

**HACETTEPE UNIVERSITY**

**COMPUTER ENGINEERING DEPARTMENT**

**UNDERGRADUATE PROJECT FINAL REPORT**

|  |  |
| --- | --- |
| **Project Name** | **Report Date** |
| On-Site Coding | 13.01.2020 |

|  |  |
| --- | --- |
| **Student Number(s)** | **Student Name(s)** |
| 21627711  21693239  21426682 | Halil Etka Tutkun  Uğur Küçükterzi  Said Furkan Ayvaz |
| **Supervisor(s)** | **Company Representative(s)** |
| Gönenç ERCAN |  |

|  |  |
| --- | --- |
| **Project Coordinator** | **Report Approval** |
| Date: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | ☐ Yes ☐ No  If no, rational of rejection: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |

|  |
| --- |
| **Project Video Youtube Link** |
| <https://www.youtube.com/watch?v=tAPY07f0N1E> |

# **TECHNICAL RESULTS**

## **ABSTRACT**

|  |
| --- |
| Developing an environment which automates the evaluation of the assigned coding challenges. |
| **Keywords:** digital-learning, online-editor, online-compiler, automated-evaluation, coding-challenges |

## **INTRODUCTION**

|  |
| --- |
| There are many coding courses and training institutes out there. Mostly they use manual processes to evaluate the coding challenges they assign to students. Addition to that, students have to go through a boring process to set-up the development environment, even for smallest challenges, which is really discouraging for newly starting students.  Intention of our platform is to make the evaluation process faster by making it automated to some degree. Also, to provide an online environment for the students that takes near zero effort to set-up to start coding. |

## **BACKGROUND**

|  |
| --- |
| In our project we used a few popular technologies such as,  ASP.NET Core MVC: It is a rich framework for building web apps and APIs using the Model-View-Controller design pattern.  <https://docs.microsoft.com/en-us/aspnet/core/mvc/overview?view=aspnetcore-3.1>  ASP.NET Core SignalR: It is an open-source library that simplifies adding real-time web functionality to apps. Real-time web functionality enables server-side code to push content to clients instantly.  <https://docs.microsoft.com/en-us/aspnet/core/signalr/introduction?view=aspnetcore-3.1>  MongoDB: It is an open-source general purpose, document-based, distributed database built for modern application developers and for the cloud era.  <https://www.mongodb.com/>  Docker: It is an open-source tool designed to make it easier to create, deploy, and run applications by using containers.  <https://www.docker.com/> |

## **RELATED WORK**

|  |
| --- |
| There are similar systems out there that uses different approaches to solve this problem. The security and resource exhaustion are still big issues, which is a result of running user inputted code. Therefore, all these systems are closed source and there is minimum information out there related to these systems. But based on the usefulness and user experience here are some similar platforms: <https://repl.it/>, <https://autogradr.app/> , <https://www.codiva.io/> |

## **METHOD**

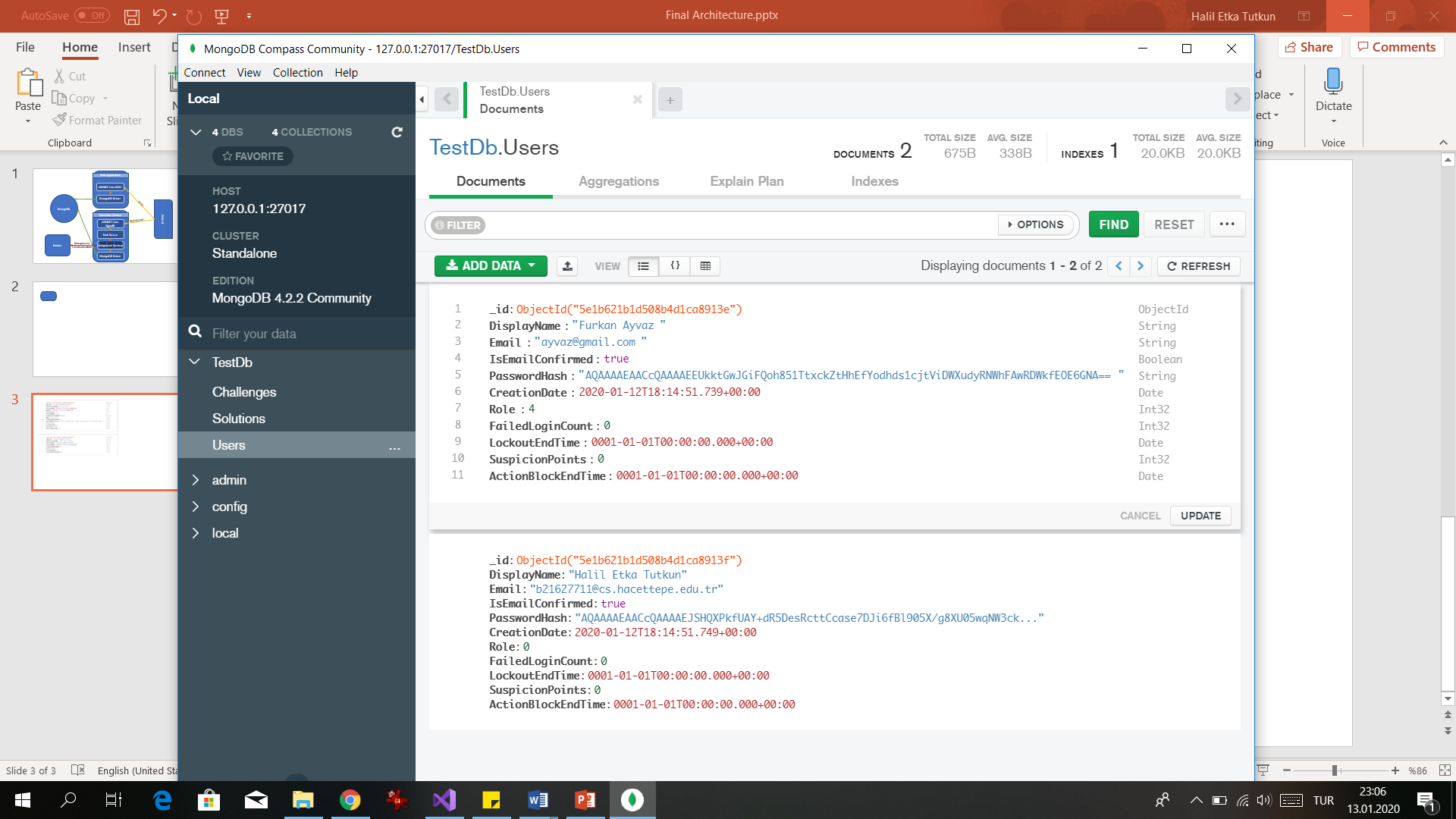
|  |
| --- |
| We have studied other similar systems and tried to come up with and idea on how we can implement our system. Due to partitioning requirements among group members, architecture of the software plays a big role in this project. It had to allow both independent development and high performance.  We’ve assigned different parts of the projects to different members which allowed us focus to these areas independently. After each member completed their parts, we integrated all of them and run some integration tests. According to these test results we applied some improvements and added new features. Throughout this process we had to keep a good communication to be able to develop parts that will integrate easily. |

## **TECHNICAL DESIGN AND CONFIGURATION**

|  |
| --- |
| Our system consists of five major components. In this section we will describe them briefly and will give more technical details in the upcoming section.   1. Web UI: This is the component which lets users to interact with the system over a browser. 2. Web Application: This is the component which performs most of the data manipulation operations and presents this data to user as a web page. This component also handles Authentication and Authorization operations. 3. Execution Service: This is the component which performs the compilation & execution related operations with the help of some sub-components. 4. Docker: This is the component where we compile & run our user inputted code. 5. MongoDB: This is the component where we persist our data.   To be able the run this project on given environment below should be configured:   * 1. .NET Core Runtime should be installed.   2. MongoDB services should be installed, and it should be hosted.   3. Docker should be installed, and virtualization should be enabled for the environment.   4. One or separate web servers should be configured to host both Web Application and Execution Service (should be configured to allow WebSocket connection).   5. Execution Service should be configured to receive CORS from the Web Application’s domain.   6. A SSL certificate should be provided for application to use (WebSocket only works on HTTPS connection).   7. Both servers should be configured to use a shared secret key for signing Authentication Tokens. Also, they should be configured with the Database connection string.   Below is the overview of the core functionality of the program. There might be other functionalities as well, but they will be explained in the upcoming sections.   * 1. Educator creates an account.   2. Educator creates a public or private challenge.   3. If challenge is private educator shares the challenge key with participants.   4. Participant creates an account.   5. Participant joins to a challenge through a shared key or by discovering public challenges.   6. Participant tries to solve the challenge and tests his/her solution.   7. Educator checks the score of each executed participant solution.   8. Educator confirms the final grading.   Below are some functional requirements the system needs to fulfill. **Educator Challenge Interaction**  * + 1. Can choose to create a public or private challenge.     2. Should choose a programming language the challenge will be solved with.     3. Can restrict access to private challenges by an email domain that belongs to an institution.     4. Should create test cases with execution command and expected output.     5. Test cases can be visible or hidden for students.     6. Can create required input files.     7. Can create a starting template.     8. Can view the participants and score of their solutions.  **Participant Challenge Interaction**  * + 1. Can join to a challenge via the shared key.     2. If there is a domain restriction for a private challenge and participant’s mail address doesn’t match with the required domain name, he/she should get an error.     3. Can discover public challenges to join.     4. If challenge is already due, he/she will get an error.     5. Can solve the challenge in the environment provided by the system.     6. Should be able to save the solution to the system.     7. Can test the solution against the test cases created by the educator.     8. Latest test results should be saved automatically as the score of the solution. |

## **PROJECT IMPLEMENTATION**

|  |
| --- |
| In this section we will give technical details about the components of our software that we introduced in the previous section.   1. Web UI: We implemented this component using HTML, CSS and JavaScript. We used AJAX method in some places to make the system more responsive. Because AJAX method requires less data transfer between server and UI. This component uses two different protocols to communicate with two different services. HTTP for Web Application and WebSocket for Execution Service. WebSocket protocol is implemented using JavaScript library of SignalR. 2. Web Application: We implemented this component using ASP.NET Core MVC. This component communicates with the Web UI using HTTP protocol. It manipulates the data using the MongoDB driver library for C#. This component handles Authentication and Authorization by providing a JWT token to user after validating his/her credentials. This token is stored in Web UI’s cookie and sent with every request to both Web Application and Execution Service. 3. Execution Service: We implemented this component using ASP.NET Core SignalR which provides structured methods for handling WebSocket connections. Because the user makes request to this server from a different origin, we had to configure CORS policies. After user makes a request to this server it puts this request to an internal thread safe Task Queue which is enqueued by multiple background services. These services take the required action for each request. For example when a user connects a “Create Environment” request is processed by copying files from database to user environment which will be used for compilation & execution. When a user sends run request a background service compiles and runs this environment inside a Docker container. By using Inter-process Communication it gets the console output from the container and sends the results using WebSocket. When user disconnects this environment is deleted again by a background service. 4. Docker: We chose this containerization technology to provide an execution environment because it’s both secure and fast. 5. MongoDB: We chose this database technology because it was the best choice for storing semi-structured data. It’s also fast and easily scalable. Below is the database scheme we used for our application. |



## **VALIDATION AND RESULTS**

|  |
| --- |
| Below are the detailed outputs of the corresponding functional requirements described in Section V with comparison to expected results: **Educator Challenge Interaction**  * + 1. Can set public privacy level as one of the following options Public, ShareWithLink, ShareWithDomain.     2. Can choose one of this three programming languages: C#, Java, Python.     3. Can choose challenge privacy level as ShareWithDomain and set privacy domain as a string.     4. Can create as many test cases as he/she wants by providing arguments and expected output as string.     5. Can mark test cases as hidden which will prevent them to being shown to user while he/she is solving the challenge.     6. Can create required input files by providing file name and content as text. Just like test cases can choose to hide input files from user.     7. Can create a starting template by choosing the file type as “Template”.     8. Can view the participants and score of their solutions by navigating to particular challenge in the “My Challenges” page.  **Participant Challenge Interaction**  * + 1. Can join to a challenge by entering shared key to textbox in “Explore” page.     2. If challenge privacy level is set to “ShareWithDomain” and user’s email doesn’t match with the required domain, he/she is redirected to an error page.     3. Can discover public challenges in “Explore” page by choosing a programming language.     4. If challenge is already due, he/she will get an error.     5. When user clicks on the particular challenge on “Explore” page he/she will be redirected to “Solve” page where he can solve it.     6. By clicking to save button located on “Solve” page user can save the challenge to both database and the temporary environment where solution will be compiled and run.     7. By clicking to run button on “Solve” page user can send execution request to server. If his/her solution compiles and runs without an error, results will be displayed on the same page otherwise he/she will get an error.     8. After user run his/her solution score will be saved to database.   As for functional requirement we mostly became successful. We also fulfilled our non-functional requirements such as providing a secure and scalable system. However, to be sure about these requirements we need to make serious tests preferably by people specialized in those fields. Thus, it’s hard to make a certain statement. |

## **CONTRIBUTION(S) TO INDUSTRY AND ECONOMY**

|  |
| --- |
| By making educational institutions use this product and by making the evaluation process more automated for coding education, learning process will be more productive for students and for educators. Which means less money spent for expected outcomes. Also, by making the it easier to start coding the sector will be more appealing for potential students to try it out. |

## **INNOVATIVE ASPECTS**

|  |
| --- |
| Due to security risks there are not many implementation details for other solutions to this problem. However, we are designed our system to be responsive by using WebSocket protocol, secure by using containerization(docker), and highly automated by making it a web application. |

## **REFERENCES**

|  |
| --- |
| <https://repl.it/> , <https://autogradr.app/> , <https://www.codiva.io/> – 20.11.2020  [Background tasks with hosted services in ASP.NET Core | Microsoft Docs](https://docs.microsoft.com/en-us/aspnet/core/fundamentals/host/hosted-services?view=aspnetcore-3.0&tabs=visual-studio) – 05.01.2020  [Process Class (System.Diagnostics) | Microsoft Docs](https://docs.microsoft.com/en-us/dotnet/api/system.diagnostics.process?view=netcore-3.0) – 15.12.2019  [Get started with ASP.NET Core SignalR | Microsoft Docs](https://docs.microsoft.com/en-us/aspnet/core/tutorials/signalr?view=aspnetcore-3.1&tabs=visual-studio) – 06.01.2020  [Enable Cross-Origin Requests (CORS) in ASP.NET Core | Microsoft Docs](https://docs.microsoft.com/en-us/aspnet/core/security/cors?view=aspnetcore-3.1) – 06.01.2020  [Dockerize your C# Application - Codefresh](https://codefresh.io/docker-tutorial/c-sharp-in-docker/) – 01.01.2020  [Create a web API with ASP.NET Core and MongoDB | Microsoft Docs](https://docs.microsoft.com/en-us/aspnet/core/tutorials/first-mongo-app?view=aspnetcore-3.1&tabs=visual-studio) – 02.01.2020  [MongoDB C#/.NET Driver — MongoDB Ecosystem](https://docs.mongodb.com/ecosystem/drivers/csharp/) – 02.01.2020 |

# **PROJECT RESULTS**

## **CHANGES TO PROJECT PLAN**

|  |
| --- |
| As we progressed in our project, we realized some other options for the system, that were not in our first project design, that would fit better with our priorities. Such as using Docker container to separate our code or separating our Execution Service completely to increase the security and overall scalability of the system.  Also, we crossed out some functional requirement from our to do list since they were not our priorities for this project. Mostly we focused on the parts which will make this project different from others. |

## **PROJECT MILESTONES AND OBJECTIVES**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Milestone #** | **Primary Objective** | **Due Date** | **Project Deliverable (if any)** | **Milestone Achieved?** |
| 1 | Designing system architecture and user interface. | 13/11/2019 | System architecture diagram, user interface design. | Yes |
| 2 | Designing database scheme, API interfaces and background service structure. | 13/12/2019 | Database scheme document, API use case document, progress report | Yes |
| 3 | Implementing user interface, APIs and background service. | 27/12/2019 | Back-end scheme and user interface code | Yes |
| 4 | Integrating the whole system and testing. | 10/01/2020 | App code and final report | Yes |

## **PROJECT PRACTICES AND MEASURES**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Task #** | **Task Description** | **Responsible Team Member** | **Start Date** | **Finish Date** | **Success Criteria** | **Task Achieved?** |
| 1 | Designing system architecture | Halil Etka Tutkun | 01/11/2019 | 13/11/2019 | Performant and secure architecture in theory? | Yes |
| 2 | Designing user interface | Ugur Küçükterzi | 01/11/2019 | 13/11/2019 | Nice looking and architecture compatible? | Yes |
| 3 | Designing background service | Said Furkan Ayvaz | 01/11/2019 | 13/12/2019 | Implementable with the C# language? | Yes |
| 4 | Designing database scheme and API interfaces | Halil Etka Tutkun | 13/11/2019 | 13/12/2019 | Database provides data for all use cases? | Yes |
| 5 | Implementing all designed parts. | All | 13/12/2019 | 27/12/2019 | Do implementations match with designs? | Yes |
| 6 | Integrating whole system. | All | 27/12/2019 | 10/01/2020 | Do all system parts compatible with each other? | Yes |

|  |  |  |
| --- | --- | --- |
| **Team Member** | **Task # Under Responsibility** | **Description of the Work Done** |
| Said Furkan Ayvaz | 3 | Implemented a service which takes user request to compile & run inputted code and returns the output of that code. This service also does all other related works to provide this functionality like creating & deleting environments and running docker containers. |
| Halil Etka Tutkun | 1,4 | Identified the functional requirements and designed the system architecture accordingly. Designed the database scheme. Implemented the WebApp which allows users to manipulate data using the interface. Also implemented the Execution Service which takes user requests through WebSocket puts them in a concurrent queue and assigns them to the background service. |
| Ugur Küçükterzi | 2 | Designed the user interface which covers all functional requirements and implemented them using HTML and CSS. Also implemented communication mechanisms between interface and server. Like AJAX calls and WebSocket client. |

## **PROJECT BUDGET**

|  |
| --- |
| No money spent to develop this project (except the money to buy coffees to keep ourselves awake at night) |

## **PROJECT RISKS**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Risk Item #** | **Description** | **Probability** | **Effect** | **Did It Happen?** | **How did you handle its occurrence if happened? (Plan-B)** |
| 1 | Sandboxing programs won’t be enough for securing user inputted code. | Medium | System won’t be secure for public use | Not encountered any security issues but due to the difficulty of implementation we gave up on this idea. | Instead we used Docker containers to securely run the user inputted code. |
| 2 | Low performance due to high text transaction. | Medium | User might experience slow response time | No, performance and response time seem to be ok with the technologies we used. |  |

## **SELF EVALUATION**

|  |
| --- |
| When we look at our final product it mostly matches with the requirements that we suggested in our project proposal, even though we had some changes in the way we implement them. That’s due to fact that we’ve been learning new methods and information on the way. Overall we can say that everyone in our team learned about some new technologies and how to use them and that’s the most important thing for us. We also witnessed all the steps of a software development process which will give us great insight on designing our upcoming development processes in our careers. |

## **LESSONS LEARNED**

|  |
| --- |
| Of course, if we knew the things that we learned during carrying out our project we would do some things faster. Especially if we had some experience with MongoDB and Docker Containers. We were aware of the existence of these technologies but didn’t know how to use them exactly. Other than that, we think we did a great job design-wise. One thing we could do would be starting to implement our theories sooner because we realized without starting the real implementation you never know if it’ll actually work. |