Bilkent University

Department of Computer Engineering

CS 492

Senior Design Project

CODED.

2023-24 Spring Semester

Final Report

Project ID: T2322

Project Team Members:

Zeynep Hanife Akgül	(22003356)
Beyza Çağlar	(22003394)
Selin Bahar Gündoğar	(22001514)
Emre Karataş	(22001641)
Zeynep Selcen Öztunç	(21902941)

Project Advisor: Halil Altay Güvenir

Instructors: Atakan Erdem, Mert Bıçakçı

Innovation Expert: Bora Güngören

13 May 2024

Table of Contents

1 Introduction..... 4

1.1 Purpose of the System..... 4

1.2 Design Goals..... 5

1.2.1 Usability..... 5

1.2.2 Performance..... 6

1.2.3 Reliability..... 6

1.2.4 Marketability..... 7

1.2.5 Security..... 9

1.2.6 Scalability..... 9

1.2.7 Maintainability..... 9

1.2.8 Flexibility, Extendibility, and Modularity..... 10

1.2.9 Aesthetics..... 11

1.3 Overview..... 11

1.3.1 Test Case Runner..... 12

1.3.2 Chatbot..... 12

1.3.3 Code Quality Check..... 13

1.3.4 Plagiarism Check..... 13

2 Requirements Details..... 14

2.1. Functional Requirements..... 14

2.1.1. Login & Sign-up Requirements..... 14

2.1.2. Course Creation Requirements..... 14

2.1.3. Lab Creation Requirements..... 14

2.1.4. Lab Submission Requirements..... 15

2.1.5. Chatbot Requirements..... 15

2.1.6. Code Analysis Requirements..... 15

2.1.7. Code Feedback Requirements..... 16

2.1.8. Test Case Runner Requirements..... 16

2.1.9. Similarity Check Requirements..... 17

2.1.10. Grading Requirements..... 17

2.2. Non-Functional Requirements..... 17

2.2.1. Usability....17
2.2.2. Reliability....17
2.2.3. Performance....18
2.2.4. Supportability....18
2.2.5. Scalability....18
2.2.6. Security....19
3 Final Architecture and Design Details....19
3.1 Overview....19
3.2 Subsystem Services....21
3.2.1 Client Subsystem....21
3.2.1.1 User Interface....21
3.2.2 Logic Subsystem....22
3.2.3 External API's Subsystem....23
3.2.4 AWS Subsystem....24
3.3 Hardware / Software Mapping....25
4 Development/Implementation Details....26
4.1 Frontend....26
4.2 Backend....27
5 Test Cases and Results....28
5.1 Functional Test Cases....28
5.2 Non-functional Test Cases....43
5.3 User Acceptance Test Cases....46
6 Maintenance Plan and Details....47
7 Other Project Elements....48
7.1 Consideration of Various Factors in Engineering Design....48
7.1.1 Data Privacy Considerations....48
7.1.2 Accessibility and Equality Considerations....49
7.1.3 Public Welfare Considerations....50
7.1.4 Global Considerations....50
7.1.5 Environmental Considerations....50
7.2 Ethics and Professional Responsibilities....51
7.2.1 Professional Responsibilities....51

7.2.2 Ethical Responsibilities....51
7.3 Teamwork Details....52
7.3.1 Contributing and Functioning Effectively on the Team....52
7.3.2 Helping Creating a Collaborative and Inclusive Environment....53
7.3.3 Taking a Lead Role and Sharing Leadership on the Team....53
7.3.4 Meeting Objectives....54
7.4 New Knowledge Acquired and Applied....54
8 Conclusion and Future Work....56
8.1 Conclusion....56
8.2 Future Work....57
9 User Manual....58
10 Glossary....59
11 References....64

1 Introduction

1.1 Purpose of the System

Bilkent University is currently facing a problem of finding lab tutors for the CS (Computer Engineering) and CTIS (Information Systems and Technologies) labs. Most students have questions about the lab assignments or have trouble debugging a specific part of their code during their lab periods; thus, these students ask their questions to tutors and teacher assistants (TAs). Unfortunately, only 2-3 TAs and 1-2 tutors can be present during each lab session. The CS courses do not have as much trouble finding TAs for lab sessions as there is a graduate program in the CS department; though, the CTIS department has to rely on successful undergraduate students to volunteer to fill the tutor vacancies for TA roles as the CTIS department does not have a graduate program. Furthermore, tutors for both CS and CTIS courses are selected among volunteering students who display proficiency in the course's coding language. Most tutors cannot stay throughout all the lab hours due to their schedules. Even when they are able to do so, the student-to-tutor ratio is around 1/30, and students have to wait around to ask their questions, and tutors are heavily overworked [1]. Henceforth, tutors either do not have the time to pay attention to each student nor have a productive lab session as they have to rush. Moreover, hundreds of non-STEM (Science, Technology, Engineering, and Mathematics) students take mandatory coding courses offered by the CS department every semester, and non-STEM students are far more likely to ask trivial questions about the lab compared to STEM students. Henceforth, there is an urgent lab tutor needed in the CS and CTIS departments, but this crisis can be solved with our application called "***CODED.***".

CODED. is a website application that assists students, tutors, TAs, and instructors in CS and CTIS labs throughout the lab process. The application answers student questions

about the lab assignments through its chatbot property. It leads students in the coding process by guiding them to write clean code and checking for any code smells, which is “any characteristic in the source code of a program that possibly indicates a deeper problem.”[2]. Following that, the completed lab assignments are uploaded to the application, where these assignments are tested for plagiarism. The instructor can compare code files they upload to the system, whether it's from students in the same section or from previous students who have taken the course. Furthermore, the code files are passed by test cases provided by the instructor and are created by AI. Instructors can grade these uploaded code files and leave comments on the uploaded code files. The students can view their grades and code feedback. Therefore, the application tremendously shortens the time it takes to grade the code and lighten the workload of tutors, TAs, and teaching assistants. Additionally, we visited the CS115, CTIS151, and CTIS152 labs to observe the lab process, interviewed the students on their needs, and received their feedback on our application. As we had first-hand experience during CS115, CTIS151, and CTIS152 labs, we have better understood the needs of tutors, TAs, and students, reevaluated the functional requirements of our application from previous versions, and further improved the application’s features.

1.2 Design Goals

CODED. focuses on specific design goals such as the following: usability, performance, reliability, marketability, extendibility, security, scalability, maintainability, flexibility, modularity, and aesthetics to make our application as ready to publish as an industry product.

1.2.1 Usability

CODED. is focused on increasing usability from both the student and the instructor's point of view as it is an interactive website. The screen placements, buttons, color scheme,

and application logic all contribute to simplifying and easing the user experience. The application also offers a dark mode for coding related pages as some students prefer to code in dark mode integrated development environments (IDE) and may prefer their screen to stay dark. Moreover, any system feature can be accessed within at most four clicks or operations. The chatbot and the code analysis tools, our application's main features, is accessible through the lab page. Overall, the application is created to be intuitive with a simple user interface to make the application as easy as possible to navigate from a user perspective.

1.2.2 Performance

Performance is a principal aspect of the application, as close to 60 people at a time are expected to use it during a single lab session. Each student is expected to ask multiple questions to the chatbot throughout a session, utilize the code analyzer feature, and upload their code to the system at least once. ***CODED.*** is able to handle the usage of multiple users without a significant performance slowdown, and the system's response time is under 0.5 seconds for any standard user action and transactions. It can reach up to 2 seconds for more complex tasks during peak usage time. After testing our application in CS115, CTIS151, and CTIS152 courses, we observed that the chatbot’s answering speed is on average 2 seconds and the system did not have any slowdowns even while up to 30 people were using the application simultaneously.

1.2.3 Reliability

The system is available to all users at all times, including lab hours and practicing hours, with minimal downtime. More specifically, the website maintains an availability of at least 99.9% days over a year, with a total downtime of no more than 9 hours annually. As we are utilizing AWS tools using credits, we expect an uptime between 99.9% to 99.999%

uptime. The AWS PostgreSQL databases prevent data loss and ensure the safe storage of personal data. The website also provides clear error messages to inform users in case of problems. Errors are logged for analysis and debugging purposes.

1.2.4 Marketability

CODED. is a unique application that is created in accordance with the needs of students and instructors in mind; thus, there is no one application that is an exact replication of our application. ***CODED.*** has distinct features that separate the application from the rest of the existing products on the market: chatbot, code analysis, code plagiarism check, and code submission. Thus, our application is highly marketable as it offers services on one platform that no other application provides. Codegrade is the most similar application to ours that is available for use in the market; though it lacks a chatbot and a submission feature which are some of the vital features of ***CODED.*** Current systems that are similar to our application are Khanmigo and CodeGrade. While none of these applications are carbon copies of ***CODED.***, they share a high similarity concerning either the idea or the features. Firstly, Khanmigo is an application that was developed by Khan Academy to help both students and teachers with learning and preparing scholarly materials. Khanmigo is similar to ***CODED.*** due to the chatbot aiming to help students learn and understand instead of giving answers directly to students. Moreover, Khanmigo also provides services to teachers: teachers can ask the chatbot to create a grading rubric or allow teachers to see their students' progress with their learning. Khanmigo also covers coding as one of its topics and does not write code for the student nor debug the code for the student. However, this application is not a coding lab application that is designed to help university-level coding courses and also does not test student codes for grading nor does it test for plagiarism [3]. Codegrade is an established application that is the most similar to our application. It is designed to be an AI tutor

application to be used during university coding labs, which both assist students and instructors and it is used by universities worldwide. The application has an integrated coding IDE that can be used to code in all streamlined coding languages. The application automatically grades student codes based on code style, code structure, and code functionality and returns written feedback to analyze the grading. The application also offers learning analytics and assignment analytics for instructors and has a code plagiarism check feature as well [4]. Nevertheless, the application does not have a chatbot feature where students can ask questions about their code or a concept. Our application covers the areas that are lacking in both of these applications. ***CODED.*** covers all areas a student would need during lab hours and alleviates the heavy workload of tutors, TAs, and instructors. Moreover, ***CODED.*** also covers code quality aspects to help students write clean code and help them become better developers. Furthermore, ***CODED.*** is marketed towards universities and possibly coding boot camps for high school coding classes. We have also participated in the start-up competition organized by YES (Bilkent University Young Entrepreneurs Society) and presented our project. Even though we did not win any awards, it was a valuable experience to be surrounded by start-up CEOs and investors. We were able to see what was expected of successful marketable project ideas and how we could further improve our project with respect to its marketability. Additionally, we visited the CS 115, CTIS 151, and CTIS 152 lab sessions to test our applications and conducted surveys to obtain user feedback. We applied to the Ethics Committee to conduct surveys at these labs and obtained our permit. We improved our application until the final demo to perfectly cater to the needs of students and instructors in these courses based on the feedback we received from these surveys. Thus, our application is ready to be used as a commercialized product and has achieved its marketability level.

1.2.5 Security

Student names, Bilkent school IDs, and email addresses are all considered to be personal data by the GDPR and are kept in our databases [5]. To ensure the security of this data, we employed encrypted databases to utilize industry-standard encryption algorithms to safeguard sensitive information. We also consider student code and grades as sensitive information; thus, the system implements several actions to strengthen security. User access to the system is strictly controlled, limiting access to authorized personnel only: enrolled students, instructors, teaching assistants, and tutors. Students are not able to enroll themselves in the application but the application sends an invitation through emails to students. The list of students that are to be enrolled in the application is provided by the course instructor in a csv file.

1.2.6 Scalability

As mentioned in section 1.2.2, 30 users were able to use the application simultaneously. Throughout one lab session, we expected the maximum number of students to be 60; however, not all users were willing to use the application. Though, with 30 users using the application at once, the performance and responsiveness of the system remained consistent. The importance of scalability will increase in the future as the system is developed regarding the possibility of usage from different schools or even companies, growing the volume content of the application. Henceforth, we will do further checks to make sure that scalability will stay as a top-notch priority.

1.2.7 Maintainability

The application has been and will be maintained throughout its lifecycle due to its heavy user interaction nature. We completed the application at the beginning of May and

tested it in CS 115,CTIS 151, and CTIS 152 labs. While testing our beta version, we found some bugs and obtained user feedback by conducting a survey. With the feedback we received from students, we improved chatbot and code analysis features further and added new features while increasing user satisfaction with the application for the next labs we tested. If Bilkent University decides to use our application in the next semester for its coding labs, the application will be maintained for the foreseeable future. Lastly, as version control is kept in Github in our private repository, we opened issues and tickets on Github. Furthermore, we heavily used unit testing, integration testing, system testing, and acceptance testing to ensure the highest quality before every application release.

1.2.8 Flexibility, Extendibility, and Modularity

CODED. offers users several types of flexibility, extendibility, and modularity by enabling several options to custom-make features. For instance, instructors have the option to make the application create the test cases for coding questions in the labs. Moreover, instructors can choose to allow the chatbot to reveal answers after the student fails to understand the topic, even after asking several questions about it. The students also have the flexibility of checking their code for code analysis before submitting it, which may give them a hint as to why their codes may have bugs or do not comply with the coding conventions of that particular coding language. This adds modularity to the user experience.

From the developer's perspective, we have coded our project in model-view-controller (MVC), which allows new features to be easily added to the application. This not only adds flexibility and extensibility but also modularity to the project.

1.2.9 Aesthetics

The user interface of the application is purposefully chosen to be visually appealing to the users. We chose to add both a light and dark mode to allow the students to choose their preferences. Furthermore, we chose white as the background color to give it a contrast with the components on the screen. We specifically chose vibrant colors to attract the attention of students and keep it entertaining. Moreover, the contrasting colors add enhanced navigation and usability intuition [6]. We believe that it also resembles the vibrant colors often preferred by developers on IDE's color schemes, such as Monokai, One Dark, Tomorrow Theme, and Dracula. It is important to note that we also preferred a minimalistic approach to the application to give a clean feeling. Moreover, we added certain UI features such as moving bubbles and loading signs to let the student know the popups and states on the screen.

1.3 Overview

CODED. is a comprehensive web application designed to address the challenges faced by Bilkent University in providing sufficient lab tutoring for CS and CTIS courses. The platform aims to improve the lab experience by facilitating efficient student-tutor interactions and streamlining the code evaluation process. Key features include a chatbot for immediate answers to student questions, automated code quality checks, and plagiarism detection. The application not only helps students refine their coding skills but also significantly reduces the workload for tutors and TAs by automating critical aspects of the lab process, such as testing for code correctness and calculating grades and partial points. Moreover, the integrated chatbot significantly enhances the learning process by providing instant, personalized support for students' questions, making it easier to deal with coding challenges. Using advanced technologies, ***CODED.*** establishes a new level of effectiveness for educational tools, offering a more efficient and improved learning experience for Bilkent University's CS and CTIS departments.

1.3.1 Test Case Runner

The Test Case Runner component is essential for evaluating student-submitted code. It allows for insights for the grading of lab assignments using custom test cases provided by instructors or teaching assistants. The tool takes each code submission made by students, runs test cases on these codes, and tests each method individually for accuracy. Instructors receive detailed insights into the grading process, while students can access their grades. The system can also generate these test cases on demand, ensuring a comprehensive assessment of each student's work. The test case runner functions are generated by the AI, which allows for creating a test case function for any uploaded lab assignment.

1.3.2 Chatbot

The chatbot is integral for student interaction, providing a platform where students can ask questions and get guidance on coding problems. The chatbot is carefully designed to facilitate learning and problem-solving, guiding students without giving away direct answers, thus upholding academic integrity. By nudging students towards resources or hinting at methodologies rather than providing outright solutions, the chatbot helps reinforce the learning objectives and strengthens problem-solving skills. This approach not only aids in the retention of knowledge but also instills a sense of achievement and confidence in students as they navigate through coding challenges.

Moreover, the chatbot is equipped with natural language processing capabilities, enabling it to understand and respond to a wide range of queries with precision. Whether it's a simple syntax question or a complex algorithmic problem, the chatbot can guide students to the appropriate section of documentation, suggest relevant examples, or offer conceptual explanations.

The chatbot also incorporates feedback mechanisms, allowing it to learn from interactions and continuously improve its responses. Students can rate the helpfulness of answers, submit queries that the chatbot couldn't address, and even contribute to its knowledge base. This collaborative aspect not only enhances the chatbot's effectiveness but also fosters a community-driven approach to learning, where knowledge sharing becomes an integral part of the educational process.

1.3.3 Code Quality Check

The Code Quality Checker component analyzes submissions for potential issues and overall quality and its main focus is focused on improving the quality of student code. It accepts single code file input. This component goes beyond only highlighting the errors and problems, it also gives feedback for working pieces of code in terms of improvements. By highlighting areas for improvement, the component aids students in adopting coding best practices and refining their code before final submission. It analyzes the code statically and assesses bugs, code smells, security issues, etc.

1.3.4 Plagiarism Check

The Plagiarism Checker component verifies the originality of student submissions by comparing each submission against others. It prompts the user to specify the language of the code when a folder of code files is uploaded and generates a similarity rate among uploaded codes. Based on this similarity rate, the code is suggested to be plagiarized. This plays a key role in upholding academic standards, as it helps identify similarities that may suggest plagiarism, thereby ensuring the uniqueness and integrity of student work.

2 Requirements Details

2.1. Functional Requirements

2.1.1. Login & Sign-up Requirements

The application should

- Allow instructors to sign up via the sign-up form on the website.
- Allow students to register via an email sent with their user information.
- Allow all users to log in once they are registered to the system.

2.1.2. Course Creation Requirements

The application should

- Allow instructors to create a new course.
- Ask for the course name, the course code, the number of sections, a CSV file containing the students that will take the course, the instructors that will teach the course, and TAs that will assist in the course.
- Send invitation emails to the students so that they can join the CODED. website and the created course.

2.1.3. Lab Creation Requirements

The application should

- Allow instructors to create a new lab.
- Ask for the lab number, the lab objective, a PDF file with lab instructions, the lab deadline, and the permissions for the test case results, code analysis, and late submission.

- Parse the lab instructions and divide the lab into parts based on the questions in the lab.

2.1.4. Lab Submission Requirements

The application should

- Allow students to upload a .zip file with their code.
- Extract the contents of a student's submission and separate the files to their respective lab parts.

2.1.5. Chatbot Requirements

The application should

- Allow students to reach the CODED. chatbot.
- Answer the student's questions with helpful hints to the solution rather than the correct answer itself.
- Answer students' questions in a readable manner, by cooperating with markdown return format.
- Prevents students to ask questions that are not related to the computer science domain.
- Reply to the answers in a stream network.
- Answer with pseudo-like code unless the student is struggling too much with the same issue.

2.1.6. Code Analysis Requirements

The application should

- Provide the instructors with the choice of revealing the code analysis results to the students while creating the lab.
- Get student code analysis file upload

- Process student submissions via SonarCloud API.

2.1.7. Code Feedback Requirements

The application should

- Allow instructors to view the students' submitted code, leave comments on the code file.
- Allow students to review the instructor's feedback once the assignment is graded.

2.1.8. Test Case Runner Requirements

The application should

- Allow instructors to create new test cases for a lab part by providing a non-mandatory input-output pair and a mandatory answer key.
- Create test case(s) based on the sample runs provided in the parsed lab instructions file, if there are any, once an answer key is provided.
- Provide the instructors with the choice of revealing the test case results to the students while creating the lab.
- Generate test files via OpenAI API using technologies based on the lab's programming language.
- Use pytest for Python labs and UnityTest for C labs.
- Upload the generated test file to AWS (Amazon Web Services) S3 (Amazon Web Services Simple Storage Service), as well as fetch it when needed.
- Automatically run the test cases on the student lab submissions and display the results.

2.1.9. Similarity Check Requirements

The application should

- Allow instructors and TAs to upload selected source code files to the MOSS similarity check tool via the CODED. website.
- Allow instructors to specify the language of the uploaded code files.
- Create a similarity rate among the codes using the MOSS API and redirect the user to the similarity report generated by MOSS.

2.1.10. Grading Requirements

The application should

- Allow instructors and TAs to grade the student submissions.

2.2. Non-Functional Requirements

2.2.1. Usability

The website has a consistent user interface to ensure clear navigation and to make it effortless for users to have a plain understanding of the fundamentals of the program. Furthermore, any system feature can be accessed within at most three clicks or operations. This focus on user-centric design aims to make the website user-friendly and engaging.

2.2.2. Reliability

The system is always available to all users, including lab and practicing hours, with minimal downtime. More specifically, the website should maintain an availability of at least 99.8% over a year, with a total downtime of no more than 9 hours annually. High-security databases, AWS PostgreSQL and AWS S3, are used to prevent data loss and ensure the safe storage of personal data. The website provides clear error messages to inform users in cases of problems. Errors are logged for analysis and debugging purposes.

2.2.3. Performance

The system is responsive even during most busy periods like lab hours. It is able to handle the usage of multiple users without a significant performance slowdown and the response time of the system is under 0.5 seconds for any standard user action and transactions. The response time for more complex actions is approximately 2 seconds, even when the website experiences a peak usage time. Because a standard lab has at most 100 students, the system maintains its performance with at least 200 users using the application simultaneously. Several APIs have been used during the development process, so the system ensures that the third-party API calls have a latency of less than 5 seconds.

2.2.4. Supportability

The website is a modular system to ensure an easy integration of new features. This design allows changes to be made without disrupting the entire system. The integration of at least 10 new features per year is supported without causing any significant corruption in the existing system. The system is maintained as long as it is being used, and the maximum response time for user support inquiries related to new and existing features are 24 hours. Furthermore, the website supports at least 3 different web browsers and any device that is able to use web browsers to surf through websites.

2.2.5. Scalability

We target a large audience using the system simultaneously since students participate in the lab during the same hours. Thus, the significance of the scalability increases so that the application easily adapts and is able to expand in response to growing demands. As the user volume increases, the performance and responsiveness of the system remains consistent. For this purpose, we set a threshold that the website will accommodate at least 50% growing demand in terms of the number of users and content volume. The importance of scalability

will increase in the future as the system is developed to allow for usage from different schools or even companies, increasing the volume of content of the application.

2.2.6. Security

The system does not store much personal information but only the names, Bilkent school IDs, and email addresses of its users. While it is crucial to store this information safely, the system is also secure to store the work students upload and their respective grades. To ensure the security of this data, we employ encrypted databases, utilizing industry-standard encryption algorithms to safeguard sensitive information. The system also implements several actions to strengthen security. User access to the system is strictly controlled, limiting access to authorized personnel only: enrolled students, instructors, teaching assistants, and tutors. Furthermore, in the cases of security breaches, the system does have an incident response time of no more than 30 minutes.

3 Final Architecture and Design Details

3.1 Overview

CODED. is divided into systems so that the development is smoother. Each system is crucial in the project's overall success and detailed enough to be implemented separately from other components. Below shows the subsystem decomposition of CODED. along with the relations among different systems.

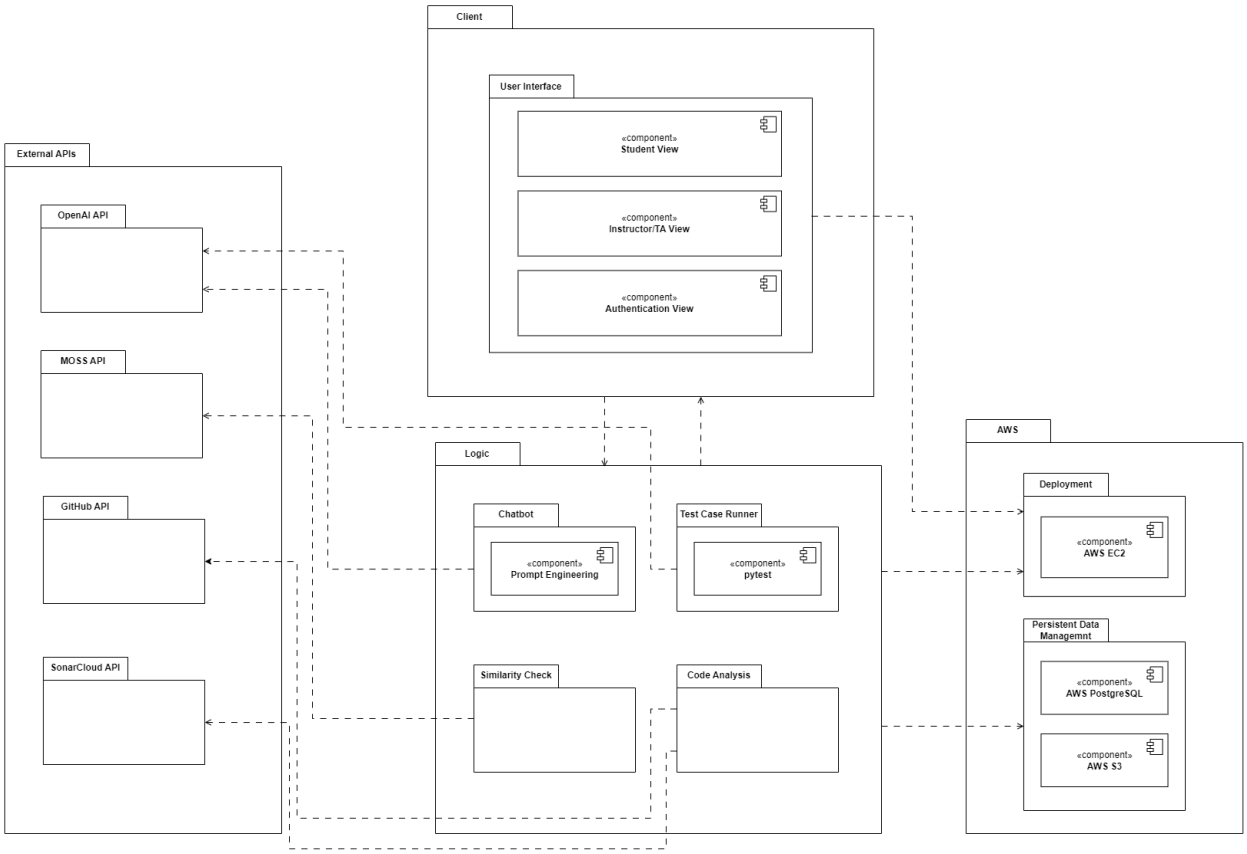

*Fig. 1: Subsystem Decomposition of **CODED.***

CODED. is consisted of the client layer, logic layer, external APIs layer, and AWS layer. The client layer includes user interfaces, which show differences based on who logs into the system. The user interface in the client layer interacts with the logic layer. The logic layer is where the functional requirements are being handled that includes components for the chatbot, test case runner, similarity check, and code analysis. Chatbot and Test Case Runner compenents interact with the OpenAI API and similarity check component interacts with the MOSS API, which are included in the external API's layer. The logic layer also interacts with the AWS layer, which includes data management and deployment components. Data gathered by the logic layer are stored in this layer, and data are retrieved or stored when needed through interaction with the AWS layer.

3.2 Subsystem Services

3.2.1 Client Subsystem

3.2.1.1 User Interface

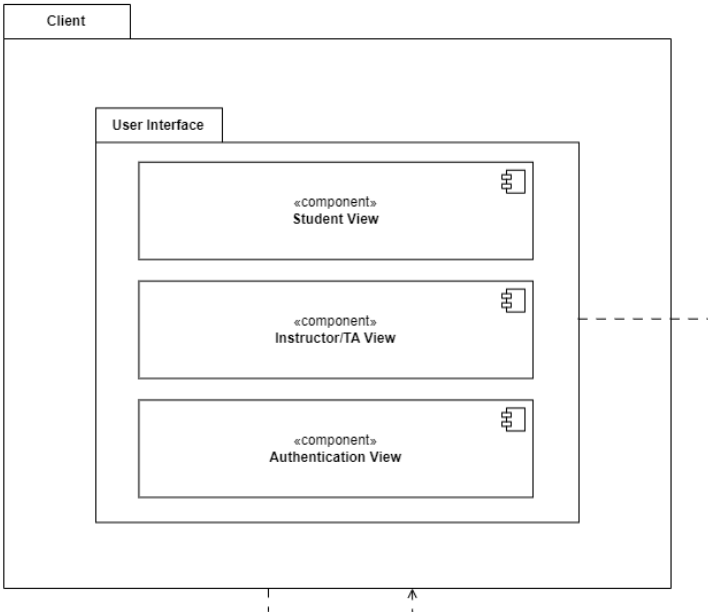

*Fig. 2: User Interface sub-module of **CODED.***

The user interface component consists of several views that are specific to the user types. These views are as follows.

Student View: Students are one type of user and when logged into the system, they interact with many distinct screens that allow them to experience the student functionalities of ***CODED.***

Instructor/TA View: Instructors and TA's are types of user and when logged into the system, they interact with many distinct screens that allow them to experience the instructor/TA functionalities of ***CODED.***

Authentication View: Before logging in, any user without any type can interact with screens that do not require to be logged in. A user can log in to the system through these screens.

3.2.2 Logic Subsystem

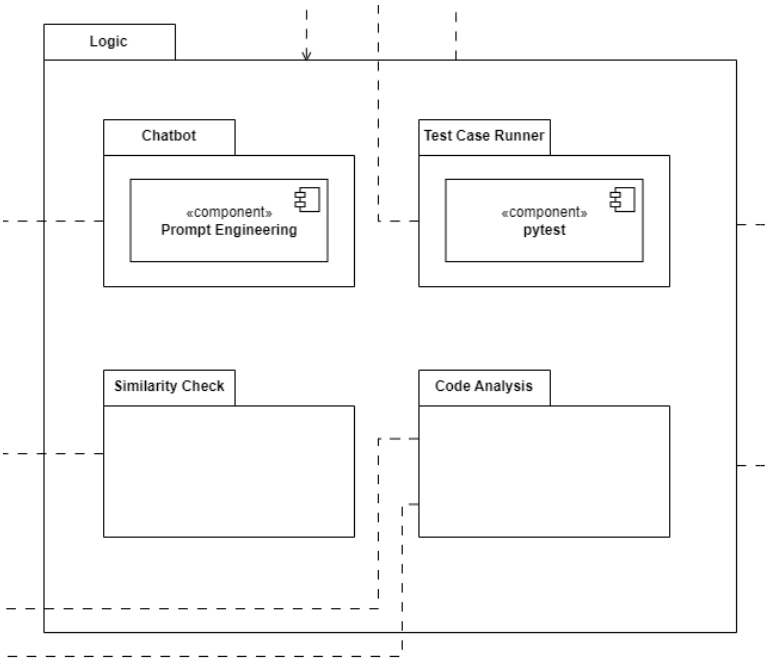

*Fig. 3: Logic sub-module of **CODED.***

The logic subsystem is where all functionalities of ***CODED.*** are handled. This layer includes a component for chatbot, test case runner, similarity check, and code analysis which stands for handling the main features of ***CODED.*** Chatbot, test case runner, code analysis, and similarity checker components interact with the External API's layer, retrieving and sending data from/to them.

3.2.3 External API's Subsystem

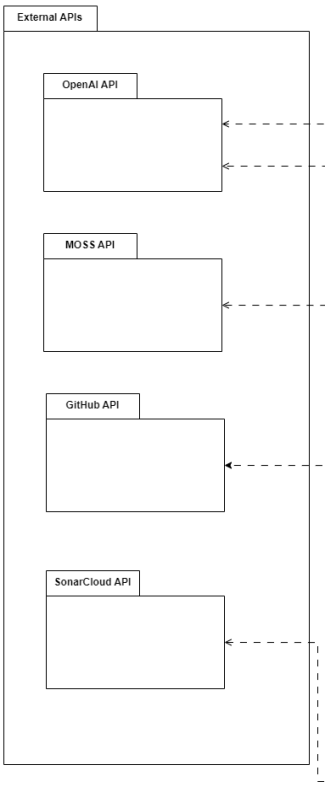

*Fig. 4: External API's subsystem of **CODED.***

Several classes in the Logic layer interact with the External APIs to be able to handle their tasks. The two API classes are as follows.

OpenAI API: Chabot class interacts with this API to provide a chatbot to the system. Also, the Testcase Runner class generates test cases and functions by interacting with the OpenAI API.

MOSS API: The similarity check component interacts with the MOSS API inside the external API's layer in order to do similarity measurements between pieces of code.

GitHub API: The code analysis component interacts with the GitHub API inside the external API's layer, uploading student codes to public repositories and deleting them to be able to use the SonarCloud API.

SonarCloud API: The code analysis component interacts with the SonarCloud API inside the external API's layer to receive static code analysis of pieces of code of students on GitHub public repository.

3.2.4 AWS Subsystem

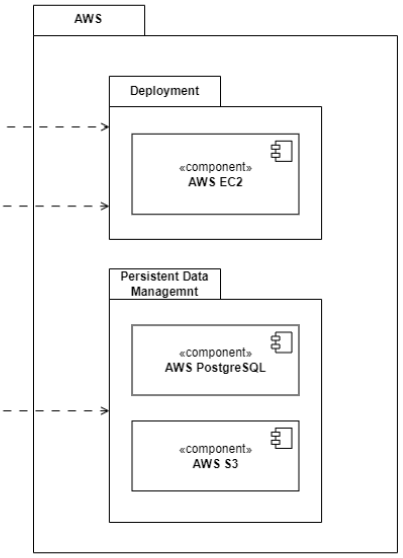

*Fig. 5: AWS RDS sub-module of **CODED.***

The system is deployed on an EC2 instance of AWS, within the Deployment component. Application also uses some other services of AWS. All data is stored in PostgreSQL within the AWS Relational Database Systems. Data is also retrieved from this database system and the AWS layer is reached by the Logic layer where necessary data processing is performed. All data that should be stored in a file format are stored inside an S3 bucket, which is one other component under Persistent Data Management.

3.3 Hardware / Software Mapping

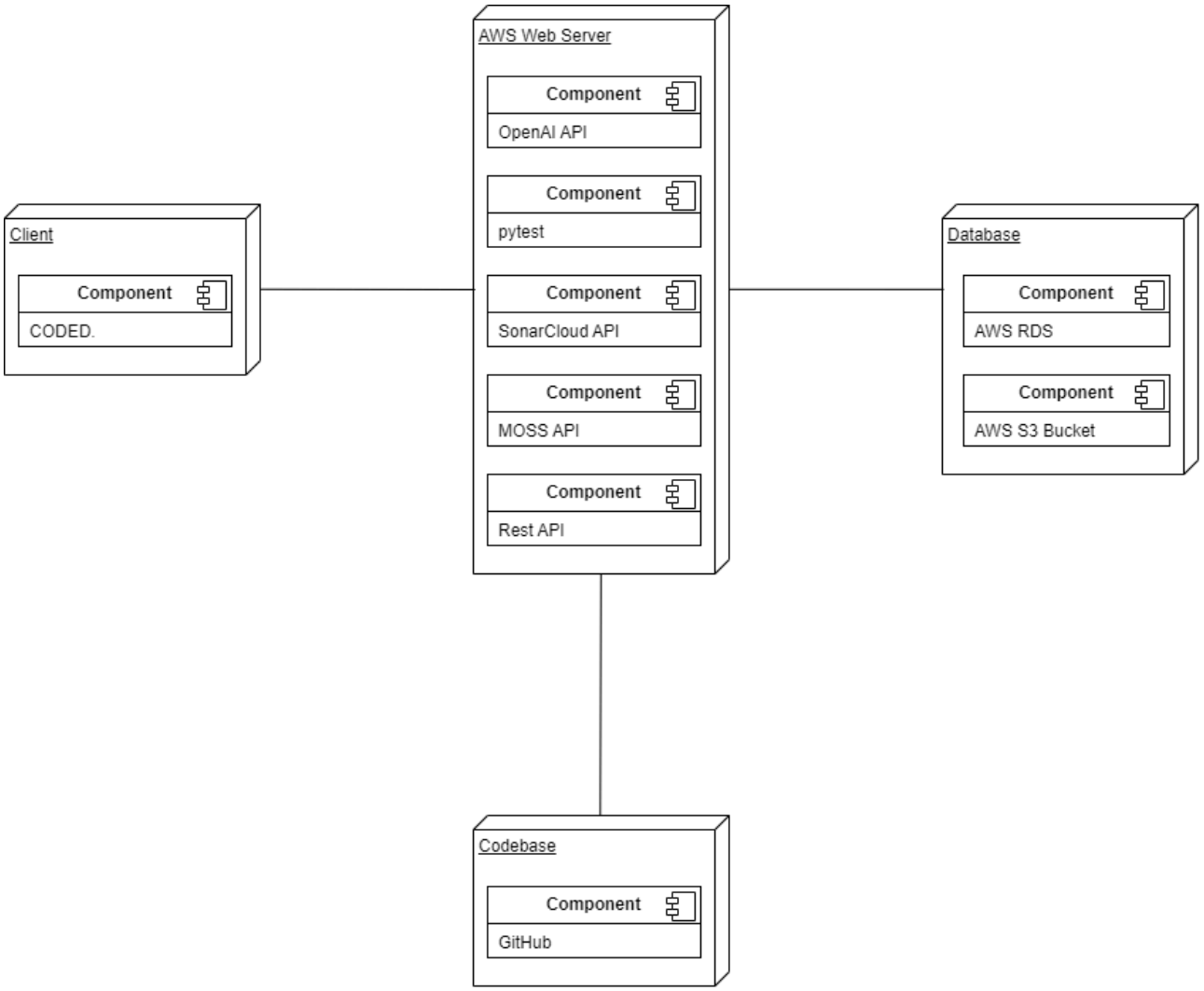

*Fig. 6: Deployment Diagram of **CODED.***

CODED. is accessible to all users via the web. The primary device to use the website would be a computer; however, the users can also access it via any mobile device. The user can send requests to the web server via the **Rest API**, which has been used in the application's back-end. The website is be deployed on an AWS cloud server using **AWS EC2 (Amazon Web Services Elastic Cloud Compute)**. Therefore, all the logic layers, including Rest API, is be contained in the cloud. Since a lot of users are expected to use the application at the same time, the server should be able to handle heavy request traffic. AWS is used for

its elastic scalability features. Moreover, **AWS RDS (Amazon Web Services Relational Database Service)**, an Amazon-based relational database service, and **AWS S3 (Amazon Web Services Simple Storage Service)**, an Amazon-based storage service, is used to store any required data. Thus, all user and application data is stored in the AWS cloud with necessary security precautions. The source code of the application is kept in **GitHub** which is be fed to the server while running the website.

4 Development/Implementation Details

4.1 Frontend

For our frontend implementation, we initially read through our requirements. We discussed the functionalities of our project to decide on a design. Then, we started with essential mockups for each member to understand how the project would look. We used React.js as our frontend framework because of its robust ecosystem and component-based architecture. Next, we decided on our color palette for the project to maintain a consistent visual identity and enhance user experience. We also decided what will be on the navigation bar and the footer since they are standard on all pages. After deciding on these, we started implementing the frontend. Each group member contributed to the front end. Once we implemented the pages, we decided on the pages' navigation and which user types would access which pages. While deciding on anything related to the frontend, we prioritized the user experience and intuitiveness. In addition to this, we also added documentation for our application so all users can understand and visualize how to use it. Overall, our frontend implementation process was handled smoothly and professionally.

4.2 Backend

To start implementing our backend, we first decided on our class diagram. We thought about our classes, their variables, and their relationships with other classes. Based on those relationships, we started implementing our model classes. We used Java's Spring Boot framework in the backend. We preferred this framework because of its extensive support for building scalable web applications. We also used Hibernate, an object-relational mapping tool for Java [7]. Using Hibernate, our code was automatically converted to database queries and tables. We chose PostgreSQL as our database because of its flexibility, robust concurrency support, scalability, and cross-platform compatibility, which made it an ideal choice for our project's database needs.

We continued implementing our repository, service, and controller classes for each model class we created. After that, we started using Spring's Security framework, an authentication and access-control framework [8]. By implementing that, we could use tokens to authenticate users and carry important information through the tokens, such as user IDs and user types. We utilized the framework to encrypt sensitive information, such as passwords, before storing it in the database.

While implementing the model, controller, service, and repository classes, we concurrently worked on each component of our application, including the Chatbot, Test Case Runner, Code Analysis, and Plagiarism Checker components. The Chatbot Component uses Open AI's API to generate responses to user questions. In the Test Case Runner Component, pytest runs test cases on Python codes, and Open AI's API generates test cases for each lab. For the Plagiarism Checker component, we are utilizing the MOSS software's API. Finally, for our Code Analysis component, we are utilizing SonarCloud which is a cloud-based static code-analysis tool [9].

We also utilized AWS's services in our backend implementation. We used RDS to store all the data that is stored in our PostgreSQL database. We are using AWS's S3 service to store and retrieve files. We also used AWS's EC2 in order to deploy our application.

5 Test Cases and Results

5.1 Functional Test Cases

Test ID: CD001	Test Type/Category: Functional Unit Testing
Summary/Title/Objection: Account Creation	
Step	Expected Output
1. Open the *CODED.* web page	Landing page is opened
2. Click on the Signup button.	The page is directed to the signup page
3. Fill in corresponding text fields	Text fields are filled
4. Click on the Signup button	User is saved to the database with the entered information.
Priority/Severity: High	
Date-Result: 05/05/2024 - Pass	

Test ID: CD002	Test Type/Category: Functional Unit Testing
Summary/Title/Objection: Logging In	
Step	Expected Output
1. Open the *CODED.* web page	The Landing page is opened.
2. Click on the Login button	The page is directed to the login page.

3. Fill in the required text fields	Text fields are filled.
4. Click on the login button	User is logged in to reach the dashboard page.
Priority/Severity: Critical	
Date-Result: 05/05/2024 - Pass	

Test ID: CD003	Test Type/Category: Functional Unit Testing
Summary/Title/Objection: Logging Out	
Step	Expected Output
1. Click on the logout button at the navbar	User is logged out to reach the landing page.
Priority/Severity: High	
Date-Result: 05/05/2024 - Pass	

Test ID: CD004	Test Type/Category: Functional Unit Testing
Summary/Title/Objection: Messaging With the Chatbot	
Step	Expected Output
1. Log in as a student	Student dashboard is opened
2. Click on the course name	Course page is opened
3. Click on the lab name	Lab page is opened
4. Type anything	Chatbot responded meaningfully.
Priority/Severity: Critical	
Date-Result: 05/05/2024 - Pass	

Test ID: CD005	Test Type/Category: Functional Unit Testing
Summary/Title/Objection: Asking Code to the Chatbot	
Step	Expected Output
1. Log in as a student	Student dashboard is opened
2. Click on the course name	Course page is opened
3. Click on the lab name	Lab page is opened
4. Paste a piece of code and ask about the problem with it	Chatbot finds the problem but does not reveal how to correct it, only gives hints on how to fix it the problem
Priority/Severity: Critical	
Date-Result: 07/05/2024 - Pass	

Test ID: CD006	Test Type/Category: Functional Unit Testing
Summary/Title/Objection: Asking a Code Question to the Chatbot	
Step	Expected Output
1. Log in as a student	Student dashboard is opened
2. Click on the course name	Course page is opened
3. Click on the lab name	Lab page is opened
4. Ask a question to generate a code	Chatbot does not give complete code that answers the question but gives hints and directions on what approach students can solve the problem.
Priority/Severity: Critical	

Date-Result: 07/05/2024 - Pass

Test ID: CD007	Test Type/Category: Functional Unit Testing
Summary/Title/Objection: Seeing the Test Case Results	
Step	Expected Output
1. Log in as a student and navigate to the upload code to the analysis page.	Upload code analysis page is opened.
2. Upload the code as a submission.	The results of how many test cases were passed and how many were failed are shown if the instructor enabled seeing the test case results.
Priority/Severity: High	
Date-Result: 07/05/2024 - Pass	

Test ID: CD008	Test Type/Category: Functional Unit Testing
Summary/Title/Objection: Submitting the Lab Work as a Student	
Step	Expected Output
1. Click on the course name on the dashboard.	Course page is opened.
2. Click on the lab name on the course page.	Lab page is opened.
3. Click on the submit button.	File explorer of the computer is opened.
4. Click on the file to be uploaded and click on the upload button.	File chosen is submitted and seen on the submitted files table with the submission date.

Priority/Severity: High
Date-Result: 05/05/2024 - Pass

Test ID: CD009	Test Type/Category: Functional Unit Testing
Summary/Title/Objection: Editing the Submitted Lab Work as a Student	
Step	Expected Output
1. Click on the course name on the dashboard.	Course page is opened.
2. Click on the lab name on the course page.	Lab page is opened.
3. Click on the ‘Submit new Submission’ button.	File explorer of the computer is opened.
4. Click on the ‘Save’ button.	File chosen is submitted and seen on the submitted files table with the new submission date.
Priority/Severity: High	
Date-Result: 07/05/2024 - Pass	

Test ID: CD010	Test Type/Category: Functional Unit Testing
Summary/Title/Objection: Deleting the Submitted Lab Work as a Student	
Step	Expected Output
1. Click on the course name on the dashboard.	Course page is opened.
2. Click on the lab name on the course	Lab page is opened.

page.	
3. Click on the ‘Delete Submission’ button.	A pop-up warning is displayed.
4. Click on the ‘I’m Sure’ button on the pop-up.	Lab submission is deleted.
Priority/Severity: High	
Date-Result: 07/05/2024 - Pass	

Test ID: CD011	Test Type/Category: Functional Unit Testing
Summary/Title/Objection: Viewing the Past Submissions and Grades	
Step	Expected Output
1. Click on the course name on the dashboard.	Course page is opened.
2. Click on the past submissions button.	List of past submissions is shown with details.
Priority/Severity: Medium	
Date-Result: 05/05/2024 - Pass	

Test ID: CD012	Test Type/Category: Functional Unit Testing
Summary/Title/Objection: Upload New Lab Assignment for Students	
Step	Expected Output
1. Log in as Instructor or TA/Instructor	TA/Instructor dashboard is opened
2. Click on the course name on the dashboard	Course page for TA/Instructor is opened

3. Click on create new lab button.	Lab creation page is opened
4. Fill in necessary fields	All fields are filled.
5. Click on the create lab button.	Lab is created and published for students to see.
Priority/Severity: High	
Date-Result: 05/05/2024 - Pass	

Test ID: CD013	Test Type/Category: Functional Unit Testing
Summary/Title/Objection: Creating New Course	
Step	Expected Output
1. Log in as Instructor	Instructor dashboard is opened
2. Click on create a course button on the dashboard	Course creation page is opened
3. Fill in necessary fields	All fields are filled
4. Click on the create the course button	The course is created and saved into a database.
Priority/Severity: High	
Date-Result: 05/05/2024 - Pass	

Test ID: CD014	Test Type/Category: Functional Unit Testing
Summary/Title/Objection: Assigning Additional Test Cases	
Step	Expected Output
1. Log in as Instructor	Instructor dashboard is opened.

2. Click on the course name on the dashboard	Course page for instructor is opened.
3. Click on the lab name on the course page	Lab page is opened
4. Click on add new test case button	Test case creation page is opened
5. Fill in necessary fields	All fields are filled
6. Click on add the test case button	Test case is saved into the database and is seen on the lis t of test cases
Priority/Severity: Medium	
Date-Result: 07/05/2024 - Pass	

Test ID: CD015	Test Type/Category: Functional Unit Testing
Summary/Title/Objection: Using the Code Analyzer	
Step	Expected Output
1. Log in as a student	Student dashboard is opened
2. Navigate to the code analysis upload page	Code analysis upload page is opened
3. Upload code file	Code is successfully uploaded to be checked
4. Website redirects to the analysis results page	Code Analysis results are displayed
Priority/Severity: High	
Date-Result: 07/05/2024 - Pass	

Test ID: CD016	Test Type/Category: Functional Unit Testing
Summary/Title/Objection: Grading of Student's Work	
Step	Expected Output
1. Log in as a student	Student dashboard is opened
2. Click on the course name on the dashboard	Course page for the student is opened
3. Click on the lab name on the course page	Lab page is opened
4. Click on the submit lab button	File explorer of the student's computer is opened
5. Choose a file from the file explorer and click on the submit button	Submission file is saved to the database and is seen in the submitted files table
Priority/Severity: Critical	
Date-Result: 07/05/2024 - Pass	

Test ID: CD017	Test Type/Category: Functional Unit Testing
Summary/Title/Objection: Checking Similarity Among Codes	
Step	Expected Output
1. Log in as a student	Student dashboard is opened
2. Click on the course name on the dashboard	Course page for the student is opened
3. Click on the lab name on the course page	Lab page is opened
4. Click on the submit lab button	File explorer of the student's computer is

	opened
5. Choose a file from the file explorer and click on the submit button	Submission file is saved to the database and is seen in the submitted files table. Similarity check is done on the back and the result is displayed for the instructor.
Priority/Severity: High	
Date-Result: 07/05/2024 - Pass	

Test ID: CD018	Test Type/Category: Functional Unit Testing
Summary/Title/Objection: Forgotten Password	
Step	Expected Output
1. Click on the forgotten password on the navigation bar	Forgotten password page is opened
2. Enter the email address to the corresponding field	Field is filled
3. Click on the send mail button	A mail is sent to the entered email address and a page is directed to the enter the code page.
4. Enter the code written on the mail to the corresponding field	If the code is correct, page is directed to the change password page on the profile page
Priority/Severity: Medium	
Date-Result:07/05/2024 - Pass	

Test ID: CD019	Test Type/Category: Functional Unit Testing
Summary/Title/Objection: Downloading the Submitted Lab Assignment	

Step	Expected Output
1. Log in as a student	Student dashboard is opened
2. Click on the course name on the dashboard	Course page for the student is opened
3. Click on the lab name on the course page	Lab page is opened
4. Click on the submitted lab assignment	Submitted lab is automatically downloaded
Priority/Severity: Medium	
Date-Result: 05/05/2024 - Pass	

Test ID: CD020	Test Type/Category: Functional Unit Testing
Summary/Title/Objection: Complaining to Lab Grade	
Step	Expected Output
1. Log in as a student	Student dashboard is opened
2. Click on the course name on the dashboard	Course page for the student is opened
3. Click on the lab name on the course page	Lab page is opened
4. Click on the complaint to your grade button.	Pop-up window is opened
5. Write the reason for your complaint and click the submit button	The complaint is sent to the grader of the lab.
Priority/Severity: Medium	

Date-Result: Not implemented

Test ID: CD021	Test Type/Category: Functional Unit Testing
Summary/Title/Objection: Change Password	
Step	Expected Output
1. Log in and navigate to the user profile section.	Profile page is opened
2. Click on the change password button	Text field is opened to enter a new password.
3. Enter your new password	The new password is saved to the database and the password is changed
Priority/Severity: High	
Date-Result: 07/05/2024 - Pass	

Test ID: CD022	Test Type/Category: Functional Unit Testing
Summary/Title/Objection: Change Profile Picture	
Step	Expected Output
1. Log in and navigate to the user profile section.	Profile page is opened
2. Click on the change profile picture button	File explorer is opened
3. Choose the new profile picture	The profile picture is changed and saved to the database
Priority/Severity: Medium	

Date-Result: 07/05/2024 - Pass

Test ID: CD023	Test Type/Category: Functional Unit Testing
Summary/Title/Objection: Viewing Students Enrolled in the Course	
Step	Expected Output
1. Log in as instructor, tutor or TA and navigate to course page	Course page is opened
2. Click to see the students who are enrolled in the course	The students who are enrolled in the course are displayed
Priority/Severity: Medium	
Date-Result: Not implemented	

Test ID: CD024	Test Type/Category: Functional Unit Testing
Summary/Title/Objection: See the Past Labs Uploaded by the Students	
Step	Expected Output
1. Log in as instructor and navigate to the course page	Course page is opened
2. Click on the lab button	Lab page is opened, displaying each lab assignment uploaded by each enrolled student with the code analysis, test case and plagiarism check results
Priority Severity: Medium	
Date-Result: 05/05/2024 - Pass	

Test ID: CD025	Test Type/Category: Functional Unit Testing

Summary/Title/Objection: Validation of Required Fields for Instructor Sign-Up	
Step	Expected Output
1. Sign up as instructor without filling all the required fields and/or with filling them in the wrong format	The application gives a warning indicating that all the required fields should be filled and/or fields should be filled in the correct format in order to sign up
Priority Severity: High	
Date-Result: 07/05/2024 - Pass	

Test ID: CD026	Test Type/Category: Functional Unit Testing
Summary/Title/Objection: View Lab Analytics	
Step	Expected Output
1. Login as instructor or TA and navigate to the lab analytics page	View lab analytics such as the most asked questions, average lab grade etc.
Priority Severity: High	
Date-Result: Not implemented	

Test ID: CD027	Test Type/Category: Functional Unit Testing
Summary/Title/Objection: Validation of Required Fields for Course Addition	
Step	Expected Output
1. Login as instructor and navigate to the add course page	Add course page is displayed
2. Click the add a course button without filling all of the required fields and/or with filling them in the	The application gives a warning indicating that all the required fields should be filled and/or fields should be filled in the correct

wrong format	format in order to add a course
Priority Severity: High	
Date-Result: 07/05/2024 - Partially passed	

Test ID: CD028	Test Type/Category: Functional Unit Testing
Summary/Title/Objection: Use of Features Based on Instructor Permissions	
Step	Expected Output
1. Log in as a student	Student dashboard is opened
2. Click on the course name on the dashboard	Course page is opened
3. Click on the lab page with already uploaded lab	The lab page opened with submitted files is shown. If instructor enabled, students can see from how many test cases the lab has passed and what is the plagiarism rate.
4. Click on the chatbot button	If the instructor enabled it, the page is directed to the chatbot page for students to use.
Priority Severity: Medium	
Date-Result: 07/05/2024 - Partially passed	

Test ID: CD029	Test Type/Category: Functional Unit Testing
Summary/Title/Objection: Validation of File Type for Checking for Test Cases and Code Analysis	
Step	Expected Output
1. Log in as a student and navigate to	Upload code analysis page is opened.

the upload code to the analysis page.	
2. Click to add a file and upload a file that is not a code file	The application gives a warning indicating that the user should upload a code file
Priority Severity: High	
Date-Result: 07/05/2024 - Pass	

5.2 Non-functional Test Cases

Test ID: CD030	Test Type/Category: Non-functional Testing
Summary/Title/Objection: Testing the System Scalability	
Step	Expected Output
1. Simulate a high number of users accessing the app at the same time	The app's performance remains stable and it efficiently manages the increased load.
Priority/Severity: High	
Date-Result: 04/05/2024 - Pass	

Test ID: CD031	Test Type/Category: Non-functional Testing
Summary/Title/Objection: Testing the System Stability and Reliability	
Step	Expected Output
1. Explore the app's features by clicking buttons, inputting data, navigating pages, repeating actions variably, and monitoring performance.	The app needs to stay stable and function reliably, without crashing or behaving unpredictably, regardless of the sequence or repetition of different actions performed by the user.
Priority/Severity: Critical	

Date-Result: 08/05/2024 - Pass

Test ID: CD032	Test Type/Category: Non-functional Testing
Summary/Title/Objection: Testing the System Compatibility	
Step	Expected Output
1. Test the application on different devices with different screen sizes	The app and its user interface and all of its functionality is consistent across all devices and screen sizes
2. Test the application on different browsers and on different operating systems	The app and its user interface and all of its functionality is consistent across all browsers and operating systems
Priority/Severity: High	
Date-Result: 05/05/2024 - Pass	

Test ID: CD033	Test Type/Category: Non-functional Testing
Summary/Title/Objection: Testing the User Interfaces	
Step	Expected Output
1. Click on the words in the navigation bar	The app navigates the user to the related page
2. Open the chatbot and type in a chat.	The user's message is displayed as well as the response of the chatbot
3. Check the plagiarism checker, test case runner, and code analyzer pages.	All the user interfaces of the plagiarism checker, test case runner, and code analyzer components are displayed correctly with accurate information

4. Check the course, lab, assignment, and profile pages	All the relevant information about each course, lab, assignment, and profile should be displayed
Priority/Severity: High	
Date-Result: 08/05/2024 - Pass	

Test ID: CD034	Test Type/Category: Non-functional Testing
Summary/Title/Objection: Testing the System Security	
Step	Expected Output
1. Attempt common security attacks (SQL injection, cross-site scripting, etc.) against the application to assess its vulnerability.	The application successfully blocks these attempts and logs the incidents without exposing sensitive data or compromising system integrity.
Priority/Severity: High	
Date-Result: 07/05/2024 - Pass	

Test ID: CD035	Test Type/Category: Non-functional Testing
Summary/Title/Objection: Load Testing	
Step	Expected Output
1. Incrementally increase the load on the system by simulating multiple requests to the server over a sustained period.	The system sustains its performance under pressure, with response times remaining within acceptable thresholds until a specific load limit is reached.
Priority/Severity: High	
Date-Result: 07/05/2024 - Pass	

5.3 User Acceptance Test Cases

Test ID: CD036	Test Type/Category: User Acceptance Testing
Summary/Title/Objection: Student acceptance and evaluation testing	
Step	Expected Output
1. Publish the ethics form for students to fill out before publishing the evaluation form	Ethics forms are filled and copies are saved to the database system.
2. Publish the evaluation form for students to fill	Evaluation forms are filled and the copies are saved to the database system. Numerical questions are expected to be more than 6 out of 10 on average and the number of positive open-ended question answers are expected to be more than the neutral and negative ones.
Priority/Severity: Medium	
Date-Result: 07/05/2024 - Pass	

Test ID: CD037	Test Type/Category: User Acceptance Testing
Summary/Title/Objection: TA/Tutor acceptance and evaluation testing	
Step	Expected Output
1. Publish the ethics form for TA's and tutors to fill out before publishing the evaluation form	Ethics forms are filled and copies are saved to the database system.

2. Publish the evaluation form for TA's and tutors to fill	Evaluation forms are filled and the copies are saved to the database system. Numerical questions are expected to be more than 6 out of 10 on average and the number of positive open-ended question answers are expected to be more than the neutral and negative ones.
Priority/Severity: Medium	
Date-Result: 07/05/2024 - Pass	

6 Maintenance Plan and Details

To ensure the long-term success and efficiency of ***CODED.*** application, we have developed a comprehensive maintenance strategy that encompasses corrective, adaptive, perfective, and preventive maintenance. Our plan is designed to handle everything from routine updates to unexpected issues effectively.

We will implement continuous system monitoring and a user-friendly feedback mechanism to rapidly address and resolve any bugs or operational issues. This ensures the system remains functional and responsive to the changing technological environment. Regular updates to software libraries and adjustments to the infrastructure will be made to accommodate growth in user numbers and system demands. This proactive approach not only addresses current system needs but also adapts to future technological advancements.

Our focus will also extend to enhancing system performance and usability based on user feedback, which involves introducing new features and optimizing existing functionalities. Regular code refactoring and security measures will be implemented to prevent potential system degradation and security breaches. This combination of ongoing

refinement and preventative strategies is essential for maintaining the robustness of the application, ensuring it continues to meet user needs effectively while remaining secure and efficient.

7 Other Project Elements

7.1 Consideration of Various Factors in Engineering Design

This section dives deep into the inner workings of the ***CODED.*** system. We'll explore how it prioritizes data privacy while ensuring accessibility and fairness for all users. We'll also discuss how it benefits the public good and acknowledges the global context, including cultural, social, environmental, and economic considerations. The following considerations in engineering design are based on the detailed analysis in our Detailed Design Report [10].

7.1.1 Data Privacy Considerations

CODED. places a strong emphasis on **data privacy**. As a core feature, the chatbot integrates robust measures to ensure adherence to the General Data Protection Regulation (GDPR). It utilizes Amazon Relational Database Service (RDS) for secure and efficient data management. The chatbot, built upon a fine-tuned version of OpenAI's API, prioritizes GDPR's principles of personal data protection. This means user data is handled lawfully, fairly, and only for specified purposes [11]. Our databases employ advanced encryption and anonymization techniques to protect sensitive user information.

The use of Amazon RDS within ***CODED***.'s infrastructure is instrumental in upholding data privacy. RDS offers a highly secure database environment with encryption for both stored and transmitted data, automated backups, and network isolation [12]. These features

fulfill GDPR's requirements for implementing robust security measures to mitigate risks of data breaches or unauthorized access.

Beyond technical safeguards, ***CODED.***'s chatbot adopts a user-centric approach to data privacy. It provides clear explanations of data handling practices, obtains explicit user consent before data collection, and empowers users with control over their personal data. This includes the rights to access, rectify, or delete information, in line with GDPR's focus on user rights and data minimization.

7.1.2 Accessibility and Equality Considerations

CODED. emphasizes accessibility and equality throughout its engineering design. This is reflected in both the overall application design and the chatbot feature. The application interface is intentionally user-friendly, making it easy for even those with limited technical experience to navigate. This focus on beginners includes clear menus, easy-to-read fonts, and adaptable design for various devices and screen sizes.

The ***CODED.*** chatbot prioritizes fairness and the avoidance of bias. It follows ethical guidelines to ensure equal and supportive interactions regardless of a user's background [13]. This means the system actively works to prevent biased responses and guarantees reliable and accurate information for everyone [14].

Regular updates to the application and chatbot incorporate the latest learning trends and user feedback. This adaptability ensures the system continues to meet the needs of a diverse group of students and faculty, further demonstrating ***CODED's*** dedication to providing an equitable learning environment.

7.1.3 Public Welfare Considerations

CODED. aims to improve the learning environment and directly promote student skill development. This focus on skill-building is essential for public welfare: a skilled workforce boosts employability and economic growth [15].

Beyond employability and economics, CODED. embraces the cutting-edge trend of large language models (LLMs) in computer science [16]. Market research shows AI's growing role in education and CODED. positions itself as a pioneer with its innovative and comprehensive feature set. This demonstrates ***CODED's*** dedication to positively impacting education through the advancement of artificial intelligence.

7.1.4 Global Considerations

CODED. is intentionally designed for global adaptability, making it suitable for universities following US or European education systems. Its flexibility ensures that it functions seamlessly within diverse academic structures. The application is highly customizable, allowing educators to tailor features to their specific needs. This customization guarantees swift and successful integration into computer science labs worldwide.

7.1.5 Environmental Considerations

CODED. aims to reduce the environmental impact of CS and CTIS labs. By digitizing processes like grading and submissions it significantly cuts down on paper use, contributing to less deforestation and waste. Additionally, ***CODED.***'s support for remote learning setups could further benefit the environment. Allowing students and staff to work online it potentially reduces the need for commuting, which in turn lessens transportation-related carbon emissions [17].

However, it's important to acknowledge that ***CODED.***'s use of the OpenAI API has an environmental cost. This advanced technology relies on energy-intensive computing, leading to higher electricity consumption and increased carbon emissions [18].

7.2 Ethics and Professional Responsibilities

7.2.1 Professional Responsibilities

We targeted high professional standards throughout our project, focusing on inclusivity and accessibility. Our design approach was inclusive, aiming to accommodate users of all technical backgrounds and physical abilities. Professionally, we maintained a commitment to clear and effective communication, both within our team and with external stakeholders, such as **CS** and **CTIS** departments of Bilkent University. This included regular updates and transparent meetings regarding on our progress, which helped in building trust and maintaining software. Our documentation for both students and instructors was thorough and aimed at providing users with clear guidance on using the application, further reflecting our professional approach.

7.2.2 Ethical Responsibilities

CODED. was guided by strict ethical standards, particularly in handling data and maintaining academic integrity. We implemented robust encryption and secure data storage mechanisms to ensure compliance with the **General Data Protection Regulation (GDPR)** and **KVKK (Kişisel Verilerin Korunması Kanunu).** This safeguarded the privacy and security of user data, ensuring that it was only used for its intended purpose and accessed by authorized personnel. Additionally, our integration of the **MOSS** tool for plagiarism detection underscored our commitment to academic honesty. This feature was designed to transparently

analyze and report similarities in code submissions, thereby upholding the academic standards of the institutions utilizing our software.

7.3 Teamwork Details

The success of the ***CODED.*** project relied heavily on the strong teamwork and collaborative approach adopted by our team. We strategically divided tasks to optimize efficiency while fostering a supportive environment where everyone's expertise could contribute to problem-solving. Specific areas of focus, detailed below, illustrate how we functioned as a cohesive unit.

7.3.1 Contributing and Functioning Effectively on the Team

Every team member was given a specific task to focus on in order to function effectively and finish our application on time. When we were writing a report, we divided it equally so that each member had a contribution to each report. We first started our project by implementing the user interfaces, and in order to do so, we divided the pages equally between the team members. After that, we started to implement the specific components of our project. Emre developed the chatbot component mainly, and Beyza and Zeynep Hanife were working on the Test Case Runner component. Selin was dealing with the Code Analysis Component, and Zeynep Selcen was working on the development of the general backend and the Plagiarism Checker component. As can be seen from our division, although some people work more on the front-end or back-end, each member has done both the back-end and front-end.

7.3.2 Helping Creating a Collaborative and Inclusive Environment

While each team member spearheaded a specific component, we always maintained a strong collaborative spirit. When faced with coding challenges, we didn't hesitate to create GitHub issues, ensuring the team had full visibility into potential roadblocks. This transparency fostered a sense of shared problem-solving, allowing everyone to contribute their knowledge and ideas. Additionally, we actively used WhatsApp and Zoom for real-time communication, discussing issues and collaboratively developing solutions. This constant flow of information kept the entire team aligned and moving forward. We recognized the importance of teamwork – even with clearly defined roles, the success of the application rested on our ability to support each other and work towards a shared goal.

7.3.3 Taking a Lead Role and Sharing Leadership on the Team

In our project with many features, each team member assumed a leadership role in a specific area. Within the Analysis and Requirements, we created work packages containing various tasks and assigned a leader to each. We used these work packages as a blueprint during code implementation, while also assigning leadership roles for more granular tasks.

Leveraging our diverse technological expertise gained through internships and past projects, we strategically assigned leadership positions. For instance, Emre led chatbot development due to their prior internship experience in chatbot creation. Similarly, Beyza led the development of the user interface (UI), and Zeynep Hanife led the development of the test case runner, both leveraging their experience with these technologies from previous projects and internships. Given her research experience in code analysis with Assistant Prof. Eray Tüzün, Selin took the lead role in the code analysis component. Drawing on her past internship experience with SpringBoot, Zeynep Selcen assumed leadership for implementing the general backend as well as the plagiarism checker component.

7.3.4 Meeting Objectives

CODED. successfully met all predefined objectives, a testament to our strategic planning, dedicated execution, and effective collaboration. At the beginning of this project, we established clear, measurable goals aligned with ***CODED.'s*** vision and the needs of our stakeholders, mainly CS and CTIS departments. These objectives were objectively designed to challenge our capabilities while ensuring they were achievable within the set timeline, 8 months.

Throughout the project, we maintained a rigorous adherence to our timeline, employing agile methodologies to ensure flexibility and responsiveness to any unforeseen challenges. Regular team meetings and reviews allowed us to monitor our progress and make necessary adjustments, ensuring that each project milestone was met with precision. The utilization of project management tools such as Jira and continuous integration practices such as Github Organizations further enhanced our workflow, enabling efficient feature development and deployment.

The successful completion of all project deliverables not only demonstrates our team's technical competence and commitment but also enhances our credibility and sets a strong foundation for future initiatives. We have not only achieved what we promised but have also established a robust platform that can be scaled and adapted for future educational and development needs.

7.4 New Knowledge Acquired and Applied

Since this being our first application development from start to release, we encountered a lot of issues, unexpected news and things that haven't occurred to us before.

Throughout the development of our project, the integration of new technical skills was paramount, particularly in areas of cloud computing, software testing, and artificial intelligence. Our team gained proficiency in deploying and managing cloud-based services using Amazon Web Services (AWS), specifically AWS RDS for database management and AWS S3 for storage solutions. This knowledge was crucial for ensuring secure and efficient data handling within our application. We also advanced our project management capabilities by adopting GitHub for version control, using a feature-branch workflow that significantly improved our ability to manage development tasks effectively and minimized integration issues.

In the realm of software testing and quality assurance, we learned to use pytest for the automated testing of students' code, enhancing the robustness and functionality of our test case runner component. SonarCloud was instrumental in conducting thorough code analyses, while MOSS was vital for performing similarity checks to detect plagiarism in code submissions. Additionally, our project's chatbot feature, powered by the OpenAI API, was a significant learning curve. We harnessed this technology to develop an intelligent interface that could interact with users in a meaningful way, providing automated responses and assistance with natural language processing capabilities.

Reflecting on this project, the acquisition and application of new technical knowledge profoundly impacted our success and fostered significant personal and professional growth within our team. By navigating through these challenges and integrating new technologies such as the OpenAI API for our chatbot, we not only enhanced our technical proficiency but also prepared ourselves for future challenges in the technology sector. This journey underscored the continuous learning process in software development, turning theoretical knowledge into practical skills that are essential for our future endeavors.

8 Conclusion and Future Work

8.1 Conclusion

As we conclude our senior design project, it is essential to reflect on the journey we have undertaken, the achievements we have made, and the lessons we have learned. This project was not just about fulfilling academic requirements or developing technical skills; it was about creating a tool that could genuinely enhance the educational experience for students and educators alike.

CODED., which centered around developing a sophisticated software system to assist in the administration of academic courses and labs, has reached its objectives successfully. We set out to create a platform that not only automates various aspects of lab management but also integrates advanced features like a chatbot for student queries, plagiarism detection, and automated code analysis. Each feature was designed with the user in mind, aiming to streamline the educational process and enhance learning outcomes.

The implementation of AWS services for database, deployment and storage solutions, the integration of the MOSS and SonarCloud for maintaining academic integrity, and the development of a user-friendly interface using the latest web technologies are testaments to our commitment to quality and innovation. Our extensive testing procedures mentioned in the test cases section ensured that the software is robust and reliable, ready to be deployed in a real-world educational setting.

Through this project, we have not only developed a functional product but also contributed to the field of educational technology by addressing some of the pressing

challenges in academia today, such as the need for effective plagiarism detection and the demand for AI tools that needed today's education specturm.

8.2 Future Work

Looking ahead, there are several avenues for further development and expansion of ***CODED***. First, we intend to explore the integration of more advanced AI features within the chatbot to enhance its capability in handling more complex student queries and providing more personalized assistance. Machine learning algorithms could be employed to analyze student interaction patterns and adapt the learning content accordingly.

Additionally, scalability is a critical area for future development. As the user base grows, ensuring that the platform can handle a significant increase in simultaneous users without degradation in performance will be essential. This may involve optimizing our cloud infrastructure and possibly employing more sophisticated data caching and load balancing techniques.

We also see potential in expanding the platform's reach to encompass a broader range of educational institutions and learning contexts, including K-12 schools and professional training programs. This would require adjusting the platform's features to meet the specific needs of these different educational settings.

Last but not least, we aimed to realize our application in a startup environment in the coming years.

9 User Manual

In order to use ***CODED.*** application easily, we have developed an enhanced user documentation for both students and instructors. The user documentation is divided into two separate sections to cater specifically to the distinct needs of students and instructors. Each section is hosted on **GitBook**, providing a user-friendly interface and interactive navigation to help you find the information you need quickly.

- **Students' Documentation Link:**

  https://coded-inc.gitbook.io/documentation-students

- **Instructors' Documentation Link:**

https://coded-inc.gitbook.io/documentation-instructors

10 Glossary

- **Amazon Elastic Compute Cloud (EC2):** A web service that provides secure, resizable compute capacity in the cloud. It is designed to make web-scale cloud computing easier for developers. Amazon EC2 offers a wide selection of instance types optimized to fit different use cases. Instances can be resized and the number of instances scaled up or down automatically, depending on demand, to manage changes in requirements or spikes in popularity, reducing the need to forecast traffic. EC2 provides developers the tools to build failure resilient applications and isolate them from common failure scenarios.

  Acronym: EC2

- **Amazon Relational Database Service (RDS):** A web service that simplifies the setup, operation, and scaling of a relational database for use in applications. It provides cost-efficient and resizable capacity while managing time-consuming database administration tasks, freeing users to focus on their applications and business. Amazon RDS supports several types of database engines, including MySQL, PostgreSQL, MariaDB, Oracle, and Microsoft SQL Server [19].

  Acronym: RDS

- **Amazon Simple Storage Service (S3):** A web service designed to store and retrieve any amount of data, at any time, from anywhere on the web. It offers a scalable, high-speed, web-based cloud storage service with high durability and availability. Amazon S3 is designed for online backup and archiving of data and applications on Amazon Web Services (AWS). It supports data storage and retrieval using a simple web service interface, allowing for a wide variety of use cases from websites to data

analytics. S3 is known for its robust security features, including access management tools and encryption options to protect data.

Acronym: S3

- **Amazon Web Services (AWS):** A comprehensive and broadly adopted cloud platform that offers over 200 fully featured services from data centers globally. AWS provides a variety of building blocks that can be assembled in order to create any type of application in the cloud. It offers services across computing, storage, databases, analytics, networking, mobile, developer tools, management tools, IoT, security, and enterprise applications. This platform is designed to help organizations move faster, lower IT costs, and scale applications, providing the flexibility to use the services individually or in combination, based on business needs.

  Acronym: AWS

- **Artificial Intelligence (AI)**: An acronym that refers to the simulation of human intelligence in machines that are programmed to think and act like humans. This includes capabilities such as learning, reasoning, self-correction, and problem-solving.[19].

  Acronym: AI

- **Application Programming Interface (API)**: A set of protocols and tools that allow different software applications to communicate with each other, enabling the exchange of data and functionalities [19].
  - **Acronym**: API

    - **Example**: The Hugging Face API allows developers to access and utilize a variety of machine learning models for natural language processing tasks, such as text generation, translation, and sentiment analysis.
- **Computer Science (CS)**: The study of computers and computational systems, focusing on algorithms, software design, and the theoretical underpinnings of computing [19].
    - **Acronym**: CS
- **Comma-Separated Values (CSV)**: A file format used to store tabular data, such as a spreadsheet or database, in plain text. Each line in a CSV file corresponds to a row in the table, and each field in that row (or cell in the table) is separated by a comma. This format is widely supported by many applications and is useful for transferring data between different programs [19].
    - **Acronym:** CSV
- **General Data Protection Regulation(GDPR)**: It is a regulation in EU law on data protection and privacy in the European Union and the European Economic Area [19].
    - **Acronym**: GDPR
- **Family Educational Rights and Privacy Act (FERPA):** A federal law in the United States that protects the privacy of student education records. It gives parents certain rights with respect to their children's education records; these rights transfer to the student, or "eligible student," when they reach the age of 18 or attend a school beyond the high school level. Schools must have written permission from the parent or eligible student to release any information from a student's education record, though there are certain exceptions. FERPA also allows parents or eligible students the right to inspect and review the student's education records maintained by the school [19].
    - **Acronym:** FERPA

- **Kişisel Verilerin Korunması Kanunu (KVKK):** A Turkish law designed to protect the privacy of individuals by regulating the processing of personal data. It establishes the principles for data processing, the rights of data subjects, and the obligations of data controllers and processors. The KVKK requires data controllers to obtain explicit consent from individuals before processing their personal data, except in specific circumstances defined by law. It also grants individuals the right to inquire about their personal data, request its correction or deletion, and learn if it has been used appropriately. The law aims to ensure personal data is protected and processed securely, in alignment with international standards on privacy [19].
    - **Acronym:** KVKK
- **Information Systems and Technologies (CTIS)**: An interdisciplinary field that combines computing technology with business practices to facilitate data storage, analysis, and communication within organizations [19].
    - **Acronym**: CTIS
- **Large Language Models (LLM)**: Machine learning models trained on extensive datasets to understand and generate human-like text based on the input they receive [19].
    - **Acronym**: LLM
    - **Example**: GPT-4 is an example of a large language model used for various natural language processing tasks.
- **Measure of Software Similarity (MOSS)**: An automated system for detecting plagiarism in software assignments. It is commonly used in academic settings to compare student submissions and identify similarities that may indicate copying [19].
    - **Acronym**: MOSS

    - **Example**: Many universities use MOSS to maintain academic integrity in computer science courses by checking if students have plagiarized code for their assignments.

- **Science, Technology, Engineering, and Mathematics (STEM)**: An interdisciplinary approach to learning that integrates these four disciplines into a cohesive curriculum focused on real-world applications [19].
    - **Acronym**: STEM
    - **Example**: Many educational initiatives aim to boost student interest and competence in STEM fields due to their importance in the modern workforce.
- **Teaching Assistant (TA)**: An individual, often a graduate or undergraduate student, assists a professor in instructional responsibilities, including grading, answering student queries, and occasionally teaching [19].
    - **Acronym**: TA

Two Factor Authentication(2FA): A security process that requires users to provide two different forms of identification before gaining access to an account or system. This adds an extra layer of security compared to single-factor authentication [19].

Acronym: 2FA

11 References

[1] K. Emre, "Discussion on CS labs in Bilkent University," Academic interview, Oct. 28, 2023.

[2] "Code smell," *Wikipedia*. [Online]. Available: https://en.wikipedia.org/wiki/Code_smell. [Accessed: Oct. 28, 2023].

[3] "Khanmigo AI," Khanmigo.ai, [Online]. Available: https://www.khanmigo.ai/. [Accessed: Accessed: Mar. 14, 2024].

[4] "CodeGrade: Deliver Engaging Feedback on Code," CodeGrade.com, [Online]. Available: https://www.codegrade.com/. [Accessed: Mar. 14, 2024].

[5] "What is Personal Data?," Information Commissioner's Office, [Online]. Available: https://ico.org.uk/for-organisations/uk-gdpr-guidance-and-resources/personal-information-what-is-it/what-is-personal-data/what-are-identifiers-and-related-factors/#:~:text=The%20username%20is%20personal%20data,%27real%20world%27%20named%20individual. [Accessed: Accessed: Mar. 14, 2024].

[6] "How can bright colors enhance UI?," Design4Users, [Online]. Available: https://design4users.com/bright-colors-ui-design/#:~:text=Vibrant%20colors%20enable%20enough%20contrast,may%20not%20always%20work%20well. [Accessed: Accessed: Mar. 14, 2024].

[7] Hibernate ORM, "Object/Relational Mapping," Hibernate ORM, Available: https://hibernate.org/orm/. [Accessed: May 7, 2024].

[8] Spring Security, "Spring Security," Spring, Available: https://spring.io/projects/spring-security. [Accessed: May 7, 2024].

[9] SonarCloud, "SonarCloud Documentation," SonarSource, Available: https://docs.sonarsource.com/sonarcloud/. [Accessed: May 7, 2024].

[10] Z. H. Akgül, B. Çağlar, S. B. Gündoğar, E. Karataş, Z. S. Öztunç, "Detailed Design Report," Bilkent University, CS 492 Senior Design Project, Project ID: T2322, March 15, 2024.

[11] OpenAI, "Security," OpenAI, [Online]. Available: https://openai.com/security. [Accessed: Nov. 28, 2023].

[12] Amazon Web Services, Inc., 'Data protection in Amazon RDS - Amazon Relational Database Service,' Amazon Relational Database Service User Guide. [Online]. Available: https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/DataDurability.html. [Accessed: Nov. 28, 2023].

[13] J. Bang, S. Kim, J. W. Nam, and D.-G. Yang, "Ethical Chatbot Design for Reducing Negative Effects of Biased Data and Unethical Conversations," in Proc. 2021 Int. Conf. Platform Technol. Service (PlatCon), 2021. DOI: 10.1109/PlatCon53246.2021.9680760. [Accessed: Nov. 28, 2023].

[14] L. Ranaldi, E. S. Ruzzetti, D. Venditti, D. Onorati, and F. M. Zanzotto, "A Trip Towards Fairness: Bias and De-Biasing in Large Language Models," 2023. [Online]. Available: https://arxiv.org/pdf/2305.13862.pdf . [Accessed: Nov. 28, 2023].

[15] T. Zobel, H. Steinbeck, and C. Meinel, "Towards Inclusive Education: A Review of and Guide to Accessible Features in Chatbots for MOOCs," 2023 IEEE Learning with MOOCS (LWMOOCS), 2023, pp. 1-5. DOI: 10.1109/LWMOOCS58322.2023.10306062.

[16] L. A. Jacques, "Teaching CS-101 at the Dawn of ChatGPT," acm Inroads, vol. 14, no. 2, pp. [pp 3-6], June 2023. DOI: 10.1145/3595634.

[17] M. P. Jarillo, L. Pedraza, P. M. Ger, and E. Bocos, "Challenges of Online Higher Education in the Face of the Sustainability Objectives of the United Nations: Carbon Footprint, Accessibility and Social Inclusion," Sustainability, vol. 11, no. 20, p. 5580, 2019. DOI: 10.3390/su11205580.

[18] "The Carbon Footprint of GPT-4," Towards Data Science, [Online]. Available: https://towardsdatascience.com/the-carbon-footprint-of-gpt-4-d6c676eb21ae. [Accessed: March 12, 2024].

[19] OpenAI, "ChatGPT [Large language model]," 2024. [Online]. Available: https://chat.openai.com.