Software Requirements Specification

for

An Interactive Web Application for Programming Learning Analytics

Prepared by

Nurperi Momunalieva 2643187 and

Zarrintaj Ahmadzada 2644037

Middle East Technical University Northern Cyprus Campus

Computer Engineering

06.04.2025

Contents

Introduction

1.1 Purpose:

This project focuses on developing an Interactive Web Application for Programming Learning Analytics, showing stages of the software development process. These stages include defining system requirements such as functional and non-functional requirements, designing the user experience, and making sure the platform meets the needs of students and instructors after registration.

The web application is designed to offer introductory programming courses by providing students with interactive exercises, a personalized recommendation system, and chatbots to help with programming exercises and learning. The goal is to create a supportive environment for students as they practice coding and improve their skills.

Instructors, on the other hand, will have the ability to set up courses, add programming exercises, and enroll students. They will also be able to track student progress through detailed reports, allowing them to provide better guidance and adapt their teaching based on these reports. All of the services provided above should ensure a comfortable learning platform for students.

1.2 Scope:

The goal of this project is to develop a web application designed for students taking introductory programming courses and their instructors. The platform will offer features that enhance the learning experience, providing guidance and a user-friendly environment to help students achieve better outcomes in their studies.

The objectives of this project include collecting and securely storing student performance data, such as accuracy of solutions, error patterns, and encountered challenges, to generate detailed reports. The application will analyze this data to identify general patterns and common mistakes, enabling instructors to refine their teaching strategies. A personalized recommendation system will suggest actions that students and instructors can take for their improvement. Additionally, there will be a chatbot that will provide real-time support, answer programming-related questions, and provide guidance on navigating the application efficiently.

This project aims to create a more personalized and targeted learning experience, making it easier for students to overcome challenges and instructors to optimize their teaching methods.

1.3 Related Work

With the fast growth of the computer engineering field, a lot of programming learning course applications have been created. Among such applications, the biggest ones are LeetCode, CodeAcademy, 3wSchool, CodeFinity, and freeCodeCamp. Although all of these web applications are directed to offer programming courses, there are some differences.

w3Schools allows users to access website features without registration. Such features are accessing articles and videos, solving programming exercises, etc. W3Schools also offers alternative ways to register, like on our website. In addition to Google, users have a chance to register through GitHub, Facebook, and Feide.

Unlike our system, which offers to register as a student or an instructor in the first step, LeetCode is a platform that is oriented to one user who is learning programming, and the user is not offered to register as an instructor making it less suitable for classroom use or guided learning environments. Our application addresses this by supporting both student and instructor roles, enabling course creation, enrollment, and learning analytics. Additionally, our system integrates a **chatbot** to provide instant assistance, helping students resolve doubts and continue learning without interruption. Another platform, W3Schools, only offers registration as a student. To become an instructor, a user needs to access another form on the website and fill out personal information and make a payment.

Most websites like CodeAcademy, LeetCode, FreeCodeCamp do not offer the feature of programming exercises connected to an online editor that tracks the solutions and errors. Moreover, our platform incorporates an **advanced plagiarism detection tool**, ensuring the integrity of assignments and preventing cheating, which is a feature not commonly found in most other programming education platforms.

With the rapid growth of the computer engineering and programming education fields, a wide range of web-based platforms have emerged to help learners acquire programming skills. Some of the most prominent platforms include **LeetCode**, **Codecademy**, **W3Schools**, **Codefinity**, and **freeCodeCamp**. While all these applications are designed to support learning programming concepts, they differ in scope, user access, and support for instructors.

W3Schools provides a straightforward learning environment where users can access tutorials, videos, and programming exercises without registering. However, registration is available for users who want to save their progress, and the platform supports multiple third-party sign-in options, including Google, GitHub, Facebook, and Feide. In contrast to our system, which requires users to register either as a student or an instructor from the outset, W3Schools only offers a general user account by default. Instructor access must be requested separately through a different form and includes a payment step.

LeetCode is highly focused on practice for coding interviews and algorithm challenges. It provides categorized exercises based on topic or difficulty and includes contest modes and leaderboards. However, LeetCode is designed for individual learners and does not offer support for instructors or class-based structures. Users cannot register as instructors or manage student groups, making it less suitable for classroom use or guided learning environments. Our application addresses this by supporting both student and instructor roles, enabling course creation, enrollment, and learning analytics.

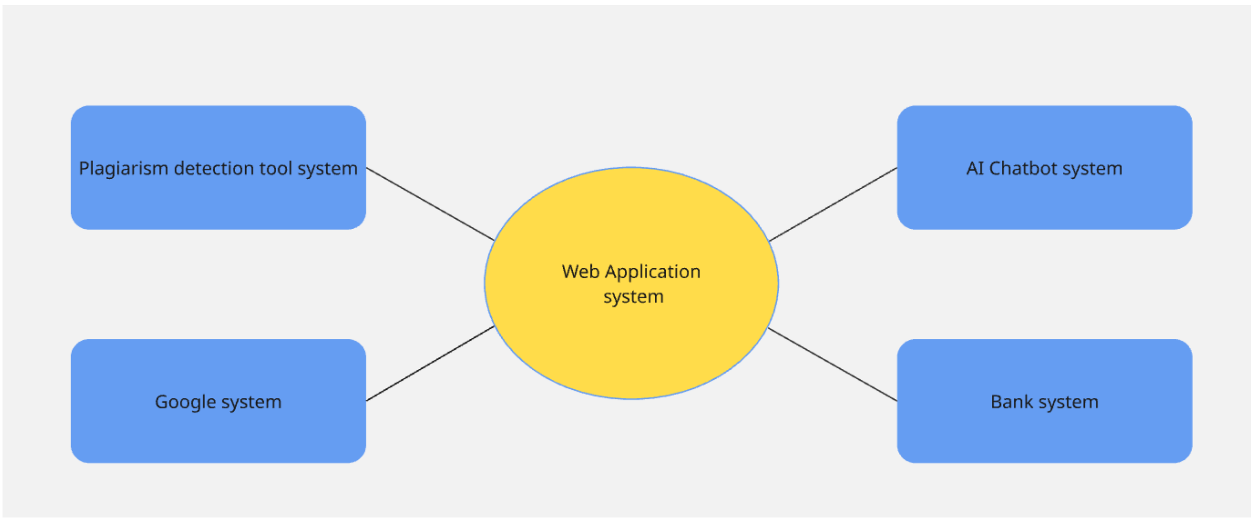
Codecademy and **Codefinity** offer guided programming courses with interactive lessons and project-based learning. While these platforms provide structured learning paths, they are subscription-based and generally lack detailed analytics tools for instructors. Students can track their own progress, but instructors do not receive automatic performance reports or suggestions to improve student outcomes. In contrast, our system is focused on delivering personalized learning insights to both students and instructors, backed by real-time performance data.

freeCodeCamp provides a vast curriculum and is entirely free to use. It includes video tutorials, interactive coding challenges, and project-based certifications. While highly accessible, it is largely self-paced and does not support instructor management, class creation, or group-level analytics. Unlike freeCodeCamp, our platform is designed to function in a structured academic setting, with instructor tools, performance tracking, and automated reports.

In summary, while these platforms contribute significantly to programming education, most are either designed for individual learners or lack real-time learning analytics and instructor-focused features. Our application fills this gap by combining real-time performance tracking, personalized recommendations, chatbot assistance, and course management tools, making it highly suitable for structured, data-driven programming instruction.

1.4 Product overview

1.4.1 Product perspective



1.4.1.1 System Interface

1. Google Authentication API

The system allows users, both students and instructors, to sign up using their Google accounts as an additional option to enhance the overall user experience. This integration provides a secure and seamless authentication process, simplifying registration and login

2. Plagiarism Detection Tool

The application integrates a third-party plagiarism detection tool that analyzes student code submissions to ensure the originality of their work. It detects similarities between current and previously submitted solutions, helping to maintain academic integrity. The tool identifies matching lines of code, calculates a similarity score, and generates a detailed report highlighting any instances of copied content along with the corresponding similarity percentages.

3. Chatbot (via a third-party API)

The web application includes an integrated chatbot via third-party API that helps users with coding questions, programming concepts, and platform navigation. It provides real-time support to enhance the learning experience and use of system

4. Online Editor

The Online Editor tracks the code students are writing, monitoring whether they solve the exercises correctly, how many attempts they make, and the types and frequencies of errors.

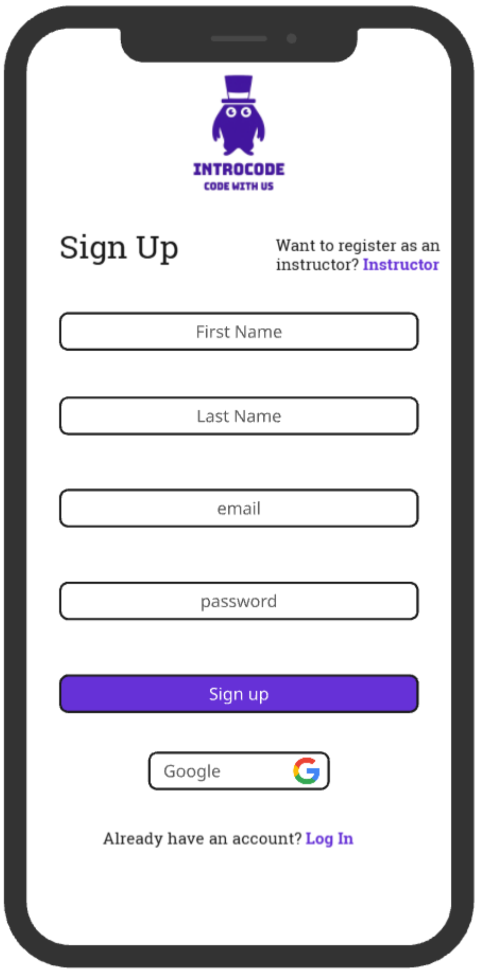
Once the code compiles successfully and produces the expected output, students can submit their work for review. Based on this data, the system provides recommendations to guide students in improving their skills.

5. Bank

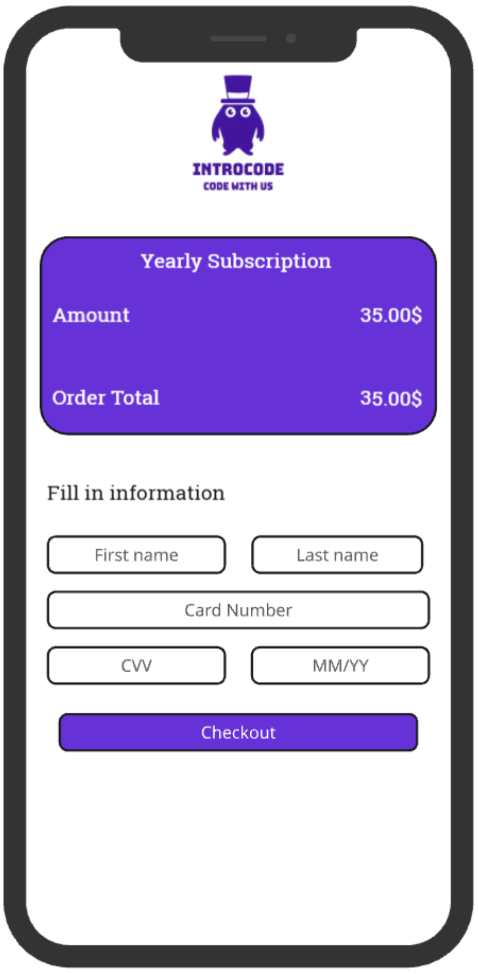
The application uses a secure third-party Bank API to handle transactions during instructor registration. This makes sure that all payments are safely handled and verified. By using an external service, the system doesn't store sensitive payment information, which eliminates the need for the application to manage sensitive financial information directly.

1.4.1.2 User Interfaces

Sign up for students


INTROCODE
CODE WITH US
Sign Up
Want to register as an instructor? Instructor
First Name
Last Name
email
password
Sign up
Google
Already have an account? Log In

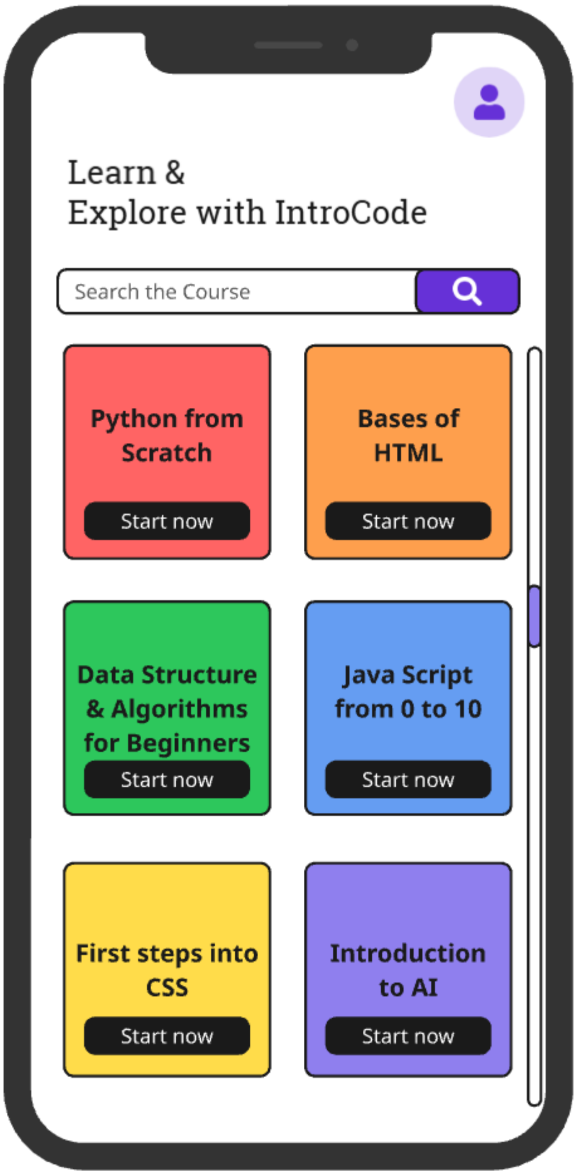
Payment system for instructors



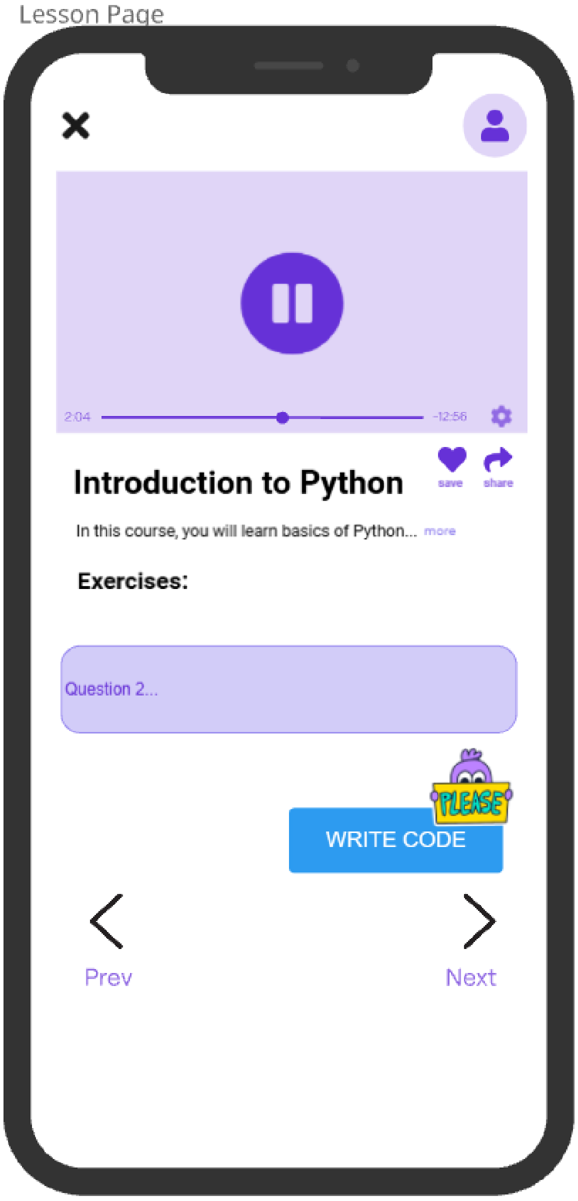
1. This is the first window when accessing the web application. The user must register in order to access the features. There is an alternative registration through Google. Also possible to register as an instructor.

2. Payment for instructors is linked to the bank

Course catalog for students



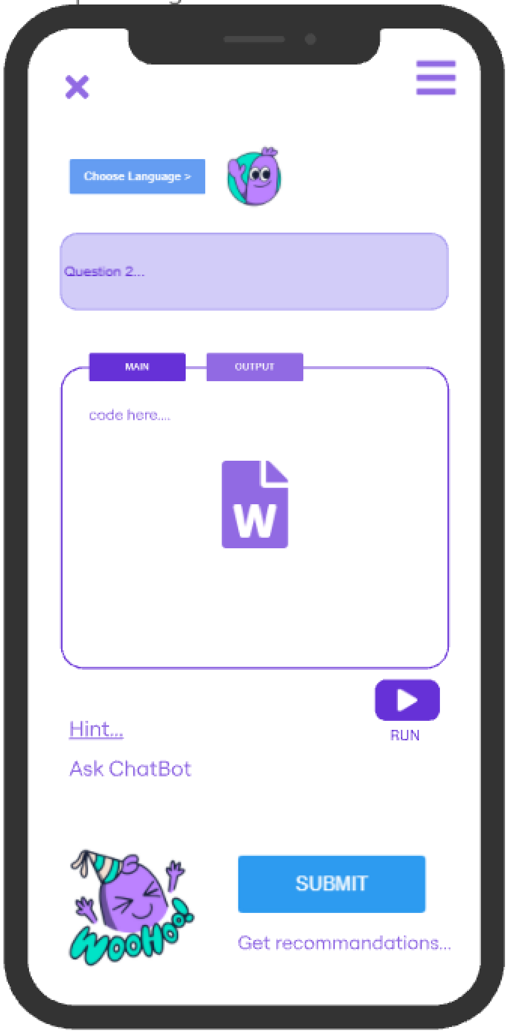
Lesson Page



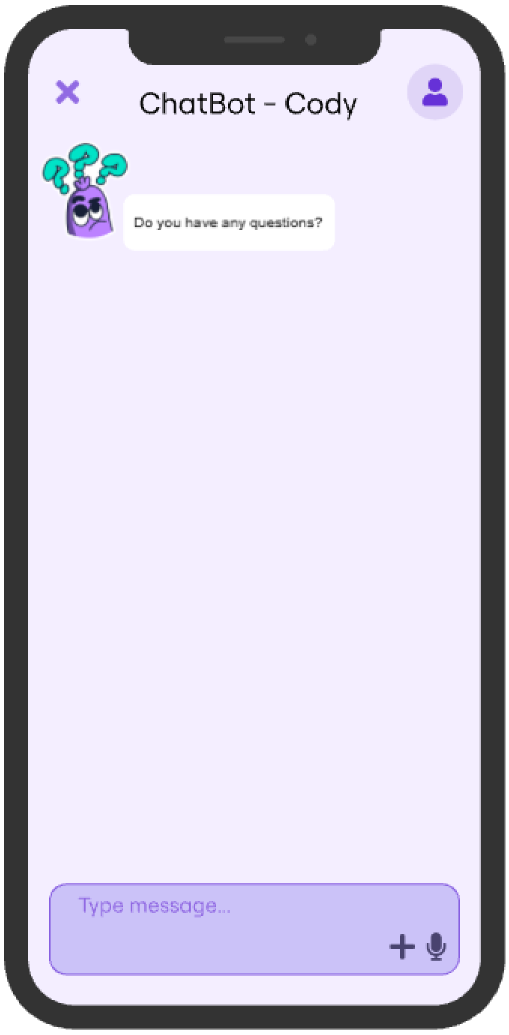
3.This is the course catalog for students that consists of different introductory courses. Students can search for the needed course with the help of a search bar.

4. After starting the course, the student has access to lessons that consist of videos and programming questions about the given video. A student can write his answer by pressing “Write Code” that opens a compiler. As well as scroll to the next question.

Compiler Page



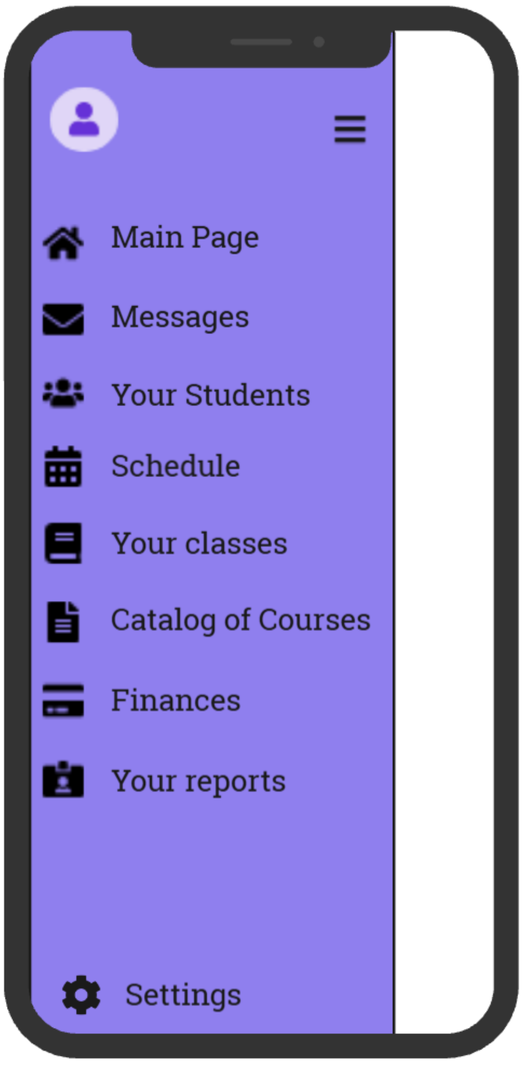
ChatBot



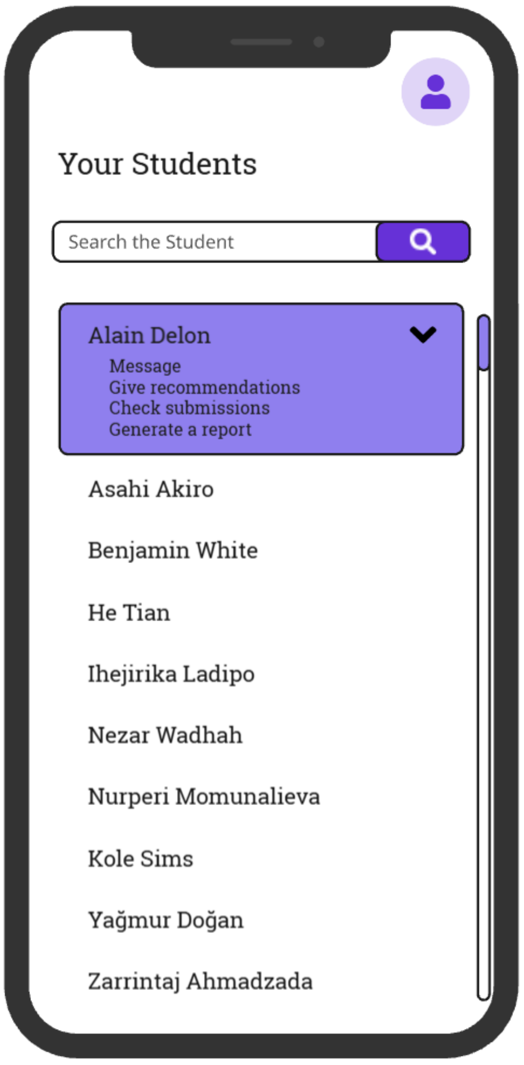
5. This window allows to choose programming language and use the compiler. After running the code the output can be seen in the "output" window. Student has an option of getting a hint, asking the chatbot, and submitting the code if it compiles.

6. Chatbot can be accessed through the manual menu or when solving the exercises. It has an option of uploading files, documents, and pictures, and recording a voice message.

Menu for instructors



Student window for instructors



7. Menu for instructors allows them to go to "Your Students" (to find the specific students). "Your courses" shows courses that the instructor teaches(Students can be added here for each course), settings, etc.

8. This window shows all students that are taking instructor's classes. It is possible to message, give recommendations, check submissions and generate report for each student.

1.4.2 Product Functions

The primary function of this web application is to provide an interactive and supportive environment for students learning introductory programming courses while equipping instructors with effective tools for course and student management. Students can solve coding exercises through an integrated compiler with the help of an online editor, that tracks their progress, such as correctness, attempts, and error types. Based on this data, the system offers personalized recommendations to guide students in improving their skills. The application also includes a built-in chatbot that delivers real-time assistance, helping students with programming concepts and navigating the platform.

Instructors are able to create and manage multiple courses, add and edit programming exercises, enroll students, and view detailed performance reports. They also receive recommendations for course improvements based on collected student performance data. Additional key features include secure registration with Google authentication and integrated plagiarism detection to ensure the integrity of student submissions.

1.4.3 User characteristics

The intended users of this web application are primarily two groups: students and instructors

Students are generally expected to be university-level learners or beginners with basic computer knowledge and limited programming experience. Their technical expertise may vary widely, so the platform is designed to be user-friendly, with a simple interface and helpful features such as real-time feedback, error tracking, and chatbot assistance. The system accommodates users who may struggle with new technologies or programming concepts by offering clear guidance and personalized learning recommendations.

Instructors are typically more experienced in programming and teaching, generally professors or assistance. Their primary needs include managing courses, tracking student progress, and interpreting performance data, all of which the platform supports.

1.4.4 Limitations

This project has several limitations that may affect its development and usage. First, students cannot enroll themselves into courses; enrollment must be done by instructors, either individually or in chunks. Second, in order to access instructor features, users must go through a verification process and complete a payment, which may limit accessibility for some users. Also, the system relies on third-party tools which can have their own limitations in usage. Additionally, the platform is strictly focused on programming education and does not support other subjects.

1.4.5 Assumptions and Dependencies

1. **Browser Compatibility**: It is assumed that users will access the web application through modern and updated web browsers such as Chrome, Firefox, or Edge

2. **Internet Access**: The system assumes all users will have a stable internet connection, as the application is fully web-based and requires online access for all features.

3. **Instructor Verification**: It is assumed that instructors registering on the platform will have valid credentials and complete the required verification and payment process to access instructor-level features.

4. **Third-Party Services Availability**: The chatbot and plagiarism detection features as well google verification and payment transaction rely on third-party APIs. It is assumed that these services will remain available, functional, and within any usage limits.

5. **Basic User Competency**: It is assumed that students have basic digital literacy and understand fundamental computer operations, even if they are new to programming.

6. **Server Hosting**: The system assumes that both the web application and the database will be hosted on the same internal server for performance and

security purposes.

7. **Course Scope**: It is assumed that the platform will only be used for introductory programming courses and not for teaching advanced or non-programming topics.

1.5 Definitions

N/A

2 Specific requirements

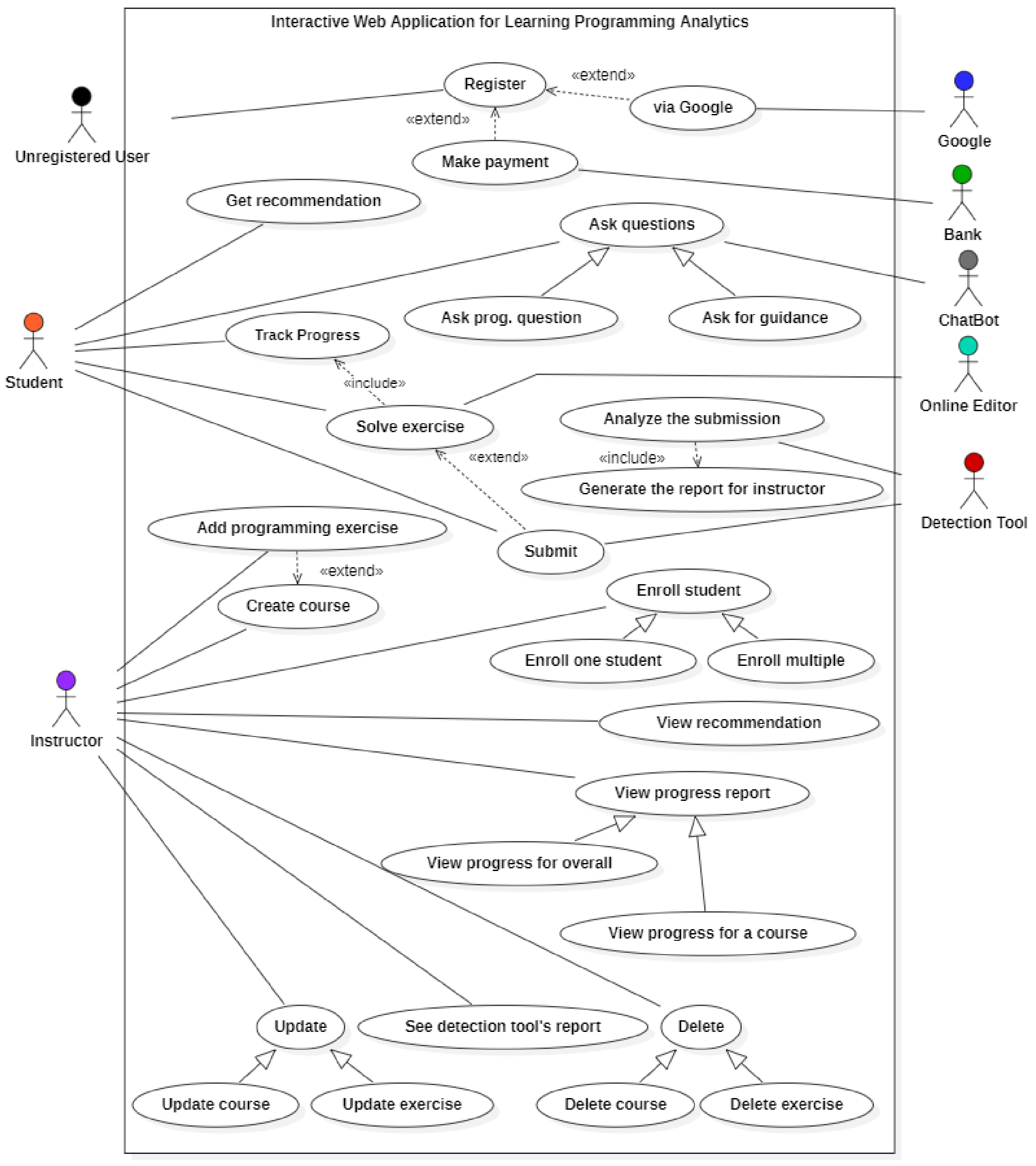
2.1 External interfaces

System Interface	**Functionality**	**Input**	**Output**
Google Authentication API	Allows users to sign up using Google by redirecting them to the Google page	Google account	Authentication result code: 0: success 1: invalid Google Account
Plagiarism Detection Tool	Compares the code that the student has submitted with previously submitted codes. Highlights similar lines and calculates the similarity percentage.	Code	Plagiarism Detection code: 0: successfully produced report with highlighted lines and a similarity percentage
ChatBot	Answers to programming questions and guides for application use	Question	Answered Message code: 0. successful Reply 1: error due to Policy Restrictions
Online Editor	Tracks the written code in the compiler for errors and records errors, number of attempts, and frequencies.	The Code Previously Submitted work	Editor result code: 0. success

Bank	Ensures safe payment transactions by redirecting to the third-party Bank page	Price Name Surname Card number, CVV Expiration date Bank Company Account number	Transaction result code: 0: success 1: insufficient amount 2: invalid card number 3: invalid security code 4: expired card 5: unknown problem

2.2 Functions

UML Use Case Diagram:



Detailed Use Case Descriptions:

Use case	Register
Actors	Unregistered User, Bank, Google
Cross references	N/A
Typical Course of Events	
Actor Intentions	System Responsibility
1. An unregistered user opens the webpage	
	2. The system shows the login window
3. An unregistered user enters their information	
	4. The system approves the registration
	5. The system redirects to the main page
Alternative Courses	
Step 3a: An unregistered user chooses to register as an instructor and gets redirected to another window to make a payment, after that returns to step 2	
Step 3b: If the unregistered user decides to sign in through Google, they choose “Google” and get redirected to the Google authentication page. After that, proceed to step 4.	
Step 4: Entered information is wrong. The system returns to step 3.	

Use case	Get Recommendation
Actors	Student
Cross references	N/A
Typical Course of Events	
Actor Intentions	System Responsibility
1. Student opens the “Recommendation” tab from the menu bar	
	2. The system shows the list of the courses that the student is enrolled in.
3. Student chooses the needed course	
	4. The system shows the list of topics and calculates performance scores on those topics.
5. Student chooses the topic they want to get report for.	
	6. The system generates the full report for that topic from collected data.
	7. The system shows the generated report
Alternative Courses	
Step 2: Student is not enrolled in any course. The system shows the empty page.	

Step 4. Student didn’t attempt any exercises. The system shows the empty page

Use case	Ask a programming question
Actors	Student, Chatbot
Cross references	N/A
Typical Course of Events	
Actor Intentions	System Responsibility
1. Student opens “Chatbot” from the menu bar	
	2. System opens window of Chatbot Cody
3. The student writes down the question	
4. Chatbot generates the answer	
5. Chatbot generates examples	
Alternative Courses	
Step 3: The student attaches a picture/document or records a voice message. Proceed to step 4.	

Use case	Ask for guidance
Actors	Student, ChatBot
Cross references	N/A
Typical Course of Events	
Actor Intentions	System Responsibility
1.The student opens the menu bar and chooses “ChatBot”	
	2. The system opens the window of Chatbot Cody
3. The student texts a message about the usage of a web application	
4. The chatbot shows detailed instructions with pictures to answer that question	
Alternative Courses	
Step 3a: The student attaches a picture/document or records a voice message. Proceed to step 4.	

Use case	Solve Exercise
Actors	Student, Online Editor
Cross references	Submit
Typical Course of Events	
Actor Intentions	System Responsibility
1. The student enters the lecture.	
	2. The system shows a list of lectures.
3. The student chooses the topic.	
	4. The system opens the window with the lecture video and a question for the exercise.
5. Student goes to “Write Code”.	
	6. The system opens the compiler.

7. The student chooses the language.	
8.The student writes the code manually.	
9. System Online Editor tracks the code	
10. The student runs the code	
	11. The system runs the compiler and generates the answer in the output window
12. The student checks the output	
Alternative Courses	
Step 5: The student goes to previous or next question without solving current question. Return to step 4.	
Step 7: If the student didn't choose a language, the system will give an error message and return to step 7.	
Step 10: The System can not compile the code. Return to step 8.	
Step 12: If the student wants and the code compiles, the student can submit the code(see Submit)	

Use case	Analyze the submission
Actors	Detection Tool
Cross references	N/A
Typical Course of Events	
Actor Intentions	System Responsibility
	1. The system provides Plagiarism Tool with previously submitted work
2. The tool checks previously submitted work for the same question	

3. The tool identifies matching lines	
4. The tool calculates a similarity score	
5. The tool generates a detailed report	
	6. The tool sends the report to the instructor
Alternative Courses	
Step 5: If the tool fails to generate the report, return to step 5.	

Use case	Create Course
Actors	Instructor
Cross references	N/A
Typical Course of Events	
Actor Intentions	System Responsibility
1. Instructor opens Catalog of courses	
	2. The system shows a list of existing courses
3. The instructor chooses to add the course	

	4. The system asks for course information
5. Instructor adds code	
6. Instructor adds name	
7. Instructor adds description	
8. Instructor submits the course	
	9. The system creates the course
Alternative Courses	
Step 9a: The Instructor has left the code of the course blank. Return to step 5.	
Step 9b: The Instructor has left the name of the course blank. Return to step 6.	
Step 9c: The Instructor has left the description of the course blank. Return to step 7.	

Use case	Submit
Actors	Student, Detection Tool
Cross references	N/A
Typical Course of Events	
Actor Intentions	System Responsibility

	1. The system loads the given code and redirects it to the detection tool
2. The detection tool loads the code	
Alternative Courses	

Use case	Enroll one student
Actors	Instructor
Cross references	N/A
Typical Course of Events	
Actor Intentions	System Responsibility

1.Instructor opens “Your classes” from the menu bar.	
	2.The system shows a list of classes that the instructor is teaching
3.The instructor chooses one of the classes	
	4.The system shows class information and gives an option to enroll students
5.The instructor chooses to enroll one student	
	6.The system asks for student information
7.Instructor fills out the necessary information and presses enroll	
	8.The system checks the data and adds that student to the class.
Alternative Courses	
Step 2: The instructor is not teaching any classes yet. The system shows a blank list.	
Step 8a: There is no such student in the system. Return to step 6.	
Step 8b: This student is already added to the class.	

Use case	Enroll multiple students
Actors	Instructor

Cross references	N/A
Typical Course of Events	
Actor Intentions	System Responsibility
1.Instructor opens “Your classes” from the menu bar.	
	2.The system shows a list of classes that the instructor is teaching
3.The instructor chooses one of the classes	
	4.The system shows class information and gives an option to enroll students
5.The instructor chooses to enroll multiple students	
	6.The system asks for students’ information
7.The instructor fills out the students’ information	
8.Instructor presses enroll	
	9.The system checks the data and enrolls them in the course
Alternative Courses	
Step 2: The instructor is not teaching any classes yet. The system shows a blank list.	
Step 9: In case at least 1 of the students’ information is wrong, the system adds students with the correct information and returns to step 6.	

Use case	View recommendation
Actors	Instructor
Cross references	N/A
Typical Course of Events	
Actor Intentions	System Responsibility
	1. The system calculates overall student performance
	2. The system sends recommendations based on student performance to the instructor
3. Instructor opens "Your reports" from the menu bar	
	4. The system shows the instructor's reports and recommendations
5. Instructor opens Recommendations	
	6. The system shows recommendations for the instructor
Alternative Courses	
Step 1: There are no students enrolled. System does not send recommendations.	
Step 6: There are no recommendations. Show a blank page	

Use case	View the progress report for the overall course
Actors	Instructor
Cross references	N/A
Typical Course of Events	
Actor Intentions	System Responsibility
1. Instructor opens “Your reports” from the menu bar	
	2. The system shows the instructor’s reports and recommendations
3. Instructor opens Reports	
	4. The system shows tabs for the overall course report and for a specific course and student submission reports
5. The instructor chooses “overall course report.”	
	6. The system generates an overall report based on student performance
	7. The system shows this report to the instructor
Alternative Courses	
Step 6a: The instructor does not have any courses. Return to step 4.	
Step 6b: The course does not have any students enrolled. Send a message and return to step 4.	

Use case	View the progress report for a course
Actors	Instructor
Cross references	N/A
Typical Course of Events	
Actor Intentions	System Responsibility
1. Instructor opens "Your reports" from the menu bar	
	2. The system shows the instructor's reports and recommendations
3. Instructor opens Reports	
	4. The system shows tabs for the overall course report and for a specific course and student submission reports
5. The instructor chooses "specific course report."	
	6. The system shows a list of courses for the instructor
7. The instructor chooses a course	
	8. The system generates a report for the specific course based on student performance
	9. The system shows this report to the instructor

Alternative Courses
Step 6: The instructor does not have any courses. Show a blank list.
Step 8: The course does not have any students enrolled. Send a message and return to step 6.

Use case	Delete course
Actors	Instructor
Cross references	N/A
Typical Course of Events	
Actor Intentions	System Responsibility
1. The instructor opens "Your courses" from the menu bar	
	2. The system shows a list of courses
3. The instructor chooses the course	
	4. The system shows details about the course, enrolled students, and options to delete and update the course
5. The instructor chooses to delete the course	
	6. The system shows a message asking for confirmation to delete the course

7. The instructor confirms the message	
	8. The system deletes enrolled students, instructor, classes, exercises, and the overall course.
Alternative Courses	
Step 2: There are no courses for the instructor. Show blank list.	
Step 7: The instructor does not confirm. Return to step 4.	

Use case	Delete exercise
Actors	Instructor
Cross references	N/A
Typical Course of Events	
Actor Intentions	System Responsibility
1. The instructor presses "Course Catalog" from the menu bar	
	2. The system shows a list of courses
3. The instructor chooses one of the courses	
	4. The system opens the course information and exercises tab

5. The instructor goes to the exercises	
	6. The system shows a list of exercises for that course
7. The instructor chooses the exercise	
	8. The system shows the contents of the exercise and an option to delete/update
9. Instructor chooses delete	
	10. The system asks for a delete confirmation
11. The instructor confirms	
	12. The system deletes the exercise
Alternative Courses	
Step 2: There are no courses for the instructor. Show blank list.	
Step 6: There are no exercises for the course. Show blank list.	
Step 11: The instructor does not confirm. Return to step 8.	

Use case	See the detection tool’s report
Actors	Instructor
Cross references	N/A
Typical Course of Events	
Actor Intentions	System Responsibility
1. The instructor goes to “Your reports.”	
	2. The system opens a list of reports and recommendations
3. The instructor goes to report	
	4. The system shows tabs for the overall course report and for a specific course and student submission reports
5. The instructor chooses student submission reports	

	6. The system opens a list of student submission reports
7. The instructor clicks on one of the reports	
	8. The system opens that exact report
Alternative Courses	
Step 2: There are no courses for the instructor. Show blank list.	

Use case	Update course
Actors	Instructor
Cross references	N/A
Typical Course of Events	
Actor Intentions	System Responsibility
1. The instructor opens "Your courses" from the menu bar	
	2. The system shows a list of courses
3. The instructor chooses the course	
	4. The system shows details about the course, enrolled students, and options to delete and update the course
5. The instructor chooses to update the course	

	6. The system opens editing mode
7. The instructor edits the course and presses “Save changes.”	
	8. The system shows a message asking for confirmation to update the course
9. The instructor confirms the message	
	10. The system updates the course
Alternative Courses	
Step 2: There are no courses for the instructor. Show blank list.	
Step 9: The instructor does not confirm. Return to step 4.	

Use case	Update exercise
Actors	Instructor
Cross references	N/A
Typical Course of Events	
Actor Intentions	System Responsibility
1. The instructor presses “Course Catalog” from the menu bar	

	2. The system shows a list of courses
3. The instructor chooses one of the courses	
	4. The system opens the course information and exercises tab
5. The instructor goes to the exercises	
	6. The system shows a list of exercises for that course
7. The instructor chooses the exercise	
	8. The system shows the contents of the exercise and an option to delete/update
9. Instructor chooses update	
	10. The system opens editing mode
11. The instructor makes changes and presses “Save changes”.	
	12. The system asks for an update confirmation
13. The instructor confirms the message	
	14. The system updates the exercise
Alternative Courses	
Step 2: There are no courses for the instructor. Show blank list.	

Step 6: There are no exercises for the course. Show blank list.
Step 13: The instructor does not confirm. Return to step 8.

2.3 Usability Requirements

Navigation and Layout: The application must feature an intuitive layout, ensuring users can navigate through the platform with minimal effort. At least 90% of users should be able to find key sections (e.g., courses, exercises, progress tracking) within 2 clicks. This will reduce confusion and improve the user experience, making it easier for users to focus on the tasks at hand instead of navigating the interface. Clear menus, visual cues, and an organized structure are essential to achieve this goal.

Performance and Loading Times: The system must deliver smooth performance and fast loading times to enhance user experience. Course content, including videos and exercises, should load within 3 seconds on average to maintain user engagement. Long loading times can lead to frustration and abandonment, so optimizing for speed is essential. Users should be able to interact with the application without noticeable delays, ensuring efficient task completion without assistance.

Easy Access and Efficient Information Retrieval: Users should be able to access any course material, exercises, or progress reports quickly. A search functionality should be provided, allowing users to find specific content in under 30 seconds. Information retrieval, such as checking exercise results or revisiting previously viewed content, should be easy and quick, minimizing any waiting time and allowing users to stay focused on learning.

User Engagement: The platform should be designed to keep users engaged with the content and provide motivation throughout their learning journey. Engagement features, such as progress tracking, interactive exercises, and regular feedback, should be prominently displayed to ensure users remain motivated and on track. Regular updates and positive reinforcement (like achievement badges or scores) will also help sustain engagement, ensuring that users continue to return and make progress.

Progress Tracking and Feedback: Effective progress tracking is essential for keeping learners motivated and informed about their development. The system should provide clear, visual progress indicators (e.g., progress bars, completed exercises) and offer timely feedback after every task or exercise. This feedback should not only indicate correct/incorrect answers but also include tips for improvement. Regular updates on progress ensure that learners can see their growth and stay motivated throughout their journey.

2.4 Performance requirements

Fast Response Time: The system must provide a quick response to user interactions such as loading exercises, submitting code, and navigating between pages. Ideally, these actions should be completed within 2 seconds under average server load. This level of responsiveness is necessary to maintain user engagement and avoid frustration. In an educational environment, especially during coding practice, any delay can disrupt the learning process and reduce the user's focus. A responsive interface encourages consistent use and contributes positively to the learning experience.

Concurrent Users Handling: The application should be capable of supporting at least 100 users simultaneously without performance degradation, such as crashes or noticeable slowdowns. These users may be compiling code, accessing learning materials, or interacting with the assistant. This requirement is essential because instructors may assign tasks to entire classrooms, and students may log in concurrently for practice or tests. A scalable and stable platform ensures that all users experience uninterrupted access, even during peak usage times.

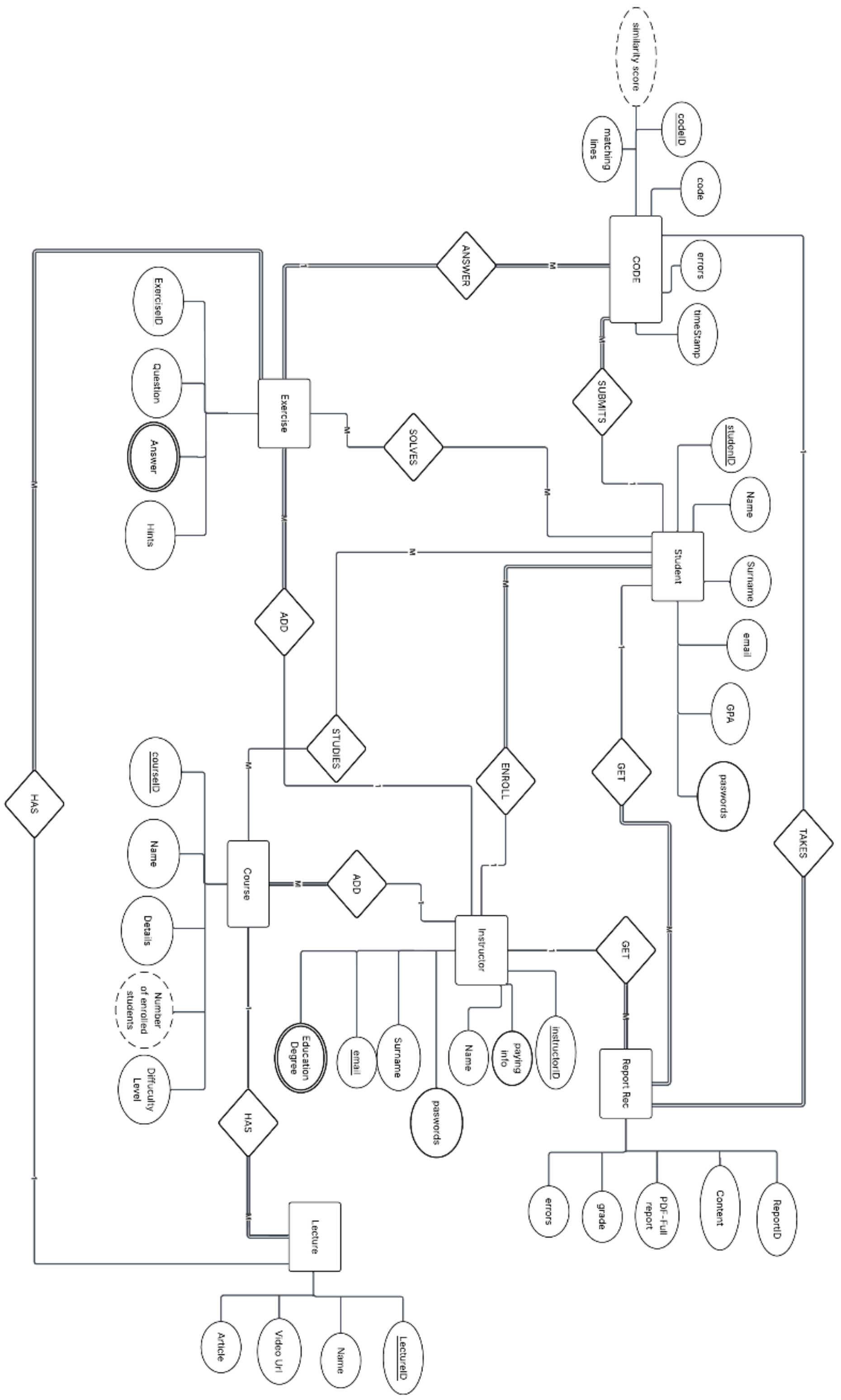
Code Compilation and Execution Time: Student-submitted code must be compiled and executed within 3 seconds for exercises of typical size (up to 100 lines of code). Fast compilation and feedback are essential in programming education, where students continuously revise their code based on results. Delayed responses discourage experimentation and hinder the learning flow. Prompt execution allows students to see immediate results from their changes, reinforcing logical understanding and debugging skills.

System Availability and Uptime: The web application must maintain a minimum uptime of 99.5% per month, excluding scheduled maintenance windows. Since students and instructors rely on the platform for daily academic tasks, downtime can interfere with lesson plans, assignment deadlines, and progress tracking. High availability is especially important for remote or asynchronous learning environments, where users expect the platform to be reliably accessible at any time.

Data Storage and Retrieval Speed: The system must allow students and instructors to retrieve data—such as solved exercises, submission history, and performance reports—within 2 seconds. Quick access to this information is vital for monitoring progress and planning next steps. Instructors often need to assess student trends to adjust their teaching strategies, while students may want to review previous attempts to reinforce learning. Delays in data retrieval could hinder this process and reduce user satisfaction.

Chatbot Response Time: The integrated chatbot, which assists users with programming queries and platform navigation, should respond to at least 90% of queries within 1 second. This real-time support is intended to mimic the availability of a tutor. If the chatbot takes too long to respond, users may lose confidence in its effectiveness or become frustrated, especially when they need quick help during practice or assessments. Fast responses ensure the chatbot remains a helpful and reliable feature.

2.5 Logical Database requirements



2.6 Software system attributes

Reliability

The system needs to be reliable so users can count on it without constant interruptions. Ideally, we’re aiming for an uptime of at least 95%(works) to avoid any significant downtime. If something unexpected happens, like a server crash, the system should have solid recovery mechanisms in place, so users can keep working with little disruption. The system should also make sure no data gets lost if something goes wrong, using transactions or backup systems to keep everything consistent. Without reliability, users would get frustrated, and the platform would lose trust.

Availability

To guarantee a defined level of availability, the system must be designed to handle high traffic and recover quickly from any failures. It should have an automated backup system running at least once per day to safeguard data and ensure that critical information is not lost. In the event of a disaster, the platform must support disaster recovery protocols that enable system recovery within 15 minutes, minimizing downtime. Furthermore, the system should include auto-scaling capabilities to accommodate sudden spikes in user demand, ensuring that performance remains consistent even during peak usage periods. Redundant servers should also be implemented to ensure the system continues to operate smoothly even if one server fails. High availability is vital to provide users with constant access to their courses and exercises, fostering trust in the platform’s reliability and minimizing disruptions to the learning experience.

Security

Security is a top priority for protecting user data and maintaining the integrity of the system. All sensitive information, such as user credentials and personal data, must be encrypted using industry-standard cryptographic techniques. The platform must also incorporate role-based access control (RBAC) to restrict access to specific data and system functionalities based on the user’s role, ensuring that only authorized users can perform certain actions. Maintaining audit logs is crucial for tracking user actions, system changes, and access attempts, with logs retained for at least one year for review. Regular security patches and updates must also be applied to address potential vulnerabilities. These security measures are necessary to protect against malicious attacks, data breaches, and unauthorized access, ensuring that the platform remains safe and trustworthy for all users.

Maintainability

We need to make sure that maintaining and updating the software doesn't become very complicated. The code should be modular, meaning each part can be updated or fixed without messing with everything else. Good documentation for both the code and API is a must, so developers can make changes without wasting time figuring out how things work. The code should be clean and follow coding standards, making it easier to debug and extend when necessary. A centralized logging system will help developers quickly spot issues and fix them. Keeping the platform easy to maintain means we can keep improving it without too much work.

Portability

The system must be designed to ensure seamless operation across a variety of operating systems and environments, including different web browsers and devices, with minimal configuration changes. The codebase should be built with flexibility in mind, avoiding strong dependencies on any specific operating system or hardware, thereby allowing for easy adaptation or porting to new environments if required. This includes ensuring compatibility across major web browsers such as Chrome, Firefox, Safari, and Edge. By focusing on portability, the system remains adaptable to evolving technological requirements and can be deployed or scaled across different infrastructures as needed, ensuring long-term flexibility and the system shouldn't be heavily tied to one specific service or technology provider. This approach is essential for accommodating future updates and expanding the platform's reach to various users and environments.

2.7 Supporting information

N/A

3 References

Codecademy. (n.d.). *Learn to code online – programming courses for beginners & experts*. Codecademy. https://www.codecademy.com

freeCodeCamp. (n.d.). *Learn to code – freeCodeCamp*. freeCodeCamp. https://www.freecodecamp.org

LeetCode. (n.d.). *Top interview questions – 150*. LeetCode. https://leetcode.com/studyplan/top-interview-150

W3Schools. (n.d.). *Learn to code – tutorials, references, and examples*. W3Schools. https://www.w3schools.com

4 Appendices

4.1.1 Acronyms and abbreviations

API - Application Programming Interface

CVV- Card Verification Value

N/A- Not Applicable

4.1.2 Google Docs link

https://docs.google.com/document/d/1I8SKHH6MQ1JYAGW1ETiTzB0BZRUSr4F1mJzcmUOh4wM/edit?usp=sharing