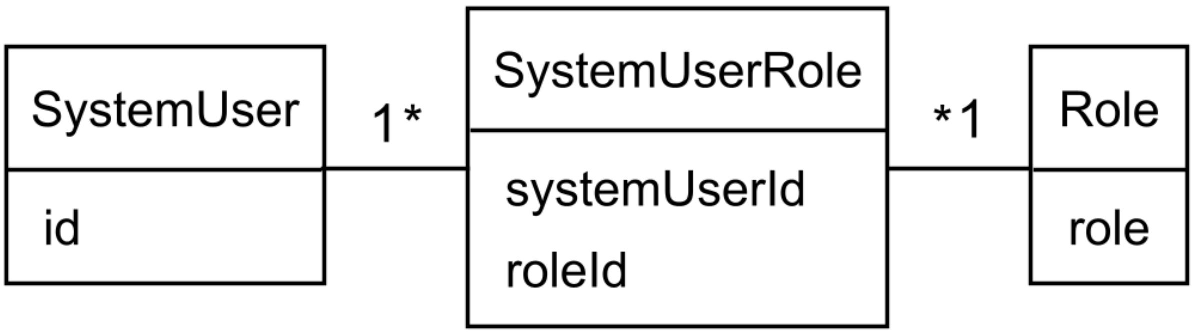
LearningStep
(course_id, step_order, content, type_id)
PPK: course_id, step_order
FK: course_id ref Course(id)
FK: type_id ref LearningStepType(id)

SystemUserRole
(system_user_id, role)
PPK: system_user_id, role
FK: system_user_id ref SystemUser(id)
FK: role ref Role(role)

UserCourseProgress
(user_id, course_id, current_step)
PPK: user_id, course_id
FK: user_id ref SystemUser(id)
FK: course_id ref Course(id)

Figure 8: Relational Schema

As can be seen, the many-to-many relationships got resolved into new relations - SystemUserRole, and UserCourseProgress.



3.2.3.3 Global Relations Diagram (GR/GRD) Relational schema and Global Relations Diagram are technically identical. In the project Learnify, the GRD was the main source of truth for later implementation of the database structure and discussing the data persistence design.

The final GR diagram is shown on the figure below:

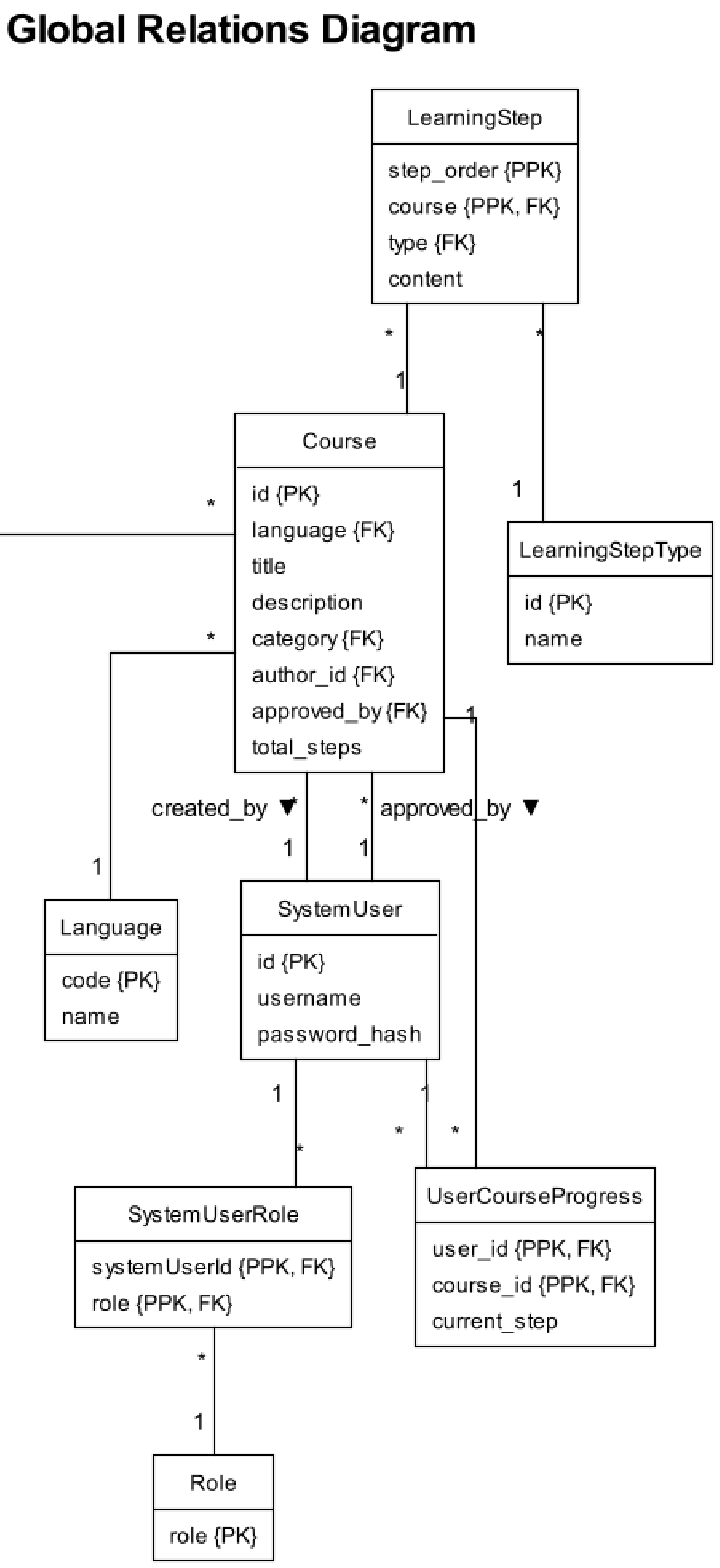


Figure 9: Global Relational Diagram

It can be seen that the GRD reflects the same concepts as the relational schema, however, in this case the positioning of the elements provides a more natural step going from the EER and the domain model; although, the positioning did not fully preserve the conceptual relationships as seen in the EER diagram.

3.2.4 Class diagram design

3.2.5 Communication Protocol Design:

3.2.6 Data Persistence Design (maybe we should add some design related to this?):

ER Diagram: showing how data is structured in the database.

Consistency Model: Since it is distributed, mention how you handle data integrity across services.

3.3 Implementation

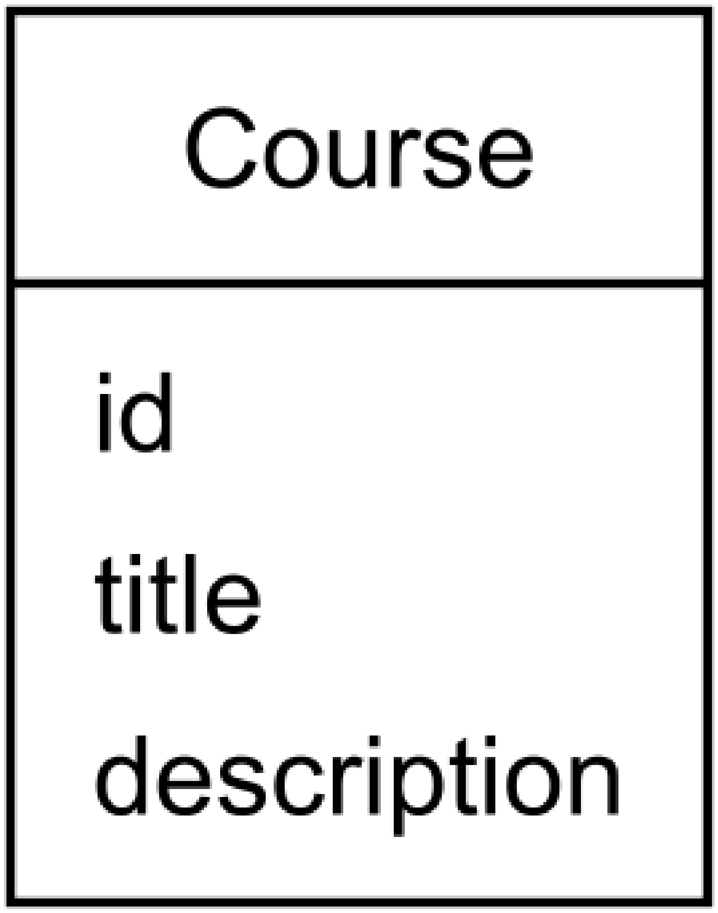
The implementation of the system followed the designed architecture and communication protocols with a focus on setting up the core of the architecture first as one continuous vertical slice.

At first, a database schema was created in PostgreSQL, a Springboot project was created for the Data Server, and two .NET solutions were created for the Logic Server and the Client Application.

This stage did not include any actual logic but rather provided a skeleton of the system. One of the decisions taken at this stage was to maintain separate solutions for the Logic Server and Client Application. Despite the initial idea of implementing a shared solution, it was decided that the feature of C# anonymous types would suffice for most of the purposes of data transfer objects (DTOs) and that the added complexity of a shared solution would not be justified.

The implementation of the skeleton was followed by a the implementation of a vertical slice which focused on fetching all courses from the database.

The individual components of the vertical slice can be seen below:



At a later stage, the original database setup was split into pure DDL script and a dummy data initial setup crucial for testing stages mentioned later in this document.

3.3.1 Methods and tools

List the two languages (e.g., Go, Java, Python) and the database (e.g., PostgreSQL, MongoDB).

Justification: Explain why each language was chosen for its specific task. (e.g., “Go was selected for the backend service due to its concurrency handling. . . ”).

3.3.2 Server A Implementation (Language 1):

3.3.3 Server B Implementation (Language 2):

3.3.4 Server C Implementation:

3.3.5 Integration Logic:

Show how the two services “talk” to each other. Provide a code snippet showing the gRPC client/server handshake or the HTTP request handling.

3.4 Testing

3.4.1 Testing Approach

Testing in this project was structured around the V-Model, where testing activities are planned in parallel with developments phases. Instead of treating testings as a final step, test considerations were introduced early, starting at the requirement analyses, and refined as the system design evolved. This approach ensured that development decision had a corresponding verification strategy.

The first step of the V-Model was requirement analysis, which directly maps to acceptance testing.

At this stage, testing focused on answering a simple but critical question:

How can we tell that the system fulfills the user's requirements?

Rather then writing code-level tests, the team defined high-level acceptance criteria for each use case. These criteria described:

- What user expects to achieve
- Under which conditions the system should allow or deny actions
- What outcome confirms that the requirement is fulfilled

These acceptance-oriented test ideas were later used to validate the system both manually and through automated test. In this way, test cases were already embedded in the analysis phase.

In practice, different types of tests naturally aligned with different parts of the V-Model:

- High-level test cases and use-case validation corresponded to the upper part of the V (requirements and acceptance testing).
- Automated tests focused mainly on the lower part of the V, especially unit and integration testing.

Automated tests focused mainly on the lower part of the V, especially unit and integration testing.

3.4.2 Tools and frameworks

A combination of automated and manual testing tools was used throughout the project:

- xUnit / JUnit – Used for writing automated unit and integration tests.
- Mockito / Mocking frameworks – Used to isolate logic and mock external dependencies.
- HTTP client–based tests – Used for endpoint-level testing.
- .http files – Used early in development for manual REST endpoint testing.
- BloomRPC – Used for manually testing gRPC endpoints.
- In-memory repositories – Used to test logic without external dependencies.
- IDE tooling (Visual Studio / IntelliJ) – Used for writing, executing, and debugging tests.

3.4.3 What was tested

Testing focused primarily on core logic and system-critical behavior, rather than attempting full coverage of all UI components.

The following aspects of the system were tested:

- Business logic in the logic server
- REST and gRPC endpoints
- Authentication and authorization behavior
- Role-based access control
- Course and draft management workflows
- Integration between services

The team aimed for high endpoint coverage, testing endpoints under different scenarios such as:

- Different user roles and credentials
- Authorized vs. unauthorized access

- Expected success paths

However, testing of faulty or malformed input data was less comprehensive. While access control and permissions were thoroughly tested, negative scenarios involving invalid data were sometimes under-tested. This limitation is acknowledged as an area for improvement.

3.4.4 Method-level test case documentation

3.4.5 Benefits and bug detection

3.5 Result

The guidelines require you to support results with data, programs, or models.

3.6 Final Product Showcase: Screenshots of the "Learnify" app UI or console logs showing successful data processing.

3.7 Ethical Considerations:

Requirement: You must describe ethical considerations and how negative impacts are minimized.

Content: Discuss data privacy (GDPR), user consent, or the societal impact of the app.

---

4 Discussion

4.0.0.1 What Has Been Accomplished The main purpose of this project has been achieved through the creation of the "Learnify" distributed heterogeneous e-learning system which targets the problem of educational inequality. The system operates through a three-tier architecture which combines a C# Blazor client application with a Logic Server built in C# .NET and a Data Server developed in Java. The PostgreSQL database functions as the storage system which maintains all relevant information for every service. The system architecture uses gRPC protocol to establish fast communication between services while it employs HTTP protocol to enable client application interaction. The system has achieved full integration of 21 specific user stories which support the complete learning process. The system enables users to create new accounts through its registration feature. Users can browse course catalogs. Users can take part in courses. The system provides users with suitable interactive learning activities that deliver instant feedback. The system enforces a rigid role-based structure which applies to all users including Learners and Teachers and Administrators. The role hierarchy establishes a governance process which requires Teachers to obtain administrator approval before their content can be published. The security implementation utilizes industrial standards with Argon2 for Password Hashing with salting and JWT for Stateless Authentication. In addition, the User Experience has been strengthened with the incorporation of leaderboards for gamification and a Color Validated User Interface for color-deficient users. The overall requirements have been designed in accordance with the primary values laid down in the project and potential customers' feedback.

4.0.0.2 What Can Be Improved Even with the successful implementation of the fundamental system, there are some aspects that need optimization towards the reduction of the identified security risks and the improvement of scalability. The basic application needs additional security measures to protect against advanced attacks even though it uses Argon2 password hashing and JSON Web Tokens. The Critical risk of unpatched software vulnerability exploitation requires automated scanning and effective patch management for future developments. The protection of high-risk systems against Man-in-the-Middle attacks and credential exploitation needs Transport Layer Security/Secure Sockets Layer combined with HTTP Strict Transport Security and Content Security Policies to prevent Cross-Site Scripting attacks and Multi-Factor Authentication. The system has passed normal functional tests to meet operational requirements but its performance under severe stress remains untested through extreme load testing. The risk assessment shows that Distributed Denial of Service requires future research to simulate high user volumes for identifying system bottlenecks.

Concerning user experience, there is still a lot of room for improving the inclusivity of the platform. Although the support for color vision disabilities is a good starting point, the support needs to be extended to implement full compatibility with the Web Content Accessibility Guidelines and screen readers for users with motor disabilities. In addition to this, the format of the learning model needs to change from the linear pattern that it currently supports to an adaptive path and also an advanced gamification system incorporating features like badges and streaks. Lastly, to avoid administrative delays that could arise in the future due to the expansion of users, it is advised that the manual content reviewing system should be integrated with an AI-powered content review system.

---

5 Conclusion and Recommendations

The main goal of this study was to develop a scalable heterogeneous distributed e-learning system which tackles educational inequalities through superior performance and system integrity, while delivering an excellent user experience. The system development process required a three-tier architectural design which implemented a multi programming language approach by combining C#, Java, and the PostgreSQL database. The tests showed that the system operates effectively when handling multiple sessions and processing data.

The system delivered its core objectives but failed to achieve complete functionality because content moderation features remained unimplemented within the project timeframe. The system establishes a core structure which enables the creation of a distributed system.

Future research should direct its attention toward creating strong security systems which include Multi-Factor Authentication and advanced permission frameworks because the existing basic systems would need to evolve for commercial deployment. The learning platform will become better after the implementation of full accessibility features which include screen reader support and adaptive layout functionality. The platform requires testing for production-level usability and maximum load capacity to achieve production readiness and acceptable latency levels.

The project achieved success through its development of a distributed learning system which met all essential criteria established during the planning phases. However, there are still some areas which needs space for an improvement. The results of this study could possibly help educational institutions improve their technology systems which currently exist in schools. Digital learning systems produce various results which need to be evaluated during the evaluation process. The development team needs to resolve privacy issues and content recommendation problems and digital divide problems for future Learnify learning platform versions.

---

6 References

- Haleem, A., Javaid, M., Qadri, M. A., & Suman, R. (2022). Understanding the role of digital technologies in education: A review. Sustainable Operations and Computers, 3(1), 275–285. https://doi.org/10.1016/j.susoc.2022.05.004
- Project Description.
- PTI. (2025, March 2). 40% global population doesn't have access to education in language they understand: UNESCO. Deccan Herald. https://www.deccanherald.com/world/40-global-population-doesnt-have-access-to-education-in-language-they-understand-unesco-3428194
- Samonas, S., & Coss, D. (2014). The Cia Strikes Back: Redefining Confidentiality, Integrity and Availability in Security. In Journal of Information System Security (Vol. 10, Issue 3). https://www.proso.com/dl/Samonas.pdf
- Siemens, G. (n.d.). Connectivism: A Learning Theory for the Digital Age. https://static1.squarespace.com/static/6820668

7 Appendices

7.1 Appendix A: Threat Model

Can be found as ThreatModel.pdf