Bilkent University

Department of Computer Engineering

CS 492

Senior Design Project

CODED.

2023-24 Spring Semester

Detailed Design Document

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

15 March 2024

Table of Contents

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 based on the instructor's criteria. The instructor can compare the code to a specific section, the whole class, the code on Github, past codes, or check for similarity in AI (Artificial Intelligence) generated code. Furthermore, the code will be passed by test cases provided by the instructor or created by the chatbot. The tool will grade the code based on the given criteria in a fair manner by looking at the main logic of the code and running every method separately if needed to check for correctness. Furthermore, the code will be statically checked for partial point calculations. The instructors will be given an insight into how the grading was tested and which sections of the code led to errors to take off points. The students will be able to view their grading report if the instructors allow it on the application, and students will be able to refute their generated grades. Therefore, the application will tremendously shorten 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 as some students prefer to code in dark mode integrated development environments (IDE) and they may prefer their screen to stay dark. Moreover, every page has a navigation bar, so reaching any main feature will require only two operations. Other than reaching the main tools, any system feature will be accessed within at most four clicks or operations. The chatbot, one of our application's main features, will be accessible and easily seen on any page as it will be on the navigation bar at the top of the screen. 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 lab session. Each student is expected to ask multiple questions to the chatbot throughout a session and utilize the code analyzer feature, and upload their code to the system at least once. ***CODED.*** will be able to handle the usage of multiple users without a significant performance slowdown and the system's response time will be 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. The system will be able to maintain its performance with at least 200 users using the application simultaneously. Several APIs will be used during the development process, so the system's goal is to ensure that the third-party API calls have a latency of less than 1 second. Furthermore, it is planned that the chatbot will respond to any input in less than 5 seconds.

1.2.3 Reliability

The system will be available to all users at all times, including lab hours and practicing hours, with minimal downtime. More specifically, the website will maintain an availability of at least 99.9% days over a year, with a total downtime of no more than 9 hours annually. High-security databases will prevent data loss and ensure the safe storage of personal data. The website will also provide clear error messages to inform users in case of problems. Errors will be 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, code and lab statistics, 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 is 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 service 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 bootcamps or high school coding classes. Currently, we are in contact with the CS and CTIS departments, and we visited the CS 115, CTIS 151 and CTIS 152 lab sessions to gain first-hand experience to make sure our application is catered to the needs of students and instructors. Thus, our application will be ready to be used as a commercialized product by the end of the semester, and we hope to enter startup idea competitions to get further feedback on the commercialized product and perfect it further.

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 will employ 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 will implement several actions to strengthen security. User access to the system will be strictly controlled, limiting access to authorized personnel only: enrolled students, instructors, teaching assistants, and tutors. Students will not be able to enroll themselves in the application but the application will send an invitation through emails to students. This list of students to be enrolled in the application will be provided by the course instructor. Furthermore, in the cases of security breaches, the system will have an incident response time of up to 30 minutes.

1.2.6 Scalability

As mentioned in section 1.2.2, up to 200 users will be able to use the application simultaneously, and be open to users at all times. Throughout one lab session, we expect the maximum number of students to be 60; though, as the user volume increases, the performance and responsiveness of the system will remain consistent. For this purpose, we set a threshold for the website to accommodate at least 50% of the growing demand regarding the number of users and content volume. The importance of scalability will increase in the future as the system will be developed regarding the possibility of usage from different schools or even companies, growing the volume content of the application. Henceforth, scalability will further become more significant as the application's user base grows.

1.2.7 Maintainability

The application will be maintained throughout its lifecycle due to its heavy user interaction nature. We plan on completing the application by the end of March to start testing it in CS 115, CS 125, CTIS 151, and CTIS 152 labs. While testing our beta version, we expect to find bugs and obtain user feedback by creating a feedback property on the application. With the feedback we receive from students, we hope to improve features further or possibly add new features while increasing user satisfaction with the application. If any features cannot be implemented by the end of March, we plan on working on them in April. Overall, we expect to do maintenance throughout April and May. 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 will be kept in Github in our private repository, we plan to open issues and tickets on Github and connect them to JIRA. Furthermore, we heavily use 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 can 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. Furthermore, the instructor can choose to add the type of plagiarism check for the assignment, which is AI-generated code, section-based code, or past codes. 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.

As from the developer perspective, we have coded our project in model-view-controller (MVC), which will allow 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 appearing 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.

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, plagiarism detection, and an advanced grading tool. 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 the grading of lab assignments using custom test cases provided by instructors or teaching assistants. The tool analyzes the core logic of the code and tests each method individually for accuracy. It also conducts static checks for partial points. Instructors receive detailed insights into the grading process, including error identification, while students can access their grades and refute them if permitted by instructors. The system can also generate these test cases on demand, ensuring a comprehensive assessment of each student's work.

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 code through both file uploads and direct text input. This component is beyond only highlighting the errors and problems, but 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.

1.3.4 Plagiarism Check

The Plagiarism Checker component checks the originality of student submissions. It compares each submission against others and a pre-existing code database and generates a similarity rate. Based on this similarity rate, the code is suggested to be plagiarized or not. 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 Current Software Architecture

There are 4 main components in our application: Test Case Runners component, Chatbot component, Code Quality Checker component, and Plagiarism Checker component. The components call servers accordingly whenever users (instructors, students, and tutors/TAs) utilize the system. There is also the database component, which uses Amazon RDS to save each piece of information related to the system. The users, the instructor, students, and tutors use their devices to connect to the system and to be able to use its features. Their browsers make requests to the servers of the system's main components to be able to use their features. To see these relationships clearly, a high-level system architecture diagram is given below:

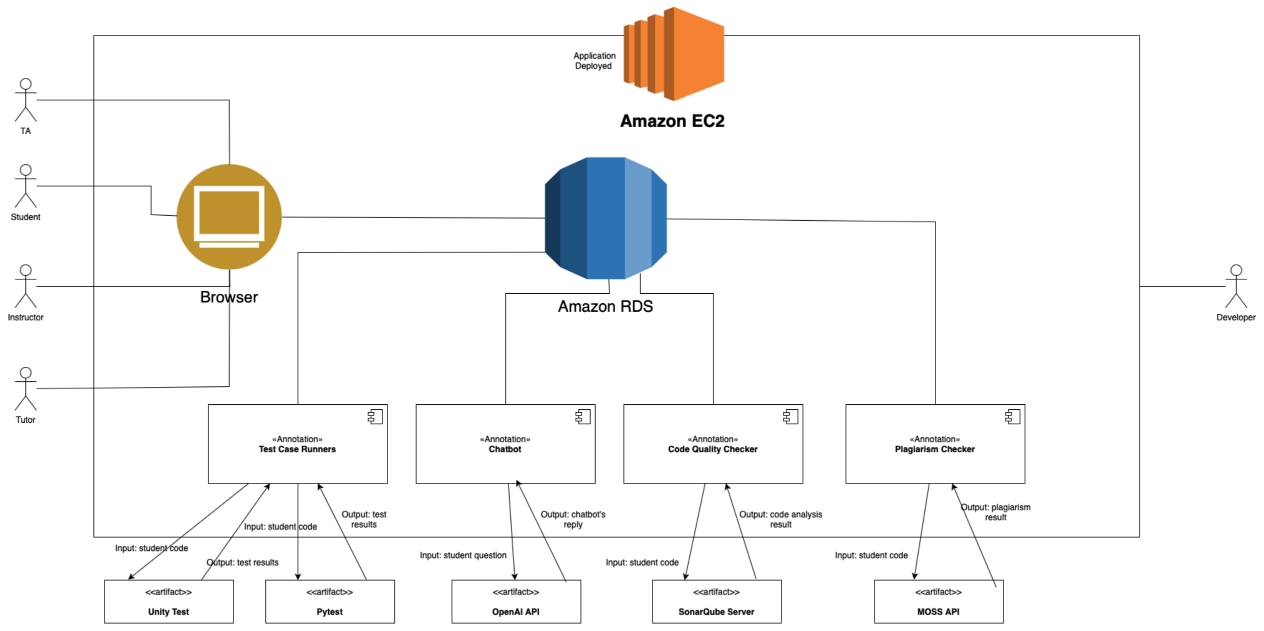

Fig. 1: Current Software Architecture

The Test Case Runners component is called when students upload their code to the system. The component has its own grading algorithm since partial points are given in labs, and the grading is not binary. In order to grade the labs, the component makes calls to different APIs (Application Programming Interface), UnityTest for C labs, and pytest for Python labs. It sends the students' codes to these APIs and obtains the test results, such as how many tests the code passed. Then, according to the specialized grading algorithm that the component has, it finalizes the grades for the students. These grades are saved to the database and then returned to the user's browser for viewing.

The Chatbot component is called when the students want to ask questions about the lab. Inside the component, prompt engineering will be used so that the chatbot only guides the student in their code process and does not actually give the solutions to the lab questions. When the chatbot component inputs students' chatbot inquiries, it sends this input to the OpenAI API. Then, it gets the chatbot’s reply to the student as an output from the API, and the reply is then immediately returned to the student's browser for viewing.

The Code Quality Checker component is called when the code the student uploads is to be checked in terms of quality. After it receives the code of the student, it calls on the SonarQube API in order for that code to be evaluated. Then, it receives the code evaluation from the API, and the resulting quality assessment is saved to the database and returned to the user's browser.

The Plagiarism Checker component compares the student codes with each other in order to understand whether an act of plagiarism has occurred. In order to do that, the students' code is sent to the MOSS API. From there, it obtains information on the similarity rates of each student code. The similarity results are saved to the database and then returned to the user's browser.

3 Proposed Software Architecture

3.1 Overview

This section explains the software architecture of the program in detail. ***CODED.*** is expected to be used in university labs by instructors, TAs, tutors, and students. Therefore, there needs to be a software plan encapsulating the needs of the various users. To demonstrate the application’s full functionality and system design different diagrams are included in this section. The subsystem decomposition discloses the layered structure of the technologies used in the implementation. The hardware/software mapping shows the type of hardware used as well as the relationship between the hardware components and the software. The persistent data management section describes the storage, retrieval, and maintenance of the data collected in the application. The access control and security section describes the security precautions taken to protect the data.

3.2 Subsystem Decomposition

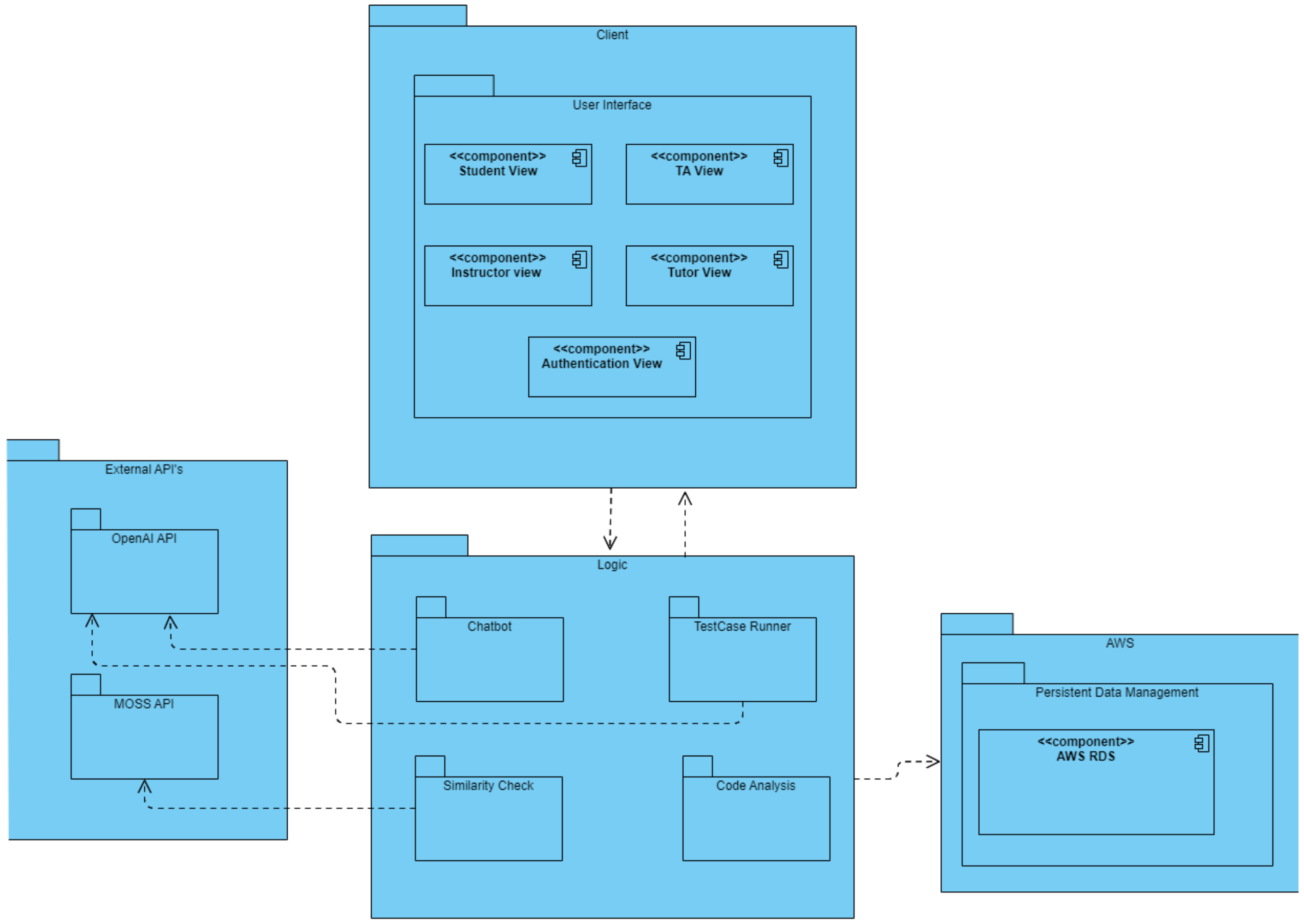

Fig. 2: Subsystem Decomposition of ***CODED.***

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 similarity check components interact with OpenAI and MOSS APIs respectively, which are included in the external API's layer. The logic layer also interacts with the AWS layer, which includes the relational database component. 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.

3.3 Hardware / Software Mapping

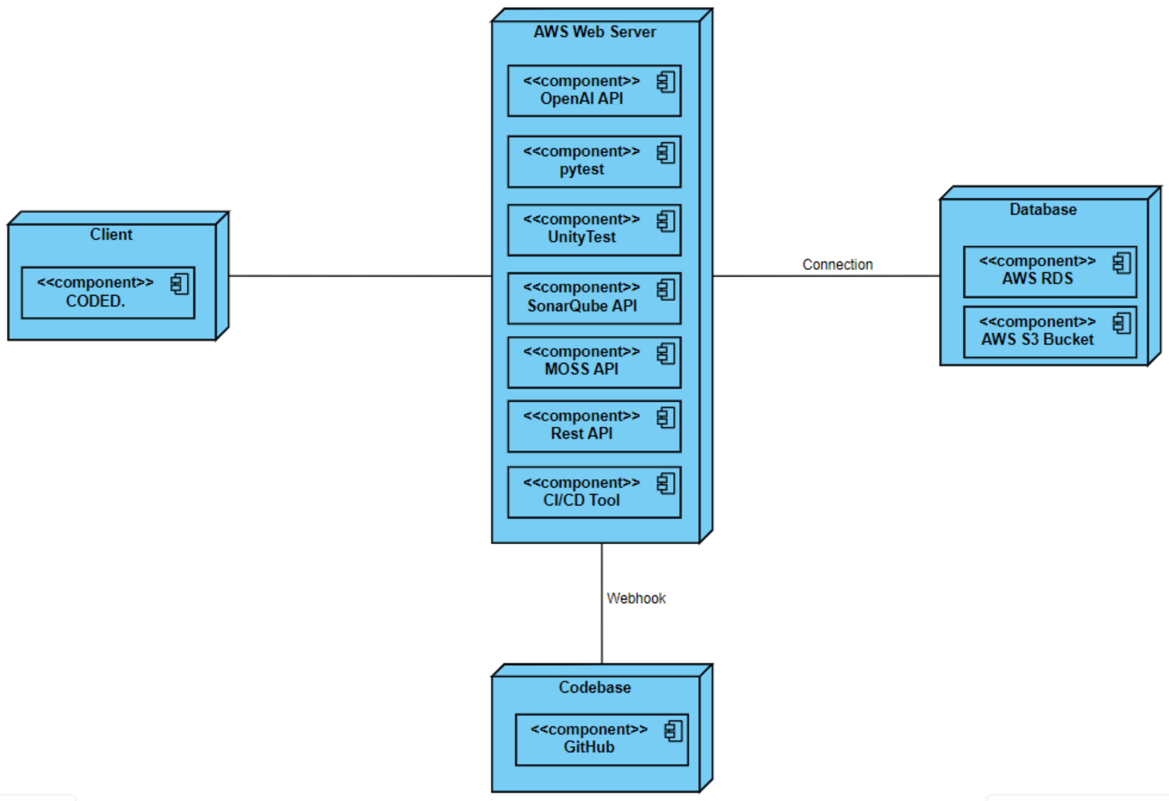

Fig. 3: Deployment Diagram of ***CODED.***

CODED. will be 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 will be deployed on an AWS cloud server using **AWS EC2**

(Amazon Web Services Elastic Cloud Compute). Therefore, all the logic layers, including Rest API, will 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 will be 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, will be used to store any required data. Thus, all user and application data will also be stored in the AWS cloud with necessary security precautions. The source code of the application is kept in **GitHub** which will be fed to the server while running the website. A **CI/CD tool** integrated into the server will help the code to be deployed and tested continuously.

3.4 Persistent Data Management

In ***CODED.*** application, persistent data management consists of using high-performance, privacy-prioritized, low-latency database services.

3.4.1 Text Data Management

In ***CODED.***, the data comprises both text and file data. We are using **PostgreSQL** in **the Amazon RDS** for text data. The text data includes login credentials, student information, course details, students' chat history, and further critical data. The basis for choosing Amazon RDS lies in the truth that its high-level security options together with low-latency and durable service providing success [7]. Considering the critical information (e.g., login credentials, grades) kept in our services, data privacy is the most important priority for ***CODED.*** We are using **Amazon Key Management Service (KMS)** to encrypt/decrypt the data at rest and in transit, compiling legal regulations such as KVKK (Schools in Turkiye), GDPR (Schools in EU/EEA Region), and FERPA (Schools in the US).

Besides **Amazon RDS** services, we are employing the robust features of the **PostgreSQL** database, such as effective table schema creation and moderation, table partitioning, keys, and indexes to provide safe, state-of-the-art data flow in the application [8].

3.4.2 File Data Management

Aside from the text data, ***CODED.*** gathers various file data types, ranging from students' code submissions to file instructions in PDF format. For that purpose, we are using **Amazon S3** to store and transmit file data records. Similar to Amazon's RDS features, **Amazon S3** provides high-scale, performant file storage with low latency and high durability [9]. To reduce input/output operations cost at the database systems for both text and file data. ***CODED.*** application uses the *Hot-Cold Data* approach, which saves the database costs in a large volume of traffic in the system.

3.5 Access Control and Security

Access control and security are among the most important key points of ***CODED.*** application, to ensure our users are flowing into the application without worrying about their privacy and security concerns. We are deploying a variety of data privacy and security algorithms and tools in our application to succeed in obeying the legal regulations issued by the EU, the US, and the rest of the world.

3.5.1 Login Security

CODED. application has multiple actors with different roles, such as students, instructors, TAs, and tutors. At the basic level, all system actors must log in to the system with their email and password credentials. As mentioned above, the aforementioned credentials are kept in the system in an encrypted version (industry-standard algorithm **AES-256**), along with the input/output and transmitting process [10]. We will also monitor unusual activities of the users by our in-built **Abuse Preventing System**, such as logging the system in the lab computer that they are not logged in with their school account. Besides students, instructors, TAs and tutors must log in to the system with 2FA authentication system in every login. We will be using **Google Authenticator** to achieve this aim. Google Authenticator will generate the one-time code every 30 seconds, and we will check whether the user input and code itself match to ensure system security. Similar to student login, the **Abuse Preventing System** will also check the instructors, TAs, and tutor accounts to log unusual activities in their accounts.

3.5.2 Access Security

For the server-side request handling process, we are implementing a spectrum of algorithms to protect the privacy and security of ***CODED.*** application. We have developed algorithms that require strict implementation of **HTTPS**, authentication, and authorization mechanisms to safeguard communications and ensure that only authorized users can access specific functionalities [11]. Using **OAuth2** with **JWT (JSON Web Tokens**) facilitates secure and efficient user authentication and role-based access control, which is critical for applications with diverse user roles like students, instructors, TAs, and tutors.

FastAPI's dependency injection system is instrumental in crafting reusable security dependencies, enabling streamlined verification of user credentials and roles per request, thereby enforcing access policies effectively. Additionally, **Pydantic models** are essential for robust input validation, automatically ensuring that incoming data conforms to the defined schemas, mitigating the risks of injection attacks.

3.5.3 Chatbot Security

Chatbot history may include personal and intellectual property. Considering the fact that OpenAI claims to take data privacy issues seriously, we also take some precautions to provide full-scale data privacy on our application side.

Every chat history is kept in our databases with an anonymized, encrypted version. We never ever keep **PII (Personal Identifiable Information)** in our dedicated database. To enhance this feature, we developed a **Personal Information Filtering System** based on **Microsoft Presidio**, which detects and carefully hashes (###) the personal information entered by the user [12]. As a result, any personal information sent by the user will not be visible, even to the developers.

In database design, we give high importance to the anonymization of chat histories of the students, to prevent de-anonymization attacks for revealing the true identity of the users. We will be using **Amazon RDS**'s in-built features to stop these cyber attacks.

Additionally, hackers mostly try to use **SQL injection** attacks to send queries to the backend by using chatbot prompts. To prevent these malicious attacks, we are implying two two-layer tokenization systems on both our application side and on the OpenAI API side. Furthermore, our chatbot has been trained more "strictly" and "conservative" in the case of SQL-related questions, to add an additional prevention layer to intercept these attacks [13].

4 Subsystem Services

4.1 Client Subsystem

4.1.1 User Interface

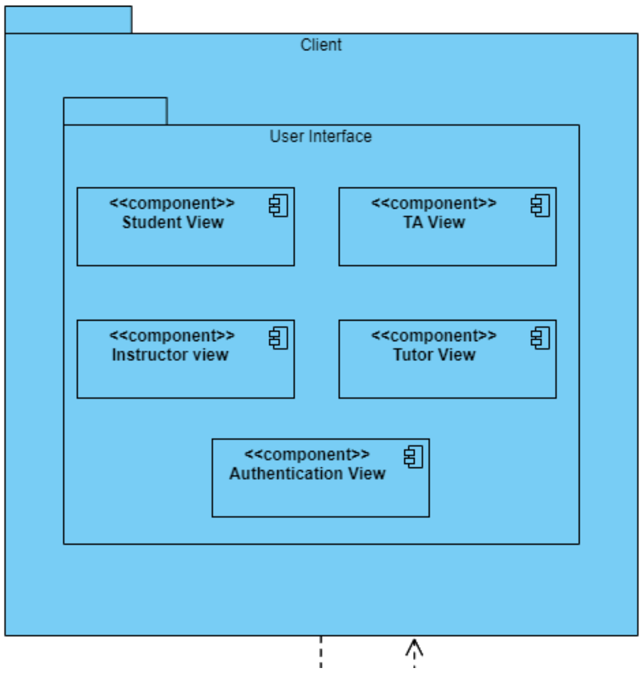
Client
User Interface
<<component>> Student View
<<component>> TA View
<<component>> Instructor view
<<component>> Tutor View
<<component>> Authentication View

Fig. 4: 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.***

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

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

Tutor View: Tutors are one type of user and when logged into the system, they interact with many distinct screens that allow them to experience the tutor 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.

4.2 Logic Subsystem

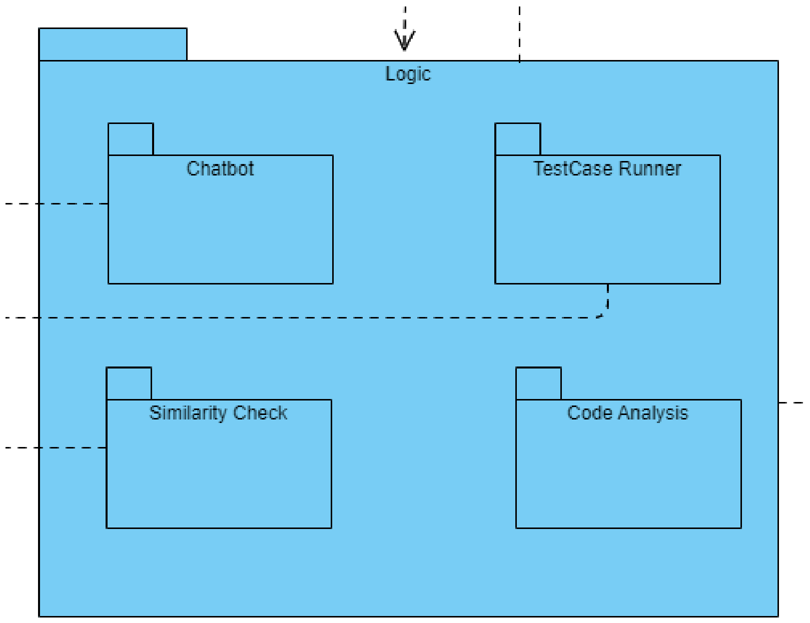

Fig. 5: Logic sub-module of ***CODED.***

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, and similarity checker components interact with the External API's layer. Whereas the code analysis component does not use APIs, but servers.

4.3 External API’s Subsystem

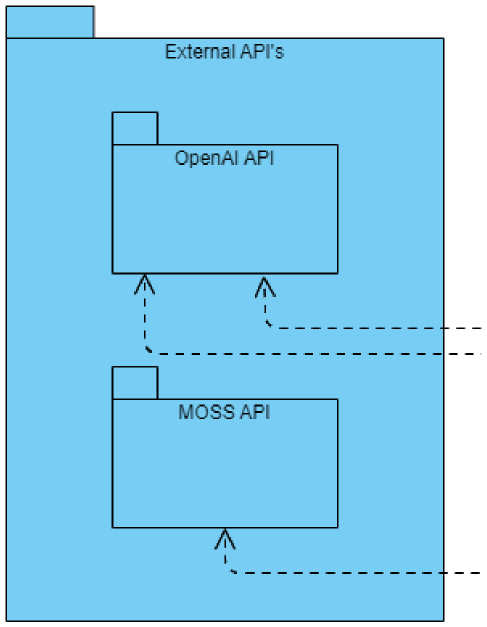

*Fig. 6: External API’s subsystem of **CODED.***

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.

4.4 AWS Subsystem

4.4.1 Persistent Data Management

4.4.1.1 AWS RDS

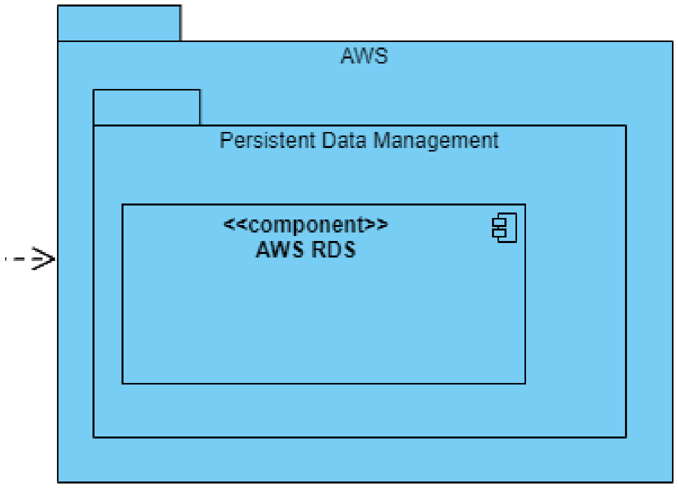

Fig. 7: AWS RDS sub-module of ***CODED.***

All data is stored in PostgreSQL within the Amazon Web Services 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.

5 Test Cases

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	

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	

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	

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	

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	

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	

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 to the code analysis.	Code analysis is done on the code. Along with the code analysis results, 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	

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

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 ‘Edit Submission’ button.	Edit submission page is opened. If the submission deadline is passed, a pop-up warning is displayed.
4. Click on the ‘Change File’ button next to the submitted file.	File explorer of the computer is opened.
5. Click on the ‘Save’ button.	File chosen is submitted and seen on the submitted files table with the new submission date.
Priority/Severity: High	

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 page.	Lab page is opened.
3. Click on the ‘Delete Submission’	A pop-up warning is displayed.

button.	
4. Click on the 'I'm Sure' button on the pop-up.	Lab submission is deleted.
Priority/Severity: High	

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	

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	

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	

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 list of test cases
Priority/Severity: Medium	

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	

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	

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	

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 mail 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	

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	

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	

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	

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	

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	

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	

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	

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

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 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 add a course
Priority Severity: High	

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	Lab page opened with submitted files are 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	

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 the upload code to the analysis page.	Upload code analysis page is opened.
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	

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	

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	

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	

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	

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	

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	

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	

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	

6 Consideration of Various Factors in Engineering Design

We talk in detail about how ***CODED.*** system is built, focusing on important issues like keeping data private, making sure it's accessible and fair to everyone, helping the public, and considering the different challenges around the world, including culture, society, the environment, and economic factors. The following considerations in engineering design are based on our detailed analysis in our previous report [14].

6.1 Data Privacy Considerations

CODED. considers Data Privacy to be a highly regarded issue. As one of the main features, ***CODED.***'s chatbot integrates sophisticated measures to ensure compliance with the **General Data Protection Regulation (GDPR)** and strictly uses **Amazon Relational Database Service (RDS)** for secure and efficient data management. The chatbot's design, which is a fine-tuned version of Open AI's API, follows GDPR's requirements for personal data protection, ensuring that user data is processed lawfully, transparently, and for legitimate purposes only [15]. Our databases utilize prosperous encryption and anonymization techniques to safeguard sensitive user information.

Moreover, the utilization of Amazon RDS in ***CODED.***'s infrastructure plays a crucial role in maintaining data privacy. Amazon RDS provides a highly secure environment for database management, featuring encryption at rest and in data transmitting, automated backups, and network isolation mechanisms [16]. This aligns with GDPR's mandates for employing advanced security measures to prevent data breaches and unauthorized access.

In addition to technical safeguards, ***CODED.***'s chatbot is designed with a user-centric approach to data privacy. It transparently communicates its data handling practices, seeks explicit user consent for data collection, and provides users with control over their personal data. This includes options for data access, rectification, and erasure, aligning with GDPR's principles of user rights and data minimization.

6.2 Accessibility and Equality Considerations

One of the main important factors for the engineering design of ***CODED.*** is making the CS and CTIS lab process accessible to everyone in an equal manner. This commitment to **equality** is evident in both the overall application design and the specific features like the chatbot. The application has been developed with a user-friendly interface that accommodates diverse user needs, including those who may not be proficient in technology, considering the fact that the target people of ***CODED.*** are the ones who learn programming as beginners. This includes clear navigation, readable fonts, and a responsive design that is compatible with various devices and screen sizes, ensuring **accessibility** for all users.

The chatbot, a prominent feature of ***CODED.***, is designed following ethical guidelines to avoid biases in interactions. This approach ensures that the chatbot interactions are fair and unbiased, providing equal support to all users regardless of their background [17]. ***CODED.*** system is designed to eliminate biased responses and guarantee that every user has access to accurate and reliable information, emphasizing the importance of equality in educational resources [18].

Additionally, the overall application and the chatbot are regularly updated to reflect the latest educational trends and user feedback, ensuring that the system evolves to meet the changing needs of a diverse user base. This adaptive approach demonstrates ***CODED.***'s commitment to providing an equitable learning environment for all students and faculty members in CS and CTIS labs.

6.3 Public Welfare Considerations

By facilitating a better learning environment, ***CODED.*** contributes to skill development among students. This is crucial for preparing a skilled workforce, which is a critical component of public welfare, as it directly impacts employability and economic growth [19].

Besides employability and economic growth, ***CODED.*** emphasizes a new trend in computer science: large language models (LLMs) [20]. According to initial market research, it is concluded that AI in education is emerging, and ***CODED.*** is hereby the key pioneer in this area, accounting for its wide range of features that have never been done by any startup before. It is an indicator of ***CODED.'s*** public welfare impact in the area of education, favoring Artificial Intelligence.

6.4 Global Considerations

CODED. is precisely designed to be adaptable for a wide range of universities worldwide, particularly those following the US or European education systems. It aims to be a globally-oriented product, ensuring its performance is not impeded by varying academic structures and requirements. The application's design is highly customizable, allowing educators to tailor features to their specific needs, enabling immediate integration into their computer science labs.

Recognizing the prevalence of English as a medium of instruction in many universities, especially in the United States, ***CODED.*** primarily uses English. This choice supports its global applicability. However, to further extend its global reach, ***CODED.*** is also available in Turkish, Spanish, and French upon request. Future updates aim to include additional languages and localization features, catering to a broader international audience and addressing cultural nuances more effectively. Enhancements in ***CODED.*** also focus on supporting diverse curricula, accommodating different grading systems, and aligning with various academic calendars. Moreover, AI components like chatbots are developed with cultural sensitivity in mind, ensuring appropriateness and inclusivity in interactions [21]. Accessibility features are also a priority, adhering to international standards to serve a broader range of users, including those with disabilities [22].

To ensure ***CODED.*** meets the diverse needs of its global user base, and ongoing collaborations with international educational institutions are established for continuous feedback and development. This approach helps ***CODED.*** stay responsive to changing educational trends and regulations across different regions. Additionally, the platform is optimized for varied network environments, ensuring efficient performance worldwide, and commits to sustainable, eco-friendly practices in its development and deployment.

6.5 Cultural Considerations

One of the prior goals for ***CODED.*** is to pay attention to the cultural or other aspects of differences between users, favoring inclusiveness and diversity in the environment. The content, including examples, scenarios, and interactions within the platform, especially in the chatbot, would need to be culturally sensitive. This involves avoiding stereotypes, using inclusive language, and being mindful of cultural nuances and differences.

As an education application, ***CODED.*** should also consider the fact that cultural differences may influence learning styles [23]. ***CODED.***'s design and teaching methodologies should accommodate various learning preferences and educational backgrounds, ensuring that all students can benefit equally from the resources.

In terms of data privacy and/or other regulations that matter for the ***CODED.***, it should be complementary with the cultural background and the influence of that culture on the regulations to obey terms and regulations within the country/state.

6.6 Social Considerations

CODED. capacity to effect positive social change, particularly in computer science education, is substantial. Efforts could be intensified to reach not only underprivileged groups but also those who are typically underrepresented in STEM fields, such as women and minorities. This commitment to diversity and inclusion would significantly contribute to leveling the educational playing field and promoting social equity.

In addition to broadening access, ***CODED.*** is instrumental in identifying and supporting students who face challenges in lab sessions. The platform could be enhanced with sophisticated analytics tools, enabling instructors and teaching assistants to gain deeper insights into individual student performance. These tools can provide detailed, actionable data, allowing for timely and personalized intervention.

Furthermore, engaging students in feedback and development processes can empower them and provide valuable insights for the continual improvement of ***CODED.*** This inclusive approach ensures that the platform evolves to meet the diverse needs and preferences of its user base, thus enhancing its social impact and relevance in the ever-changing landscape of computer science education.

6.7 Environmental Considerations

CODED.'s transition of education to a digital platform plays a pivotal role in reducing the environmental footprint of CS and CTIS labs in a limited perception. By digitizing many traditional paper-based processes, especially in areas like grading and assignment submission, ***CODED.*** significantly diminishes paper usage, contributing to a reduction in deforestation and waste.

In addition to minimizing paper waste, ***CODED.'***s compatibility with remote learning setups presents an opportunity for further environmental benefits. Facilitating online education potentially reduces the need for students and academic staff to commute, thereby lowering carbon emissions associated with transportation [24].

However, an area of environmental concern is ***CODED.*** is the use of the OpenAI API, which, despite its advanced capabilities, has a considerable carbon footprint due to high computing costs. The reliance on intensive GPU usage for this chatbot inevitably leads to increased electricity consumption and, consequently, higher carbon emissions [25]. While this is a current limitation in balancing the project's environmental goals with its technical needs, it is acknowledged as an area for future improvement.

To address this challenge and align more closely with sustainable practices, ***CODED.*** could explore alternative, more energy-efficient AI models or invest in offsetting its carbon footprint through renewable energy usage or other sustainability initiatives. Moreover, engaging with ongoing research in reducing the environmental impact of chatbots and AI systems is crucial. As technology evolves, there is optimism that more sustainable solutions will emerge, allowing ***CODED.*** to maintain its technological edge while improving its environmental sustainability.

6.8 Economic Considerations

The economic framework of ***CODED.*** encompasses several key aspects. Firstly, efficient budget management is crucial to ensure that funds are judiciously allocated for development, maintenance, and updates. This involves a thorough cost-benefit analysis to justify the investment by weighing the platform's benefits against its operational expenses.

Resource optimization, particularly in human resource deployment, is essential to enhance productivity while minimizing costs. Exploring various revenue models, such as subscriptions or educational grants, can provide sustainable financial support for ***CODED.***

Moreover, maintaining economic accessibility is vital, making ***CODED.*** affordable or potentially free for educational institutions, especially those with constrained budgets, like the universities of the least developed countries. This approach not only broadens the platform's reach but also reinforces its commitment to educational equity. Also, the long-term economic impact of ***CODED.*** should be considered, particularly its role in enhancing skills and employability among students, thereby contributing positively to broader economic development.

6.9 Table of the Aforementioned Considerations

Consideration	**Effect Degree [0 - 10]**	**Comment**
Data Privacy	10	*The most important factor.*
Accessibility & Equality	8	*One of the most important ones.*
Public Welfare	4	*Less important.*

Global	8	*Being a global key software is very important for **CODED.***
Cultural	6	*Somewhat important.*
Social	6	*Somewhat important.*
Environmental	3	*Due to limitations, it has very little effect for now.*
Economic	7	*As a non-invested startup, the economic situation is important for **CODED.***

Table 1: Considerations of Engineering Factors Summary Table

7 Teamwork Details

7.1 Ensuring Proper Teamwork

Since our project consists of many components that require hard work and detailed analysis, ensuring proper teamwork is a key concept in order for us to finish this project on time. We realized this earlier in the project, and that is why we have been using Jira from the very start of the project. Jira is a leading agile project management tool widely used by teams for planning, tracking, releasing, and supporting high-quality software [26]. We use its timeline feature to add our deadlines, such as reports and meetings, in order to help us visualize how much time we have left for various tasks. We also use its Kanban board feature, which is an agile project management tool that enhances work visualization, limits ongoing work, and boosts efficiency, benefiting teams in organizing and completing their tasks effectively [27]. We have columns on this board such as “To-do,” “In Progress,” “Finished,” “Being Tested,” and “Done,” which represent the different steps of our tasks. Each team member is responsible for updating the progress of their tasks on the Kanban board.

There are also other software that we use to ensure proper teamwork other than Jira. We are using WhatsApp and Zoom for communication, GitHub for version control, Google Drive for working on reports and documents, keeping logs, and Lucidchart and diagrams.net software to work on diagrams.

Other than using online software for teamwork, we also meet up 2 or 3 times a week to keep each other informed about our progress and the challenges that we are facing. We also keep logs of our meetings. It is also worth mentioning that we divide the workload evenly so that no member is doing significantly larger tasks than others. We try to ensure proper teamwork by doing all the steps mentioned above [14].

7.2 Contributing and Functioning Effectively on the Team

Every team member is given a specific task to focus on in order to function effectively and finish our application on time. When we are writing a report, we divide it equally so that each member has 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 is mainly developing the chatbot component, and Beyza and Zeynep Hanife are working on the Test Case Runner component. Selin is dealing with the Code Analysis Component and Zeynep Selcen is 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 Helping Creating a Collaborative and Inclusive Environment

Everybody on the team is the leader of one of the main components, however, this does not stop us from helping each other with the struggles that they had with their code. When each team member is faced with a problem while writing code, they open a GitHub issue. That way, every one of us would be aware of the situation, and we would look at and analyze the code to see how we could solve the issue. We also use WhatsApp and Zoom to inform each other about the problems that we have faced and how we are planning to solve them in order to keep each other informed. We are aware that this is a group project, and even though we split the tasks among ourselves, we are all responsible for creating a fully developed application.

7.4 Taking the Lead Role and Sharing Leadership on the Team

Since we had a project with many different features, everyone on the team took a leadership role in a different part of the project. In the Analysis and Requirements report, we created work packages that consisted of various tasks and assigned a leader to each work package. While implementing the code, we followed these as a blueprint, but we also added assigned leaders to more specific tasks. Each member of the project was familiar with various technologies, so we chose the leadership positions based on our previous internship and project experiences.

For example, Emre is the leader of the Chatbot component since he was experienced in developing chatbots because of his past internships. Beyza is the leader of the UI and Zeynep Hanife is the leader of the Test Case Runner component since they both had familiarity in using these technologies because of previous projects and internships. Selin takes the lead role in the Code Analysis Component since she has knowledge about code analysis from her research with Assistant Prof. Eray Tüzün. Zeynep Selcen takes the leadership role in implementing the general backend and also the Plagiarism Checker component since she was familiar with SpringBoot because of her past internships.

8 Glossary

- **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 [28].
    - **Acronym: RDS**
- **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.[28].
    - **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 [28].
    - **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 [28].
    - **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 [28].
    - **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 [28].
    - **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 [28].
    - **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 [28].

  - **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 [28].
  - **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 [28].
  - **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 [28].
  - **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 [28].
    - **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 [28].
    - **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 [28].
    - **Acronym**: 2FA

9 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] Amazon Web Services, Inc., "Amazon RDS," 2023. [Online]. Available: https://aws.amazon.com/tr/rds/. [Accessed: Mar. 12, 2024].

[8]PostgreSQL Global Development Group, "About PostgreSQL," 2023. [Online]. Available: https://www.postgresql.org/about/. [Accessed: Mar. 12, 2024].

[9]] Amazon Web Services, Inc., "Amazon S3," 2023. [Online]. Available: https://aws.amazon.com/tr/pm/serv-s3/. [Accessed: Mar. 12, 2024].

[10] Progress Software Corporation, "Use AES-256 Encryption to Secure Your Data," 2023. [Online]. Available: https://www.progress.com/blogs/use-aes-256-encryption-secure-data. [Accessed: Mar. 12, 2024].

[11] Cloudflare, Inc., "What is HTTPS," 2023. [Online]. Available: https://www.cloudflare.com/learning/ssl/what-is-https/. [Accessed: Mar. 12, 2024].

[12] Microsoft, "Presidio," 2023. [Online]. Available: https://microsoft.github.io/presidio/. [Accessed: Mar. 12, 2024].

[13] "Security threats and security testing for chatbots," Chatbots Life, 2023. [Online]. Available: https://chatbotslife.com/security-threats-and-security-testing-for-chatbots-325d704da9af. [Accessed: Mar. 12, 2024].

[14] Z. H. Akgül, B. Çağlar, S. B. Gündoğar, E. Karataş, Z. S. Öztunç, "Analysis and Requirements Report," Bilkent University, CS 491 Senior Design Project, Project ID: T2322, December 8, 2023.

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

[16] 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].

[17] 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].

[18] 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].

[19] 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.

[20] 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.

[21] C. Kelsey, "AI Chatbot to Increase Cultural Relevancy of STEM," Indiana University News, Indiana University, Oct. 17, 2023. [Online]. Available: https://news.iu.edu/live/news/31938-ai-chatbot-to-increase-cultural-relevancy-of-stem. [Accessed: Dec. 4, 2023].

[22] S. Federici, M. L. de Filippis, M. L. Mele, S. Borsci, M. Bracalenti, G. Gaudino, A. Cocco, M. Amendola, and E. Simonetti, "Inside Pandora's Box: A Systematic Review of the Assessment of the Perceived Quality of Chatbots for People with Disabilities or Special Needs," Disability and Rehabilitation-Assistive Technology, vol. 15, no. 7, pp. 832-837, 2020. DOI: 10.1080/17483107.2020.1775313.

[23] V. Prabhakaran, R. Qadri, and B. Hutchinson, "Cultural Competencies in Artificial Intelligence," presented at the First Workshop on Cultures in AI/AI in Culture (non-archival), NeurIPS 2022. [Online]. Available: [https://ai-cultures.github.io/papers/Cultural_Competencies_in_Artificial_Intelligence__NeurIPS_2022_Culture_AI_Workshop.pdf. [Accessed: Dec. 4, 2023].

[24] 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.

[25]. "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].

[26] Atlassian, "What is Jira Software?," Atlassian.com, [Online]. Available: https://www.atlassian.com/software/jira/guides/getting-started/introduction#what-is-jira-software. [Accessed: Nov. 24, 2023].

[27] Atlassian, "What is a Kanban board," Atlassian.com, [Online]. Available: https://www.atlassian.com/agile/kanban/boards. [Accessed: Nov. 24, 2023].

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