Bilkent University

Department of Computer Engineering

Senior Design Project

Final Report

Project Name: Mutrivia

Team ID: T2316

Group Members: *Lara Fenercioğlu, Gökhan Taş, Sebahattin Utku Sezer, Zeynep Büşra Ziyagil, Bedirhan Sakinoğlu*

Supervisor: *Halil Altay Güvenir*

Innovation Expert: *Melih Gezer*

Table of Contents

1. Introduction

Museums are learning environments that primarily focus on engaging and educating communities. Nevertheless, many people visit museums without paying enough attention to learn about exhibited objects. People miss tons of information, and even though they read it carefully, the information is lost after a while. Besides, some people consider museum trips as boring activities. To prevent these issues, museums have initiated projects which will increase visitor interaction and make museums more informative. For this purpose, gamification is one of the methodologies used at museums [1]. As Mutrivia, we aimed to develop a quiz game that utilizes artificial intelligence to generate multiple-choice questions. During museum trips, the visitors will play this quiz to evaluate how well they performed and at the end, we will give an award from the museum's gift shop. Therefore, the visitors will learn while having fun.

2. Requirements Details

Mutrivia is a quiz-based competitive game powered by natural language processing for museum visitors. Mutrivia will consist of two different types of clients: Museums and visitors. Museums will pay Mutrivia and be registered into the database system. They will use it for increasing the flow of people, increase the interaction, and make their museum more enjoyable and informative at the same time. For a museum to be available on Mutrivia, that museum's artifacts' information must be uploaded to Mutrivia so that questions can be generated using NLP (Natural Language Processing). When it comes to the visitors, they will be able to play a quiz-based competitive game that consists of questions about the museum they have visited. They can play either with a group or solo. When they play as a group, a session will be created, and there will be a host user. The host will share the session ID, which will be unique for the current session. When the game starts, participants of the session will earn scores for each correct answer according to their response time. According to the final scores of the participants, museums will be able to give some small rewards to top scorers. The museums will decide on these rewards. During the game, visitors answer the quiz questions that were generated before. The system will generate questions from the given text and provide four answer options, one of them being the correct answer. One of our challenges will be generating false answer choices as distractions. The system must provide answers similar to the real answer; thus, the player should be distracted.

3. Final Architecture and Design Details

3.1. Software Architecture

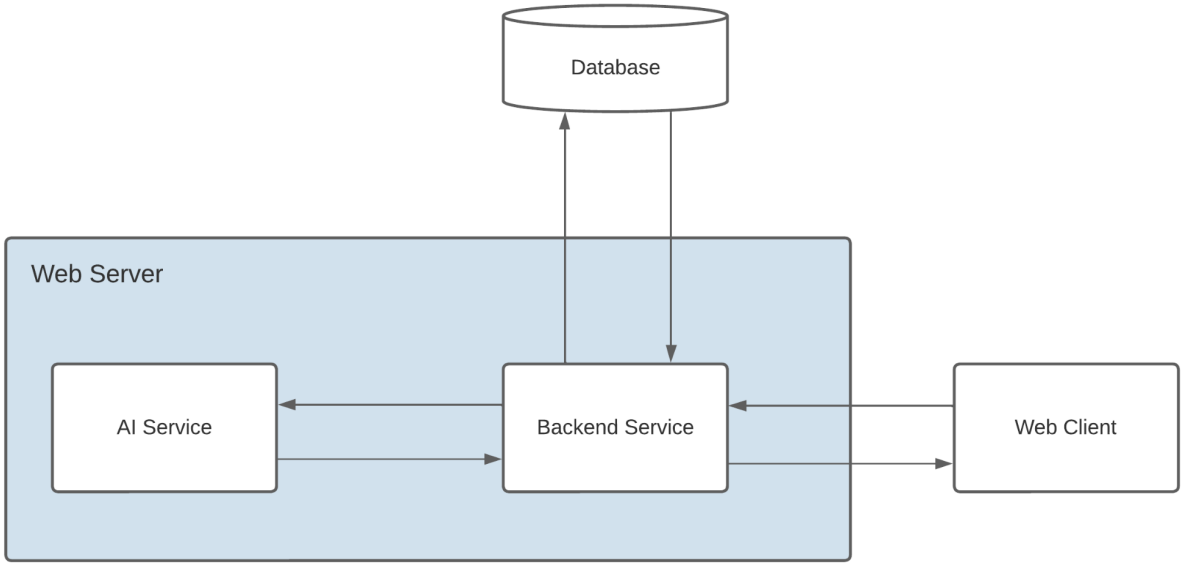

Figure 1: Architectural Overview of Mutrivia

The software architecture of Mutrivia is an example of client-server architecture. Artificial Intelligence Service is utilized for generating questions from texts and the Backend Service which is responsible for database operations and business logic of Mutrivia is deployed on the web server. Mutrivia web client initiates a request from the server to utilize services and resources and the server side responds to the request by providing the requested services and resources to the web client. There are several communication protocols used between services and clients of Mutrivia for communication. Between web client and backend service, HTTP and WebSocket protocols are used. This connection between the web client and the backend service is used for delivering the requests of the client to the backend service, delivering the response of the backend service to the client, and updating the client side according to the changes in the backend service. Backend service and AI service communicate through RabbitMQ which is a message broker using Advanced Message Queuing Protocol (AMQP). This connection is utilized for delivering new texts from the backend service to the AI service and delivering the generated questions from the AI service to the backend service.

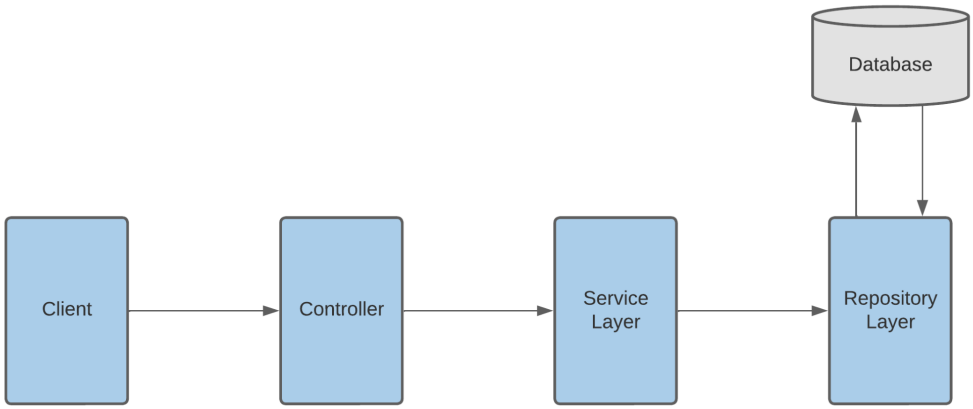

Figure 2: Backend Service Architecture

In the backend service of Mutrivia, layered architecture is utilized as we implemented it by using the Spring Boot framework. Layers in the backend service are arranged hierarchically and each of them has specific responsibility and functionality. The controller layer of the backend service is responsible for communicating with the web client by getting requests from the client and providing the responses to these requests. The business logic used in the backend service is implemented in the service layer. The repository layer is responsible for communicating with the database in order to perform database operations.

3.2. Server Architecture

Server architecture refers to the design and structure of a server system that performs various tasks. It consists of different components such as hardware, software, and networking components. They work together and provide services to the users. In order to explain our server architecture, it was split into four different sections. Our server architecture diagram can be seen below:

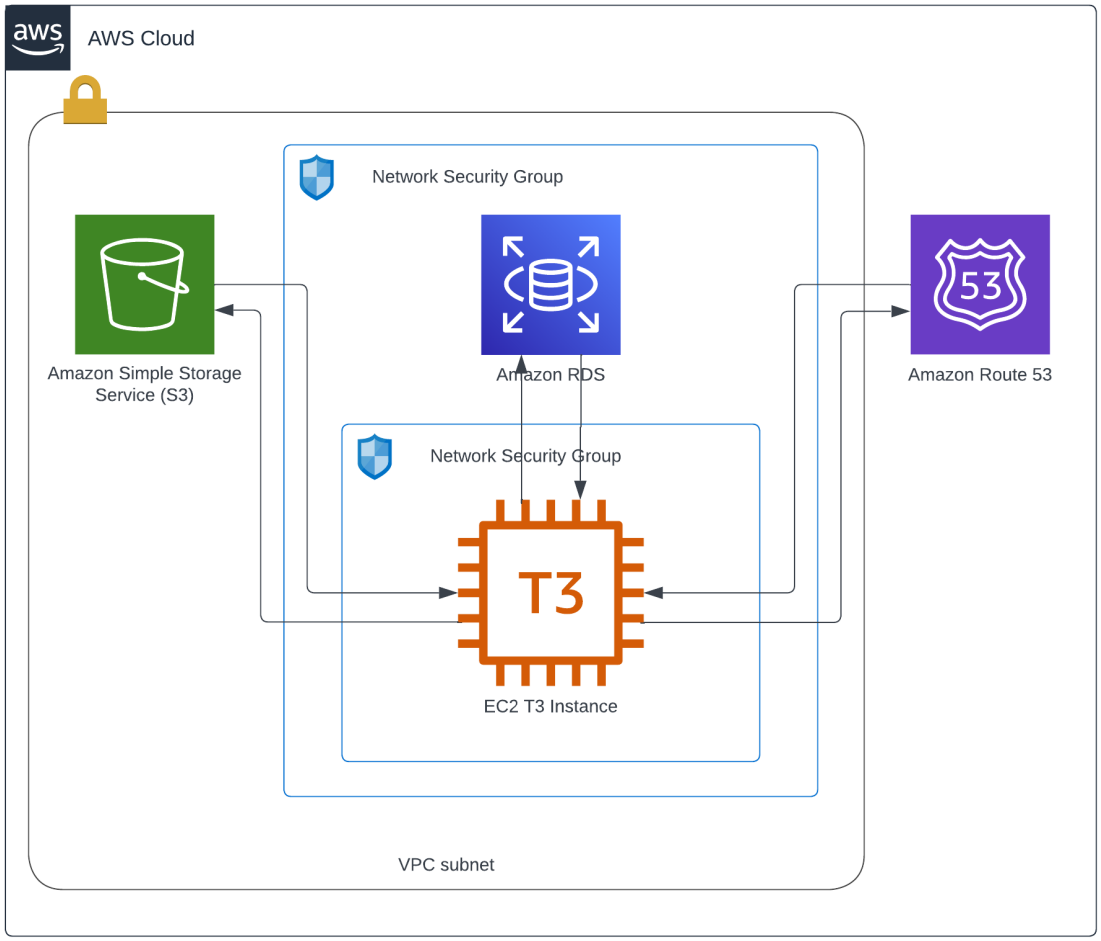

Figure 3: Server Architecture

1. Hardware and Operating System

In our design, we have an EC2 T3 instance that contains our web server. It has 8 GB RAM and 16 GB storage capacity. Also, it has 2 vCPUs. Web server runs on Fedora which is a popular Linux distribution. Since we used a Linux distribution, it reduced our cost and it is easy to configure compared to Windows operating system-based servers.

2. Software Services

In our design, we have an Amazon Simple Storage Service (S3) which contains our files such as executables, and configurations. It enables us to store parts of our application so that we can restore them if our web server fails.

When it comes to databases, we used Relation Database Service (RDS) which is based on MySQL. Since we used a service in the AWS Cloud and our services are also in the same cloud, it helped us to configure and maintain the connectivity between services.

In order to host our application, we used Nginx as a web server to handle HTTP requests and serve our web application to clients. Moreover, it was configured as a reverse proxy to handle our API request in the Spring Boot application. Since we were familiar with Nginx, and it provides lots of additional functionalities such as reverse proxy, and load balance, it was easy to use and configure.

3. Networking

In our design, we used Route 53 Service as our DNS server. We configured DNS records according to our domain name. In addition to that Network Security Groups and Virtual Private Cloud (VPC) are used to enable or limit the connection. According to our needs, we implemented several rules in our access control list (ACL).

4. Scalability and High Availability

Since we use AWS Cloud Service, it is highly scalable in both vertically and horizontally. When it comes to availability, we aimed to minimize our downtime by using the services of AWS and its tools.

4. Development/Implementation Details

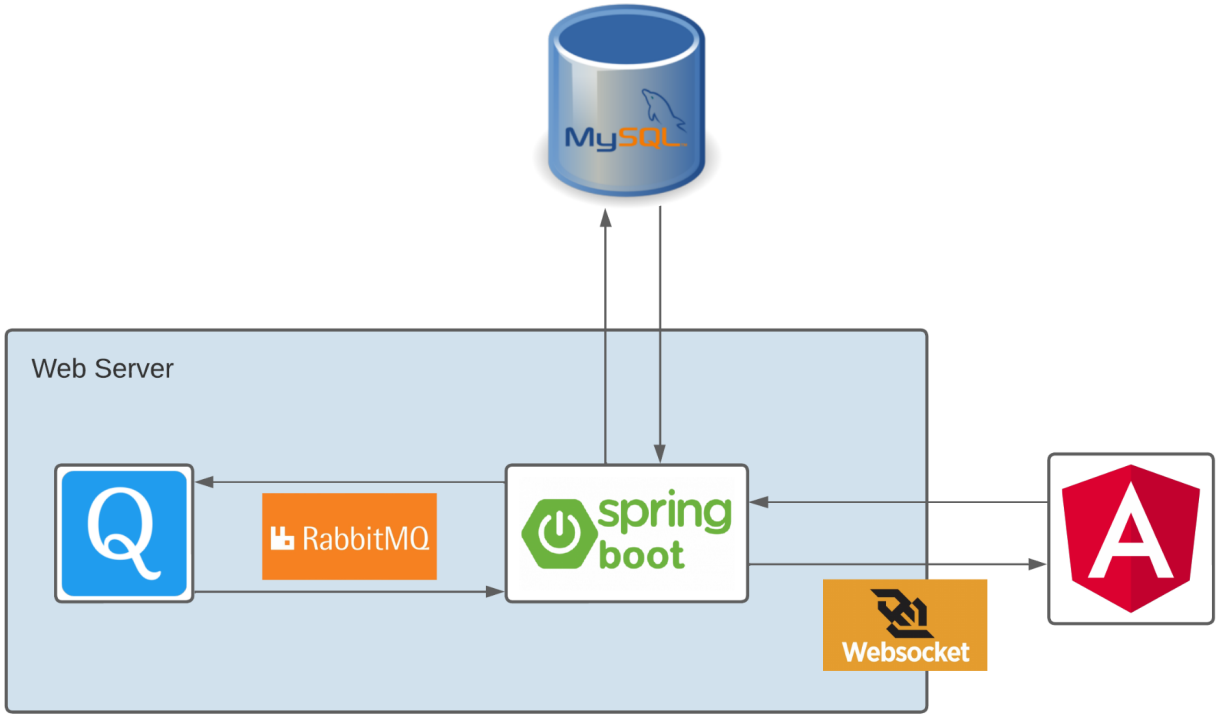

Figure 4: Technologies used in Mutrivia

- **Angular:** Angular is utilized for developing frontend of Mutrivia. Angular is a web application framework that is used for developing dynamic and interactive single-page web applications.
- **Spring Boot:** The backend Service of Mutrivia is developed by using the Spring Boot framework. Spring Boot is a Java framework that is used for developing a wide range of applications, specifically web applications.
- **WebSocket:** Apart from RESTful API which is used for HTTP communication, there is a WebSocket connection between the backend service and the web clients. WebSocket is utilized in Mutrivia to update the user interface according to changes happening in the backend service.
- **MySQL:** As a design choice, we decided to use a relational database model for Mutrivia. We chose MySQL due to its high performance, popularity, and community support.
- **RabbitMQ:** RabbitMQ is an open-source message broker software that provides a messaging system for applications to communicate with each other. It uses a “producer-consumer” model where the producer sends messages to the broker

and the consumer receives those messages from the broker. In order to establish connection asynchronously, and to ensure that messages are delivered reliably.

- **Questgen:** Questgen is a AI-powered tool that generates questions according to a given text. We use texts of artifacts in the museum to feed Questgen. Questgen was trained by using 2015 Reddit data. With the help of Questgen, we are able to generate questions from the given text and provide four answer options, one of them being the correct answer.

5. Test Cases and Results

Test ID	**Test Type**	**Summary/Title**	**Procedure**	**Expected Results**	**Priority/ Severity**	**Test Results: Success or Not Applicable (NA)**
1.1	Functional	Score increases when you answer correctly on the quiz page	User entered the session via id. The host starts the session. The user is shown a question and the answer is selected right. The score of the user has checked whether it increased or not.	The score is increased after the right answer is chosen	Major	**Success**
1.2	Functional	After the countdown finishes, the answer buttons disable	the Session is started by the host. The user is shown a question and the countdown starts. The countdown finished the buttons are checked whether they are available or not.	Buttons are disabled according to time successfully	Major	**Success**
1.3	Functional	Score changes according if the question is	Session is started by the host. The user is shown a	Score is computed according to the	Major	**Success**

		answered correctly depending on the time left	question and the answer is selected right. The score is checked if it is computed according to the remaining time.	remaining time		
1.4	Functional	Question answering time still decreases even app is closed	Session is started by the host. The user is shown a question and the countdown starts. The app is left for another. Time is checked whether the countdown continues.	The countdown still continues	Major	**Success**
1.5	Functional	When the answer is correct the score is added to the database	User entered the session via id. The session is started by the host. The user is shown a question and the answer is selected right. Check whether the score of the user is posted to the database.	The score of the user is posted to the database	Major	**Success**
1.6	Functional	Score does not change when the answer is wrong	The user entered the session via id. The session is started by the host. The user is shown a question, and the answer is selected wrong. The score of the user has been checked whether it increased or not.	The Score is the same after wrong answers are chosen	Major	**Success**
1.7	Functional	All question	Session is started	The buttons are	Major	**Success**

		choices can be clicked only once	by the host. The user is shown a question, and the countdown starts. A button is clicked as an answer. The buttons are checked to determine whether they are available or not.	unavailable		
1.8	Functional	Count down starts as the questions are available	Session is started by the host. The user is shown a question and the timer is checked whether the countdown started.	The countdown started	Major	**Success**
1.9	Functional	When the username is set it is added to the database	User opened the app. Username is asked for by the app and entered by the user.	User name is set and sent to database	Major	**Success**
1.10	Functional	Checking if the start session button starts the session for users in session	User opened the app and started a session as host. Shared the id of the session and other users entered. The host pressed the start session button. All users in the session check whether their quiz started or not.	All users in the session started their quiz	Major	**Success**
1.11	Functional	Checking the right museum is chosen based on location data	User opened the app. The museum user is shown via location data. The museum shown is checked whether it	The museum shown is based on the location	Major	**NA**

			is based on the location.			
1.12	Functional	Check if the location data is available	User opened the app. Checks whether the location data is available.	The location data is available	Major	**NA**
1.13	Functional	Checking when the host ends the session, the scoreboard is shown	User opened the app and started a session as host. Shared the id of the session and other users entered. The host pressed the start session button. Questions are answered. Then host pressed the end session button. Check whether the session ends with attendees.	The session ends with attendees	Major	**Success**
1.14	Functional	Checking when the session end, new scores are added to the leaderboard if there are any all-time high scores.	User opened the app. Username is asked for by the app and entered by the user. Checks whether users can see the scores in the leaderboard.	Users can see the scores on leader board	Major	**Success**
1.15	Functional	Checking when the session ends, attendees see the scoreboard	The host pressed the start session button. Questions are answered. Then host pressed the end session button. Check whether the session attendees see their session	Attendees are able to see their session scoreboard.	Major	**Success**

			scoreboard.			
1.16	Functional	Checking when the host ends the session, participants can see the winner	The host pressed the start session button. Questions are answered. Then the host pressed the end session button. Check whether the host can see the session scoreboard and winners.	The host can see the session scoreboard and winners.	Major	**Success**
1.17	Functional	Checking if there is a call ongoing, ensure the app is running in the background	User opened the app. There is a call ongoing. Checks whether the app runs in the background.	The app runs on the background.	Moderat e	**NA**
1.18	Fuctional	If a user joins another host's session, check the users shown on the same session	After a user, other than the session host, joins a session, check which usernames can the newly joined user see.	There should be at least the host's username on the same session and the newly joined user's username. If there are more users joining in the same session, the names should appear.	Major	**Success**
1.19	Functional	If a user starts a solo session quiz starts with the start button	The user pressed the start session button. Check whether the user can answer the questions within the quiz.	The questions are shown, and the countdown starts.	Major	**Success**
1.20	Functional	If a user is in a session and during the session can see her score	The user pressed the start session button. Questions are answered. Check whether the user can see the	The score is shown to the user while playing.	Major	**Success**

			score while playing.			
1.21	Functional	If a user is in a solo session and ends the quiz, the leaderboard is shown with the user's place if the user is a high scorer.	The user pressed the start session button. Questions are answered. Then user pressed the end session button. Check whether the users can see the session leaderboard with his name included if he is a high scorer.	The leaderboard is shown with the user's place.	Major	**Success**
1.22	Functional	Check if a user can see the question's answer by using the 'inspect' functionality that the web browser provides	If a user tries to inspect all answers to a question, they should not be able to observe if that answer is the correct answer for that particular question.	User should not be able to see if an answer is the correct answer by inspecting all answers on any web browser user uses	Major	**Success**
1.23	Functional	User can't enter a session ID that is not available	Is a user tries to enter a session ID on the main menu and if there is no such ID, then the user should be prompted and can't enter the session.	Session not available	Major	**Success**

1.24	Functional	User can change location sharing setting to on and off via the setting menu	User opens the app. Enters the settings menu on the top right of the screen. Modifies the location-sharing setting. Checks whether it is changed to choice.	Location sharing setting is modified	Moderate	**NA**
1.25	Functional	User can change music setting to on and off via the setting menu	User opens the app. Enters the settings menu on the top right of the screen. Modifies the music setting. Checks whether it is changed to choice.	Music setting is modified	Moderate	**NA**
1.26	Functional	User can change the sound effects set to on and off via the setting menu	User opens the app. Enters the settings menu on the top right of the screen. Modifies the sound effects setting. Checks whether it is changed to choice.	Sound effects setting is modified	Moderate	**NA**
1.27	Functional	User can change language settings from dropbox via the setting menu	User opens the app. Enters the settings menu on the top right of the screen. Modifies the language setting. Checks whether it is changed to choice.	The language setting is modified	Moderate	**NA**

1.28	Functional	User can change the font size setting via the setting menu	User opens the app. Enters the settings menu on the top right of the screen. Modifies the font size setting. Checks whether it is changed to choice.	Font size setting is modified	Moderate	**NA**
1.29	Functional	Check for vibration effect if present after time is up	After time is up on the answering page, the phone should vibrate indicating the countdown has finished.	Vibration is enabled.	Moderate	**NA**
1.30	Functional	Check for vibration effect if present after the game is finished.	After the game session is finished, the phone should vibrate indicating that.	Vibration is enabled.	Moderate	**NA**

1.31	Functional	User can change the vibration effects setting to on and off via the settings menu	User opens the app. Enters the settings menu on the top right of the screen. Modifies the vibration setting. Checks whether it is changed to choice.	Vibration setting is modified	Moderat e	**NA**
1.32	Functional	Verify if the loading symbol is displayed wherever required	User is in the session. The loading symbol is shown in between questions due to generation time.	The loading symbol is shown	Minor	**Success**
1.33	Functional	Check for scrolling	In the page scoreboard or leaderboard, the user may see lots of scores of each user and they can't be displayed on one page so scrolling to see each user's score should be enabled.	Users can scroll through the scoreboard or leaderboard.	Minor	**Success**

1.34	Functional	Max character in nickname input field	A user must enter a nickname before logging into the session and	the User is not allowed to get a nickname with characters more than the defined max character value	Minor	**NA**
1.35	Functional	If a user is inactive for too long, the user should be kicked out of session.	After the user enters the session, if s/he stays away for too long then the app will automatically remove the user	App won't allow users to stay away for a long time to play the game	Moderate	**NA**
2.36	Functional	Check what happens if the AI service cannot produce any question for any particular text.	When the backend service sends a text for any artifact to the AI service, it is expected for the AI service to create at least one question. What happens if it cannot create any question for that text (this is expected if the text has lack information).	a User should not see a blank question with blank answers. Instead, that text should be skipped for all existing users in that session if the AI service cannot create any questions for it.	Major	**NA**

2.37	Functional	Check if the questions generated by the AI service are inverted.	After the AI service creates a question for any quiz session, check if the question sentence is inverted or if the sentence has a meaning.	All questions need to make sense for humans for Mutrivia to be consistent, and the sentences should not be inverted	Major	**Success**
3.38	Compatibility	Cross-browser compatibility	Each scenario is checked in browsers such as Internet Explorer, Firefox, Chrome, Safari, and Opera via BrowserStack.	The game runs in each case via BrowserStack.	Major	**NA**
3.39	Compatibility	Testing on mobile devices	The application will be tested on both ios and android OS.	The game run on both OS.	Moderate	**NA**
3.40	Compatibility	To validate that the user interface of the application is as per the screen size of the device, no text/control is partially invisible or inaccessible	The application will be tested on different-sized mobile phones.	The game should be the same scaled visibility in all mobile apps.	Moderate	**Success**
3.41	Compatibility	It is to check the application in different networks like 4G, WIFI, etc.	The application is started. Checks the connected network is compatible with the app, and data is exchanged	The game is available, and data is exchanged	Major	**Success**
4.42	Performance	Showing scoreboard less than 2 seconds	User will click the show scoreboard button.	Scoreboard should be displayed in less	Moderate	**Success**

				than 2 seconds.		
4.43	Performance	Generating questions in less than 5 seconds	User will start the session and view questions one by one	Each question should be generated in less than 5 seconds.	Moderate	**Success**
5.44	Rationality	Generating meaningful questions	Since we generate questions through AI, questions should reflect meaningful statements and answers.	Users will see rational questions.	Major	**Success**
5.45	Rationality	Check if the scores are calculated properly	After answering a question correctly, the user should get the point according to the countdown left.	User will see his/her score properly according to the questions answered correctly.	Major	**Success**
6.46	Reliability	Right answer is chosen the score is calculated immediately	User has chosen the right choice. Checks the score calculation is started immediately.	The score calculation is started immediately.	Moderate	**Success**
6.47	Reliability	Web app must be accessible on the website of Mutrivia	User will access the application through the website	App should be accessible.	Moderate	**Success**
7.48	Security	Only authorized persons should reach the artifact information of museums	Unauthorized persons will try to reach information in the database	Connection should be refused.	Major	**Success**
8.49	Installation	App should be available to be installed on App Store	User will try to install the app on App Store	App should be installed	Moderate	**NA**
8.50	Installation	App should be available to be	User will try to install the app on	App should be installed	Moderate	**NA**

		installed on Google Play Store	Play Store			
9.51	Usability	Texts on the application should be readable by users	User will try to read prompts and texts on mobile device	Texts should be easily readable	Moderate	**Success**
9.52	Usability	Answer options should be displayed correctly	When the user gets the question, answer options are selectable and readable.	Answer options are displayed correctly	Moderate	**Success**

Table 1: Table of test cases

6. Maintenance Plan and Details

A comprehensive maintenance plan is required to maintain the smooth running and continued enhancement of the Mutrivia application.

To address potential bugs, security vulnerabilities, and compatibility issues, regular updates for the Mutrivia application should be implemented. These updates will include improvements to the NLP algorithm, performance optimizations, and new feature additions. The updates will be released periodically, and feedback from museums and visitors will be taken into account for continuous enhancement.

The museum artifacts' information database is crucial for generating questions in Mutrivia. Regular maintenance and backups of the database should be performed to prevent data loss. Additionally, as new artifacts are added to museums or existing artifacts are modified, an efficient mechanism should be in place to update the database accordingly.

Mutrivia will handle a significant amount of user traffic during peak periods, especially when multiple museums and visitors engage simultaneously. Continuous monitoring of the server infrastructure will ensure optimal performance and availability. Scaling mechanisms, such as load balancing and auto-scaling, should be implemented to handle increased traffic and ensure a seamless experience for users.

A dedicated user support team should be available to address any queries, issues, or feedback from museums and visitors. This team will provide timely responses, troubleshooting assistance, and guidance to maximize user satisfaction. Regularly analyzing user feedback will help identify areas of improvement and drive future updates and enhancements.

Mutrivia should be optimized for fast and responsive performance to provide an enjoyable user experience. Regular performance monitoring and profiling should be conducted to identify and resolve any bottlenecks. Techniques like caching, database indexing, and code optimization should be employed to improve system responsiveness and reduce latency.

As stated in the requirements, the initial version of Mutrivia will support the English language. However, future plans involve expanding language support to include Turkish. The maintenance plan should consider the implementation of language localization, including translation of question texts, user interfaces, and support documentation, to reach to a wider audience.

Thorough testing and quality assurance processes should be implemented to ensure the stability, functionality, and usability of Mutrivia. This includes comprehensive testing of new features, regression testing, usability testing, and performance testing. Continuous integration and automated testing practices should be employed to streamline the development and deployment process.

Integration of analytics tools within Mutrivia will provide valuable insights into user behavior, engagement patterns, and system performance. Tracking metrics such as the number of active users, average session duration, popular artifacts, and user feedback will help in making data-driven decisions for future updates and improvements.

The maintenance plan for Mutrivia involves regular system updates, database management, server monitoring and scaling, user support and feedback, security measures, performance optimization, language support and localization, continuous testing and quality assurance, and analytics integration. By implementing this plan, Mutrivia can ensure a stable, secure, and engaging experience for both museums and visitors, fostering increased interaction and learning within the museum environment.

7. Other Project Elements

7.1. Consideration of Various Factors in Engineering Design

As part of our analysis, we consider how our system will be constrained or affected by the factors such as public health, safety, and welfare, as well as global, cultural, social, environmental, and economic factors.

Public Health

Mutrivia is a quiz game aiming to enhance the museum trip experience for visitors, making it more fun and informative. There is no direct effect of Mutrivia on public health. However, the pandemic might have a negative impact since Mutrivia can increase the flow of people in museums.

Safety

Since Mutrivia is designed for museums to be more informative while being fun, it does not directly correlate to individual or public safety.

Welfare

Mutrivia aims to enhance the museum trip experience of visitors, making it more fun and informative. There is no direct effect of Mutrivia on public welfare. It may increase people’s knowledge gained from museums, resulting in social and cultural awareness indirectly. But, increasing social and cultural awareness is not one of Mutrivia’s main aims.

Global

The target audience for Mutrivia is everyone around the globe. Since every museum can support Mutrivia’s quiz at no additional cost, everyone can use Mutrivia in the supported museums. Mutrivia will only support museums with English artifact descriptions to be accessible by any museum. The data of museums will be collected in English to be compatible globally.

Cultural

The app aims to increase visitor interaction in museums by using gamification to increase the tour experience. Mutrivia will enhance the visitor’s engagement, and the user will learn more from the experience due to the increase in engagement. Through this process, users' cultural engagements will be augmented. Since the game will be in English and it is used worldwide, there will not be a language barrier between people from different countries and cultures. Moreover, Users' personal beliefs or preferences will not be collected or used in any way.

Social

The only information asked from the users is the username, and it does not have to be the real name of the user; they all will be considered nicknames. From a social perspective, since no personal data of any user is saved in the platform, there are no concerns for the sake of the user's privacy.

Environmental

Mutrivia has no environmental considerations, such as reducing its carbon footprint. So, the application has no direct or indirect effects that can affect environmental issues.

Economic

Mutrivia is expected to be accessible to any user. An annual subscription must be bought for a museum to be added to Mutrivia for a year. To make the application free for users, development costs must be minimized. Some free alternatives, such as adding ads to applications, can be considered if needed.

	Effect Level	**Effect**
Public Health	2	Mutrivia may increase the number of visitors during a pandemic.
Public Safety	0	No effect
Public Welfare	0	No effect
Global Factors	5	English is used as it is a global language. The addition of a museum should only require data in English.
Cultural Factors	9	No discrimination will be made in terms of language and questions, as the same algorithm will be used for every museum.
Social Factors	3	Mutrivia does not require personal data.
Environmental Factors	0	No effect
Economic Factors	5	Costs must be minimized.

		Free alternative ways may be considered.

Table 2: Table of factors and effects

7.2. Ethics and Professional Responsibilities

Since we are a group of 5 people, we all have a responsibility to each other. We need to respect each other since all of us have different school lives where some of us had more courses than others which may lead to unequal work division. However, we put strict deadlines and work divisions accordingly. Furthermore, we made sure that each of us take the responsibility for where s/he had more knowledge and confidence.

We didn't consider privacy as a design goal so there were no ethical responsibilities that we take care of however the data coming from museums might be confidential thus we kept those in a database that only admins can reach.

Another responsibility that we considered is the implementation. There is a possibility that our group may have to deal with bugs at the end of the implementation. Thus, to solve the bugs more easily and early, group members are required to follow coding conventions such as commenting on the code and writing clean code as a professional responsibility.

7.3. Teamwork Details

7.3.1. Contributing and functioning effectively on the team

We mainly distribute the work equally among each other while considering the load we have in school as well. Some of us have a different amount of knowledge in the field of the project. Thus, we can help each other at any time. When we have meetings, we usually divide work and talk about future plans. If most of us have time, we start a task such as a final report. After finishing our own part, we always ask one of us to check the writing.

For proper division of the tasks, the documentation of the project is done with the equal contribution of every member of the group. Front-end development and overall testing are performed by Sebahattin Utku Sezer and Lara Fenercioğlu. are done by Bedirhan Sakinoğlu. Back-end implementations and bug fixing are completed by Bedirhan Sakinoğlu. Finally, deployment, Database implementation, and integrations were

Gökhan Taş's and Zeynep Büşra Ziyagil's responsibility. All the members of the project contribute to the project demo.

Moreover, we hosted Discord meetings at least twice a week to ensure that everyone was participating in the project, and when we started implementation we met twice a day so that we can ask our technical questions easily.

7.3.2. Helping to create a collaborative and inclusive environment

Since our team is all friends from the past, we usually easily communicate with each other and create a collaborative environment. Our team members met via online platforms like Discord in general. To pursue a stronger connection, we met inside of school and made crucial decisions face-to-face by meeting with our supervisor Halil Altay Güvenir. We do, however, host regular meetings once a week to keep everyone involved.
Sometimes we divide a task and work in pairs, and it helps us to see each other's mistakes. Therefore, we learn synchronously. After finishing a paired task, we usually show the work done to each other so that we get additional feedback. We really emphasize working in an inclusive environment. Thus, we work all the time closely.

7.3.3. Taking the lead role and sharing leadership on the team

Team of Mutrivia does not have a specific leader who leads the whole team. We mainly make essential decisions by consultation in our meetings. Although we do not have a leader within our team, we distribute tasks equally by our interests and availability. As a team, we always meet once a week to keep track of our tasks and decide on what can be done next. On our project's milestones, we arranged a meeting and worked together for everyone to get an overview of the tasks until the upcoming milestone of our project, Mutrivia.

7.3.4. Meeting objectives

We planned to build a mobile application however due to a couple of Flutter-issued problems we couldn't manage our time to understand and solve the problem. Therefore, we didn't have an application to be used on Android or iOS. However, we finished the quiz web application version on time. We were planning to create a web version of the project since some

people may not want to deal with downloading the game during their short museum trip. Thus, we started on the web version earlier and after we discovered that we can't finish the mobile version on time; we worked more on the web version such as improving the user interface and finding bugs. On the other hand, we finished the artificial intelligence-powered quiz and the game conventions on time which was the main milestone of our project.

7.4. New Knowledge Acquired and Applied

Although this was a senior project, we wanted to explain our idea to other people as well by joining competitions like Ankara Startup Summit. After competing there, we won pre-incubation support from Bilkent Cyberpark, CyberCulture thus we had the opportunity to further develop our project by getting feedback from various mentors. We learned how to build a startup company. Afterward, we decided to join a Hackathon organized by VakıfBank where we had the chance to expand our idea of Mutrivia and go beyond our scope. There, we learned how to develop an idea and present it in front of the public. Since we had previous courses related to presentation and reporting from distinctive ENG and CS courses, we had no trouble in these competitions.

On the other hand, we learned different technologies. We utilized technology with which we had a bit of knowledge but no expertise throughout the creation of this project. Although some of us already knew the technologies we used, some of us had to learn to understand the implementation and the bugs related to them. These were Dart, Flutter, AWS, Angular, and Spring. We have constantly searched for better ways while implementing the functionalities so that we could maximize the technicality of this project and minimize the defects. We have looked for similar projects and which technologies they used and completed literature reviews to understand which technologies are more suitable.

8. Conclusion and Future Work

After finishing two semesters focusing on this project, we achieved many milestones in our lives. We learned how to be a team and work collaboratively. We learned how to build a project from zero to a fully functional application.

We are going to talk with the Ministry of Culture and Tourism since we would like our application to be used all over Turkey and in order to take this milestone, we need data from the museums. Although we had some connections with private

museums like Rahmi Koç Museum, it is not enough. When we attended the CyberCulture program with Mutrivia, we had the chance to speak to the General Director of Copyrights of the Ministry of Culture and Tourism, Mr. Ziya Taşkent, and we got positive feedback from them which motivated us to keep going. Also, the General Manager of Bilkent Cyberpark, Faruk İnaltekin listened to our project and said that they can help us to reach the museums. All of these experiences contributed to our success besides school thus we plan to implement this project in real life where people can actually benefit from museums by having fun while learning. Since we couldn't implement the mobile application, we will try to implement it in Android first but using another technology rather than Flutter, afterwards, we will work on the IOS version.

9. Glossary

Gamification: The use of elements of game-playing in another activity, usually to make that activity more interesting [2].

Natural Language Processing (NLP): The use of computers to process natural languages, for example, for translating [2].

Artificial Intelligence (AI): The study and development of computer systems that can copy intelligent human behavior [2].

Flutter: An open-source, cross-platform UI development kit developed by Google.

AWS: Amazon Web Services (AWS) is an online platform that provides scalable and cost-effective cloud computing solutions.

Amazon Elastic Compute Cloud (Amazon EC2): provides scalable computing capacity in the AWS Cloud.

REST API: A REST API is an Application Programming Interface (API or web API) that conforms to the constraints of REST architectural style and allows for interaction with RESTful web services [3].

Treasure Hunt: A game in which a group or individual tries to be the first to find a hidden object.

10. References

[1] A. López-Martínez, Á. Carrera, and C. A. Iglesias, “Empowering museum experiences applying gamification techniques based on Linked Data and smart objects,” Applied Sciences, vol. 10, no. 16, p. 5419, 2020.

[2] “Oxford Learner's dictionaries: Find definitions, translations, and grammar explanations at Oxford Learner's dictionaries,” *Oxford Learner's Dictionaries | Find definitions, translations, and grammar explanations at Oxford Learner's Dictionaries*, 2022. [Online]. Available: https://www.oxfordlearnersdictionaries.com/. [Accessed: 21-Feb-2023].

[3] IBM Cloud Education, “Rest-apis,” *IBM*, 06-Apr-2021. [Online]. Available: https://www.ibm.com/cloud/learn/rest-apis. [Accessed: 12-Nov-2022].