Bilkent University

Department of Computer Engineering

CS 413

Software Engineering Project Management

2023-24 Spring Semester

Team Project Charter

Project Name: CODED.

Group Name: Blank Space

Project Team Members:

Zeynep Hanife Akgül	(22003356)
Beyza Çağlar	(22003394)
Alper Göçmen	(22002948)
Deniz Tuna Onguner	(22001788)
İlayda Zehra Yılmaz	(22001769)

Instructor: Onur Karabulut

7 April 2024

Executive Summary

CODED. is a website application that assists students, tutors, and instructors in CS (Computer Engineering) and CTIS (Information Systems and Technologies) labs at Bilkent University throughout the entire lab process. The application has four features: a chatbot, a code analyzer, a test case runner, and a grading bot. The chatbot assistant answers students' questions related to their code without revealing the "exact" answer but giving clues as to how to reach a possible solution in order to encourage the student to learn. The similarity check feature does a similarity check among codes on Github/sections/all past codes/classes, based on the instructor's preferred settings. The clean code evaluator checks students' code quality, gives them feedback on the quality of their code, and helps them in the debugging process. Finally, the test case runner feature does automated tests on student codes to provide an auto-grading system. Overall, CODED. tutors lab students while minimizing the effort of lab assistants and instructors [1].

Goal and Objectives

Bilkent University is currently facing a problem of finding lab tutors for the CS and CTIS labs. Most students have questions about the lab assignments or have trouble debugging a specific part of their code during their lab periods, but only 2-3 teaching assistants (TAs) and 1-2 tutors can be present during each lab session. The CS courses do not have 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 students to volunteer to fill the tutor vacancies for TA roles as the CTIS department does not have a graduate program. Furthermore, most tutors, who are volunteering students who display proficiency in the course's coding language, cannot stay throughout all of the lab hours due to their schedules. Even when they are able to do so, the student-to-tutor ratio is around 1/30 [2]. Henceforth, tutors do not have the time to pay attention to each student. Furthermore, 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. Therefore, there is an urgent lab tutor needed in the CS and CTIS departments. Nevertheless, 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.”[3]. Followingly, the completed lab assignments are then uploaded to the application, where these assignments are tested for plagiarism based on the criteria that the instructor selects. The instructor can compare the code to a specific section, the whole class, code on Github, past codes, or check for AI (Artificial Intelligence) generated code similarity. 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 [1].

Outcomes and Deliverables

The outcomes and deliverables of *CODED.* are chosen based on the best practices that will represent the stakeholders’ requirements and needs from the application. The primary major outcome to arise from the project is a web application that will ease the workload of instructors, TAs and tutors while maintaining the quality of education for the students. By using *CODED.*, the students can find answers to their basic code-related questions without waiting in line for a tutor. The academic personnel can handle the cases where the application may not be adequate, such as problems with IDE setup. There should be a prototype to test the application in live lab sessions and regular deliverables to ensure alignment with stakeholder requirements.

The planned deliverables and their respective delivery dates can be seen below with respect to the starting date T_0:

Deliverable	Description	Delivery Date	
Project Charter and Sprint Backlog	Outline of the project scope, objectives, stakeholders, and requirements. Plan the first sprint.	T_0 + 1 week	
Project Management Plan (PMP)	Defines the project scope, objectives, goals, budget, timeline, and resource management in detail.	T_0 + 2 weeks	
Requirements Analysis Document	Documents the stakeholder and software requirements in functional and non-functional aspects.	T_0 + 2-3 weeks	
System Design Document	Demonstrates the system architecture and components.	T_0 + 3 weeks	End of Sprint 1
Mockup	Initial User Interface (UI) design and presentation to the stakeholders.	T_0 + 3-4 weeks	
Codebase and Documentation	Initial source code and repository structure.	T_0 + 4 weeks	
Functional Prototype	The first functioning representation of the software available for basic	T_0 + 4-5 weeks	

	testing.		
Minimum Viable Product (MVP)	The simplest version of the end product that provides value to the users.	T_0 + 6 weeks	End of Sprint 2
Test Plans	Outline of the test strategy for the future versions of the application.	T_0 + 7 weeks	
Alpha Version	The software version available for internal testing.	T_0 + 8-9 weeks	
Alpha Version Testing and Code Quality Report	Report on the results of the alpha version tests.	T_0 + 9 weeks	End of Sprint 3
Beta Version	The software version available for testing with the stakeholders and a small group of users.	T_0 + 12 weeks	End of Sprint 4
Beta Version Testing and Code Quality Report	Report on the results of the beta version tests.	T_0 + 15 weeks	End of Sprint 5
User Acceptance Testing (UAT)	End-user tests to determine whether the application is ready to be deployed.	T_0 + 17-18 weeks	End of Sprint 6
Final Release	The end product, that is ready to be deployed.	T_0 + 21 weeks	End of Sprint 7

User Manual	Documentation for the end users, explaining how to use the application.	T_0 + 22 weeks	
Final Budget Report	Finalization of expenditures and budget allocations.	T_0 + 23-24 weeks	
Project Closure Report	Summary of the project, outcomes, lessons learned, and future plans.	T_0 + 24 weeks	End of Sprint 8

Definition of Scope

1. Technology Stack: The project will leverage a diverse technology stack including the React.JS framework for frontend development, the Spring Boot framework and Python for backend development, and PostgreSQL hosted by Amazon RDS server for database management. The application will be hosted on GitHub Pages.

2. Code Quality and Testing: To ensure code quality and adherence to conventions, SonarQube will be used. Plagiarism checks will be conducted using MOSS. For testing, pytest will be used for Python codes and either UnityTest or CUnit for C codes.

3. Chatbot Development: The chatbot will be developed using the OpenAI model with a fine-tuned version developed by the CODED. team. The chatbot will provide instant, personalized feedback to enhance the learning process.

4. Data Protection: The system will store only necessary personal details and will comply with data protection regulations such as the General Data Protection Regulation (GDPR). The application will not directly collect user data from the chatbot, but will retain user data from code analytics and personal information such as email.

5. Application Focus: The application is designed to improve the student lab experience by helping them write code. It aims to streamline the code evaluation process and facilitate efficient student-tutor interactions. The application will address challenges faced by Bilkent University in providing sufficient lab tutoring for CS and CTIS courses.

6. Key Features: Key features of the application include a chatbot for immediate answers to student questions, automated code quality checks, plagiarism detection, and an advanced grading

tool. The application will also reduce the workload for tutors and TAs by automating critical aspects of the lab process.

7. Supported Courses: The project will support C and Python coding languages for CTIS 150, CTIS 151, CTIS 152, CS115, and CS125 courses.

8. Release Plan: The goal of the beta release is to test the application for performance, efficiency, customer satisfaction, and bug discovery.

9. System Architecture: The system architecture includes client interfaces tailored for different users, logic layers handling functional requirements, external APIs for additional functionalities, and AWS services for data management. The application emphasizes usability, performance, reliability, marketability, security, scalability, maintainability, flexibility, modularity, and aesthetics.

10. Economic Considerations: Due to economic restrictions, the project will use as many open-source frameworks as possible.

Budget

The budget for PROJECT is parted into different categories : engineering, business, hardware, software, office, services, and other miscellaneous costs. The profit is selected to be 25% as similar to industry standards. The Project timeline is 16 weeks, and the budget is calculated for 6 months.

Engineering (Person-Months) Cost Details:

Staff Security Engineer: 6 person-months x $10,000 = $ 60,000

2 Senior Security Engineer: 2 x 6 person-months x $9,500 = $114,000

2 Junior Security Engineer 2 x 6 person-months x $8,000 = $96,000

Staff Software Engineer: 6 person-months x 9,000 = $54,000

2 Senior Software Engineer: 2 x 6 person-months x $8,000 = $96,000

Senior Software Architect: 6 person-months x $10,000 = $60,000

2 Junior Software Architect: 2 x 6 person-months x $9,000 = $108,000

Staff Test Engineer: 6 person-months x $7,000 = $42,000

Test and QA Lead: 6 person-months x $8,000 = $48,000

Senior Automation Engineer: 6 person-months x $8,500 = $51,000

Senior QA Engineer: 6 person-months x $7,500 = $45,000

Junior QA Engineer: 6 person-months x $7,000 = $42,000

Staff ML Engineer: 6 person-months x $10,000 = $60,000

Lead Data Scientist: 6 person-months x $10,000 = $60,000

Senior ML Engineer: 6 person-months x $9,500 = $57,000

Senior Data Engineer: 6 person-months x $9,500 = $57,000

Senior NLP Engineer: 6 person-months x $9,500 = $57,000

Junior ML Engineer: 6 person-months x $8,000 = $48,000

UI/UX Manager: 6 person-months x $7,500 = $45,000

Lead UI Designer: 6 person-months x $7,500 = $45,000

UX Designer: 6 person-months x $6,500 = $39,000

UX Researcher: 6 person-months x $6,500 = $39,000

3 Candidate Engineer: 3 x 6 person-months x $1,000 = 18,000

Project Manager: 6 person-months x $8,500 = $51,000

Business (Person-Months) Cost Details:

Executive Manager - CEO: 6 person-months x $20,000 = $120,000

Finance Manager: 6 person-months x $8,000 = $48,000

Product Manager: 6 person-months x $8,000 = $48,000

Technology Manager: 6 person-months x $8,000 = $48,000

Marketing Manager: 6 person-months x $8,000 = $48,000

Legal Affairs Manager: 6 person-months x $8,000 = $48,000

Hardware Cost Details:

Development Hardware (Laptops, Monitors, etc.): $20,000

Software Cost Details:

Development IDEs, environments, licenses: $8,000

Management Tools: $1,000

AWS RDS (Relational Database Service) Subscription: $3000/month x 6 = $18,000

AWS EC2 (Elastic Compute Cloud) Deployment Cost: $1000/month x 6 = $6,000

OpenAI API Usage Cost: $1000/month x 6 = $6,000

Services Cost Details:

Marketing costs: $3,000

Customer support: $3,000

Other Cost Details:

Employee Training: $1,500

Contingency Fund: $6,000

Total Budget Details:

Engineering Person-Months Cost: $1,392,000

Business Person-Months Cost: $360,000

Hardware Cost: $20,000

Software Cost: $39,000

Services Cost: $6,000

Other Cost: $7,500

Overall Cost: $1,824,500

Profit: 25% of 1,824,500 = $456,125

Total Budget: $2,280,625

Major Milestones

Major milestones are the important steps in the project's process and helps us break the project into more manageable, smaller parts. This is beneficial for having clear goals and expectations as well as more manageable activities and processes in our project's development. Using major milestones, we are able to keep track of our results, the risks the project possess, our documentation and work more efficiently.

(T_0 is the starting date)

1. **Project Initiation**

   Start Date: T_0

   End Date: T_0 + 1 week

   - The project goals and objectives are defined.
   - Stakeholders are identified.
   - Project Charter that includes project risks, constraints, scope, budget, milestones and team contributions is delivered.
   - Project schedule is established.
   - Sprint Backlogs are created.

2. **Requirements**

   Start Date: T_0 + 1 week

   End Date: T_0 + 2-3 weeks

   - Requirements Analysis Document that documents the non-functional and functional requirements of software and stakeholders is finalized.
   - A Project Management Plan (PMP) that includes the project's objectives, goals, budget, resource, and time management is created.

3. **Design**

   Start Date: T_0 + 2-3 weeks

   End Date: T_0 + 3-4 weeks

   - System Design Document that includes system design, components and architecture is delivered.
   - Mockup that includes Initial User Interface (UI) should be finalized and presented to the stakeholders.

4. **Development**

   Start Date: T_0 + 3-4 weeks

   End Date: T_0 + 6 weeks

   - Codebase and its documentation should be handled.
   - Functional Prototype that is the first functioning representation of the software should be finished.
   - Minimum Viable Product (MVP) should be created.

5. **Quality Assurance and Testing**

Start Date: T_0 + 6 weeks

End Date: T_0 + 17-18 weeks

- Testing Plans should be done.
- System should be tested in multiple versions and test cases.
- Test cases, bug reports, bug fixes and quality assurance checks should be done.
- Code Quality Reports should be created and finalized.
- User Acceptance Testing (UAT) should be handled.

6. **Deployment**

Start Date: T_0 + 17-18 weeks

End Date: T_0 + 22 weeks

- System should be deployed.
- User Manual that describes the usage of the application for the users should be created and finalized.

7. **Invoice & Payment**

Start Date: T_0 + 22 weeks

End Date: T_0 + 23-24 weeks

- Invoicing is created and payment processes should be handled.
- Final Budget Report that includes finalization of expenditure and final budget allocations should be finalized.

8. **Closure**

Start Date: T_0 + 23-24 weeks

End Date: T_0 + 24 weeks

- Project Closure Report which includes the project summary, the lessons learned, outcomes and future plans should be created and finalized.

Assumptions and Constraints

Assumptions

1. **Availability of Stable Internet Connection:** Assuming that there will be stable internet connectivity provided by the environment, i.e. labs, during lab sessions for students to access the services of the web app.
2. **Availability of Requisite Hardware:** Assuming that students in labs will have access to devices/computers which are capable of running a web browser to utilize the web app.
3. **Compatibility with Primary Web Browsers:** Assuming that the web app will be compatible with commonly and widely used web browsers such as Safari, Chrome, and Firefox.
4. **Active User Engagement:** Assuming that students in labs will actively engage, or are compelled to engage, the web app during lab sessions.
5. **Exclusive Use of the Web App:** Assuming that students in labs will solely use, or be only allowed to use, the web app and no other tools or services in order to enhance consistency and control over lab assignments for instructors.
6. **Systematic User Training:** Assuming that necessary training and support will be provided to students and instructors, especially to the ones in a major other than engineering, for them to use the web app as expected/planned.

Constraints

1. **Time Constraints:** The web app needs to be fully developed and be ready for deployment before the starting of the next semester to align with the academic calendar and to be available for use throughout the whole semester.
2. **Performance Constraints:** The web app needs to be able to handle a varying number of users/students during lab sessions without experiencing performance issues. It is almost certain that there will be more than a lab session at a time divided into several sections.
3. **Instructor Discretion:** Instructors should have the ability of disabling any of the provided services of the web app (i.e. chatbot, plagiarism check, code quality measurement, and pass-failure of test cases) as they see fit.
4. **Privacy Constraints:** Considering the system will store both personal and academic data, the web app needs to be complying with data privacy regulations such as GDPR.

Initial Risks

Threat - Dependency on External Services:

- **Cause (Reason)**: The project's infrastructure is built on AWS, making it heavily reliant on this external service provider. This dependency is necessary to leverage the scalability, flexibility, and broad service options that AWS provides.
- **Risk (Uncertainty)**: AWS, like any other cloud service provider, can experience downtime due to technical issues, maintenance, or even cyber attacks. Additionally, AWS may change its service policies, pricing, or even discontinue certain services.
- **Effect (Consequence)**: Any disruption in AWS services or changes in its policies can lead to downtime in your project, affecting its availability and reliability. This could result in delays in project timelines, dissatisfaction among users, and potential cost overruns if AWS increases its prices or if you need to switch to another service provider.

Opportunity - Economic Constraints:

- **Cause (Reason)**: The project uses open-source frameworks and models, specifically the OpenAI model family, to keep costs low. These models are complex and computationally expensive, but they are necessary for the project's data processing needs.
- **Risk (Uncertainty)**: There's a risk of running into resource limitations due to the complexity and computational expense of these models. These limitations could be in terms of processing power, memory, or storage.
- **Effect (Consequence)**: If resource limitations are encountered, it could slow down the project's progress, affect its performance, and lead to longer processing times. However, this also presents an opportunity to optimize resource usage and explore more efficient models. This could lead to cost savings, improved performance, and even open up possibilities for scaling up the project in the future.

Threat - Cost Overruns:

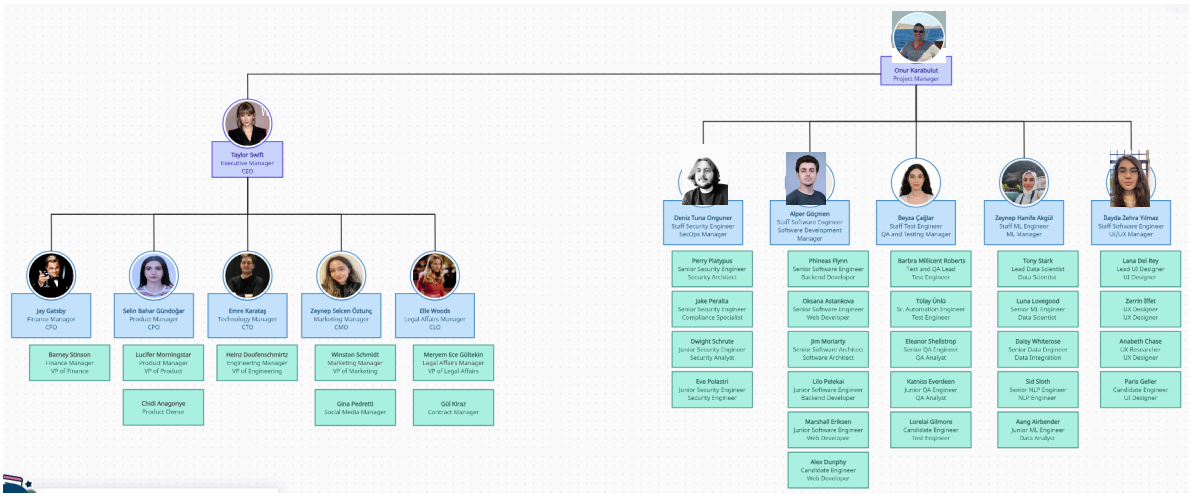
- **Cause (Reason)**: The project uses AWS services, which, while scalable, can lead to costs that exceed the estimated projections if the usage is higher than anticipated.

- **Risk (Uncertainty)**: There's a risk of cost overruns if the project's AWS usage exceeds the estimated projections. This could be due to higher than expected traffic, data storage needs, or computational requirements.
- **Effect (Consequence)**: Cost overruns could strain the project's budget and potentially require cuts in other areas or additional funding. This could also lead to delays if the project needs to be paused to secure additional funding.

Opportunity - Coding Language Constraints:

- **Cause (Reason)**: The project supports C and Python coding languages to cater to certain courses. This decision was made based on the popularity and versatility of these languages in the academic and professional world.
- **Risk (Uncertainty)**: There's a risk of compatibility issues with different APIs or frameworks due to the specific coding languages chosen. This could limit the functionality or performance of your application.
- **Effect (Consequence)**: If compatibility issues arise, it could affect the functionality of your application and lead to longer development times. However, this also presents an opportunity to explore other coding languages and broaden the application's scope. This could lead to improved compatibility with APIs and frameworks, enhanced functionality, and a broader user base for your application.

Project Organization Chart



Approval and Authorization

Prepared by the Project Manager:

Onur
Karabulut

Sign

Name and Last Name: Onur KARABULUT

Date: 07.04.2024

Approved by the Project Sponsor:

Eray Tüzün

Sign

Name and Last Name: Eray Tüzün

Date: 07.04.2024

Contributions

Zeynep Hanife Akgül: Goals and Objectives, Budget.

Beyza Çağlar: Outcomes and Deliverables, Project Organization Chart.

Alper Göçmen: Itinial Risks.

Deniz Tuna Onguner: Assumptions and Constraints.

İlayda Zehra Yılmaz: Milestones.

References

[1] Z. H. Akgül, B. Çağlar, S. B. Gündoğar, Z. Öztunç, and E. Karataş, "Detailed Design Document", Coded, 15 March 2024.

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

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