Bilkent University

Department of Computer Engineering

Senior Design Project

Detailed Design 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*

March 13, 2023

This report is submitted to the Department of Computer Engineering of Bilkent University in partial fulfillment of the requirements of the Senior Design Project course CS491/2.

Table of Contents

1. Introduction

1.1. Purpose of the system

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]. It is planned to develop a mobile application that utilizes artificial intelligence and gamify museum trips. In addition to the mobile version of our application, we are going to design a web version of it to eliminate the burden of downloading the application. We will build an interactive question-based game that can be played on mobile phones.

1.2. Design Goals

1.2.1. Performance

Each question will be generated in real time, and the process of generation should not take more than 5 seconds for a smooth user experience. Mutrivia will store only the information texts of artifacts of museums. When a user plays Mutrivia, the user will see multiple questions which are generated in real-time. If the question generation time is too long, the user will wait too much to see the question, which will be annoying. Also, it should be scalable because multiple museums are going to be enrolled, and numerous people will be online in the system, and all requests should be handled without delay or any problem. To improve the performance of our application, we will use a dedicated server for question generation, and all the operations, such as user registration and session formation, will be made on the server side, as mobile phones are much slower at doing such calculations.

1.2.2. Usability

Both mobile and web applications will be used by many people and people of many ages. Therefore it should be user-friendly. Since we aim to make museums more entertaining, the applications should not frustrate users. Also, they should look attractive and likable since they will be constructed as games. In short, simple, easily understandable, and appealing interfaces will be provided. The user interfaces should be simple, with no

distracting themes. As the main aim of this project is to increase the interaction of people who visit museums with the exhibited objects, complex and detailed user interfaces may distract users from museum objects to only the application. However, to entertain children, for instance, the buttons can be colorful. There can also be an option for elderly people to increase the font size etc. As Mutrivia, we want everyone to enjoy our application.

1.2.3. Extensibility

We will begin with a couple of museums. Therefore, the application should be adaptable to other museums that may be included in the future. In addition, we may add new functionalities in the future, such as different language options, different levels of difficulty, and different question types.

1.2.4. Rationality

Questions that Mutrivia generates using natural language processing must be compatible with the corresponding object description text. The questions should not be out of context from the objects' descriptions. Also, the answer choices of the question should be relevant to the question, and the wrong answers should be compatible with the correct answer so that it can distract the player.

1.2.5. Portability

As it was mentioned before, there will be both mobile and web applications for our product. There will be no desktop version of this project. Mutrivia will be available on both Android and iOS operating systems. As for the web version, the latest version of current browsers will support our product.

1.2.6. Security

All the data collected from museums will be kept securely. No third-party user is going to be accessing this private data.

1.2.7. Reliability

The applications must not have extended server downtime to give users an uninterrupted experience. It also must not be laggy and have delays. As the users are competing while trying to answer the questions, the

sessions and users' local experience must not be interrupted for long or frequently.

1.3. Overview

Mutrivia is a quiz-based competitive mobile game powered by natural language processing for museum visitors. Mutrivia will consist of two different types of clients: Museums and visitors. Museums will buy the application and be registered into the system. They will use it for their purposes, such as increasing the flow of people, increasing the interaction, and making 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, each question will be generated instantaneously. 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.

Since we are planning to develop this application for global usage, the language of the application will be English. In the future, we plan to make a version that supports the Turkish language, and it will generate questions in Turkish.

2. Current software architecture

In the current market, there are some applications that aim to solve the same problem as Mutrivia does. The names of the applications we found are THATMuse and Virtual Museum.

THATMuse aims to solve the museum's lack of interest problem from the visitors by making it interactive like Mutrivia. The main difference between THATMuse and Mutrivia is that THATMuse is a treasure hunt game being played in

museums, which has no effect on interactive learning. With a treasure hunt, users of THATMuse try to be the first to find the artifact that is in the museum.

Another application that aims to solve the same problem with Mutrivia is Virtual Museum. Virtual Museum aims to carry the museum environment into virtual reality. By using this competitor application, users can travel inside the museum and learn from the artifacts as if they were in the museum. However, Virtual Museum does not encourage its users to be in the museum to utilize the application physically. While it may increase the effects of museums' learning factors in pandemics or similar situations, it does not make museums more interactive.

That being said, Mutrivia has some competitors that aim to solve common problems as Mutrivia, with different approaches. Despite these applications being a great source for both interactivity and enhanced learning for museums, none of them makes both of these improvements. With the use of artificial intelligence in question generations, we aim to make museum trips more instructive, and by making question-answering a competitive game, we aim to make museum trips more interactive and entertaining.

3. Proposed software architecture

3.1. Overview

In this section, the software architecture and the subsystem decomposition of the system are explained. Data management and security concerns are also addressed below.

3.2. Subsystem decomposition

Mutrivia has a client-server architecture. The server side is divided into two; ai service and backend service. Client-side is the mobile and web applications. Client-side will use and depend on the server side to handle user inputs, updates and get information from the AI service. It consists of a user interface, which includes views such as quizzes or menus. Then, there are its helper modules, such as request and update modules which are used for communicating with the backend service. The project will be up and running on AWS servers. The backend and AI services, and persistent data will be handled on that side. Backend services consist of a user module, game module, session, and text data modules. Then, the connection modules such as WebSocket, Rest APIs, and RabbitMQ. AI

services consist of a question generation module, and it also contains a RabbitMQ module to communicate within persistent data that will be handled by MySQL. The users can only access these layers via API requests.

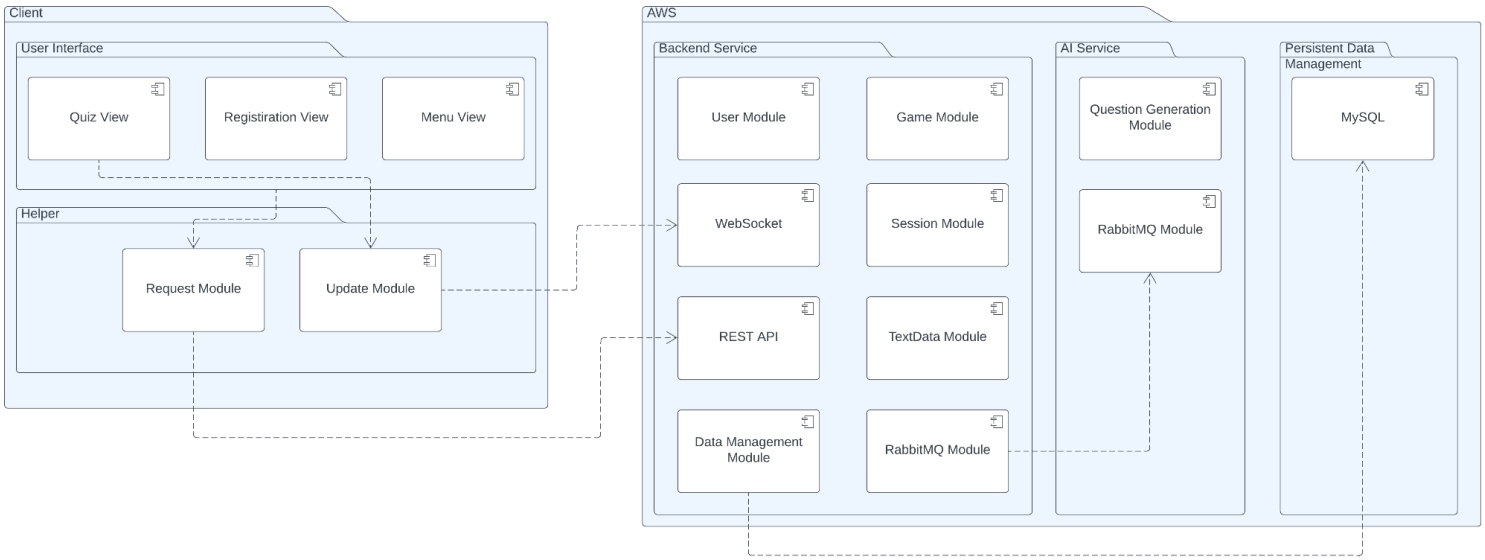

Figure 1: Subsystem Decomposition
https://imgur.com/a/sjXrh2B

3.3. Hardware/Software Mapping

Our mobile application operates on both iOS and Android devices on the client side, with the client hardware solely responsible for displaying views. We will also have a web application as a client-side application. The business logic and data layer are in the cloud, leveraging AWS. As a result, clients must use HTTP to send requests to the server for communication. On the server side, we use EC2 machines and workers to handle the traffic. Our system's hardware/software mapping is described below by a UML Deployment Diagram.

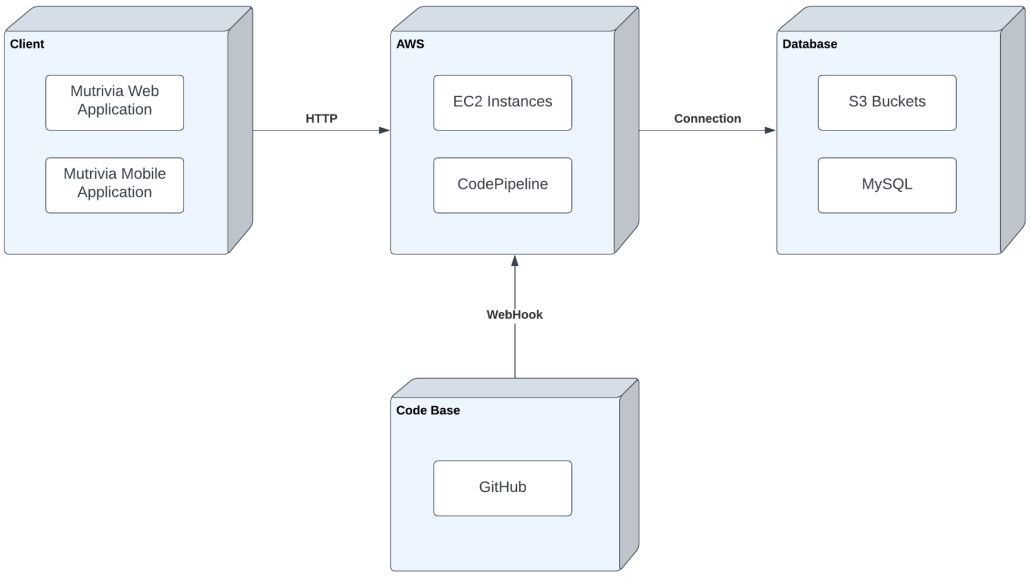

Figure 2: Deployment Diagram
https://imgur.com/a/0UluhZv

3.4. Persistent Data Management

Data obtained from museums will be stored in Mutrivia's database. Information texts about objects in museums will be utilized to generate questions by retrieving them from the database. Additionally, user data, such as their nickname, status, sessions, and scores, will be stored on this database. In addition, information about created sessions will also be stored in the database. Also, there will be a leaderboard showing all-time high scorers, which we will be held in the database.

3.5. Access Control and Security

Mutrivia keeps the user's nickname and score information. These data will be stored specifically for a session, but only scores will be added to the all-time leaderboard. Also, artifact texts from the museums will be stored in the database, and these won't be shared with third-party systems. Therefore, it is possible to encrypt stored data using MySQL's functionality. Thus, there are no security concerns. Besides, there are two types of users where one user must be the host, and the rest will be the players. Therefore, the host will have special rights such as creating, starting, and ending the session. The players will have only access to play the game and see the results.

4. Subsystem Services

4.1. Client

Client is the first package that includes components that enables user to interact with the application. It consists of two sub-packages which are User Interface and Helper.

4.1.1. User Interface

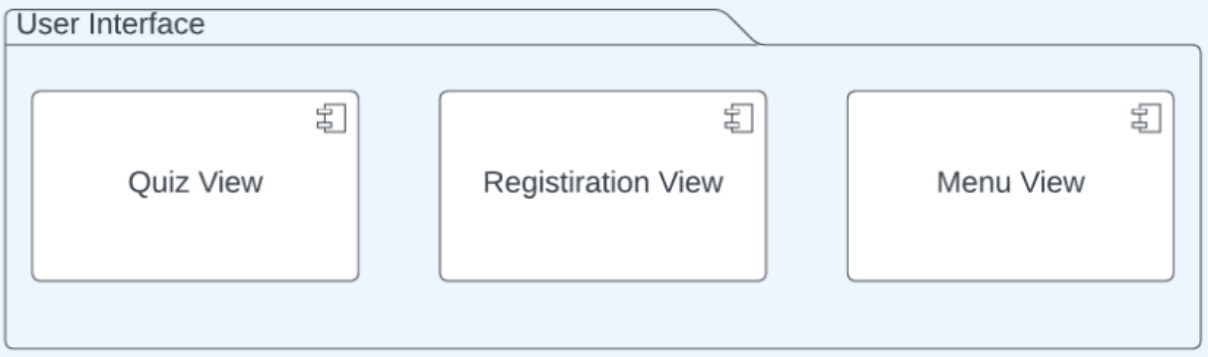

Figure 3: User Interface Module of Mutrivia

User interface is the part where the user will interact with the application. The user interface component contains the screens that are explained below:

- **Quiz View:** This view contains the quiz competition that includes questions and answers. It also contains the scores of each user in that competition.
- **Registration View:** This view contains the first screen that users will see to start playing the game. Before joining, users should register and enter a nickname.
- **Menu View:** This view contains the screen where the user chooses to play solo, join a session, or host a session. If a user wants to join a session, the session ID must be entered on this screen.

4.1.2. Helper

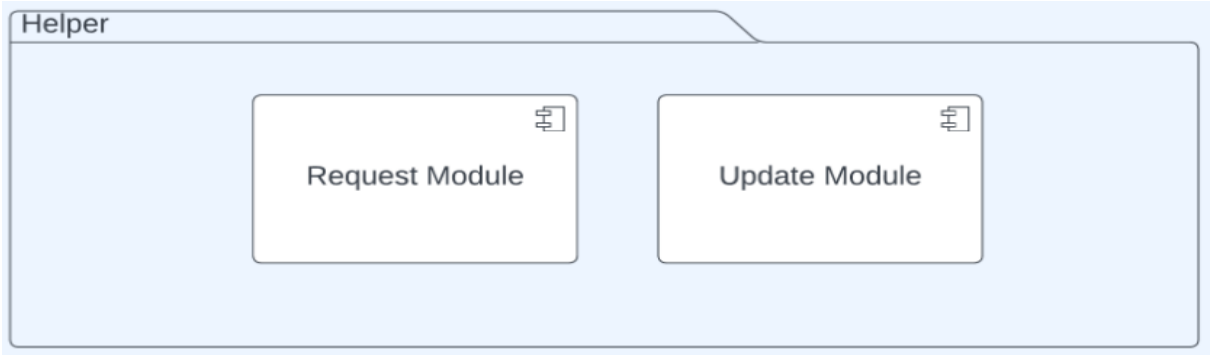

Figure 4: Helper Module of Mutrivia

Helper module functions synchronously with the user interface and is responsible for communicating with the backend service of Mutrivia.

- **Request Module:** User-generated requests will be handled by this module.
- **Update Module:** Backend-generated updates will be handled by this module.

4.2. AWS

AWS is the second package of our system. There are three sub-packages which are Backend Service, AI Service, and Persistent Data Management.

4.2.1. Backend Service

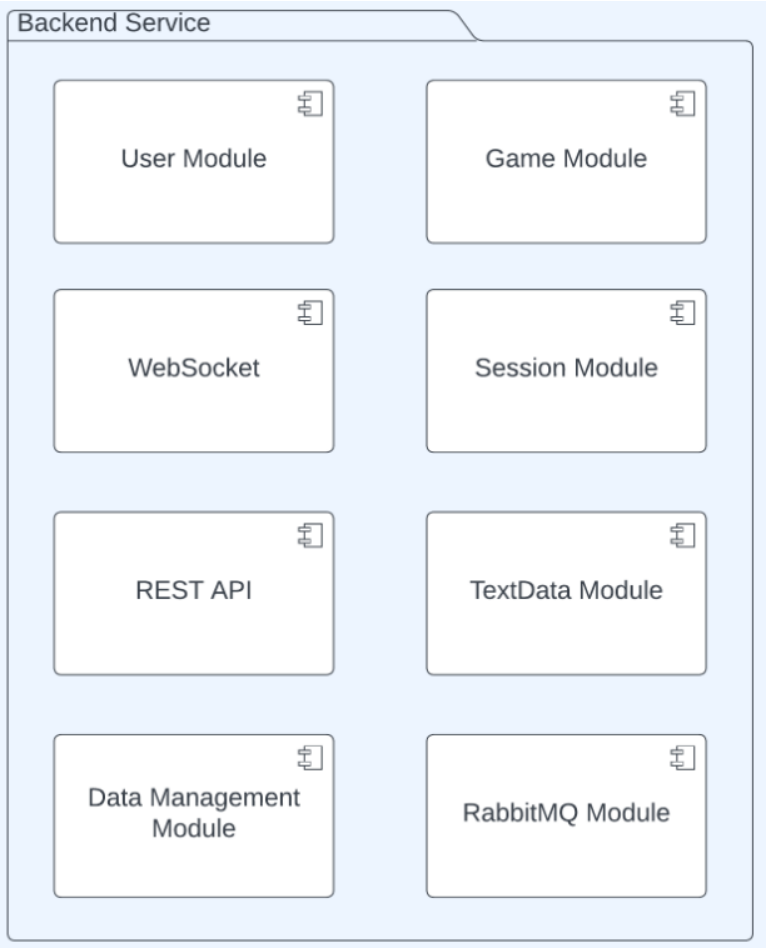

Figure 5: Backend Service of Mutrivia

Backend Service is the part where all business logic is established. It contains several modules that mostly establish connections between the Client package and other sub-packages. All sub-packages are explained below:

- **User Module:** This module contains the implementation of operations related to users, such as registration, retrieving users, deleting users, and managing user points.
- **Session Module:** Operations related to sessions, such as creating a session, retrieving session information, and deleting sessions, are implemented in this module.
- **Game Module:** This module contains the implementation of operations used for the quiz game, such as triggering question generation, starting the quiz, ending the quiz, and so on.

- **TextData Module:** This module contains implementation related to managing information texts from museums.
- **RabbitMQ Module:** This module is responsible for using a message queue to send information texts and receive the generated questions from the AI Service.
- **WebSocket:** WebSocket is used for updating the user interface without the user sending a request.
- **REST API:** It is utilized to handle HTTP requests sent by the user interface.
- **Data Management Module:** This module is responsible for data management and communicating with the database.

4.2.2. AI Service

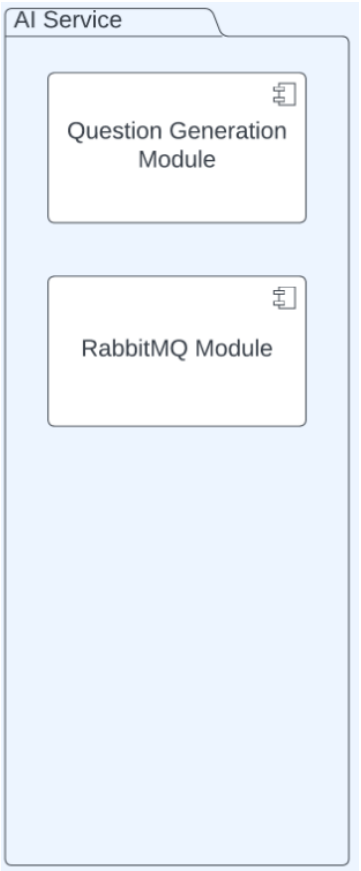

Figure 6: AI Service of Mutrivia

AI module of Mutrivia handles the question generation functionalities of Mutrivia. After it receives the text that needs a question from the backend service, it creates as many

multiple-choice questions as it can and returns the questions to the backend service.

- **Question Generation Module:** The question generation AI will be up and running in this module. The consumed messages from the RabbitMQ module will be sent to this module, and this module will use NLP technology to create questions and send them back to the RabbitMQ module for it to publish the questions back.
- **RabbitMQ Module**: RabbitMQ module, as the name implies, is a message broker module. It uses RabbitMQ for its message queue. Every time the AI module needs to consume messages from the backend service, it arrives at the RabbitMQ module first. In the same way, every time that an AI module creates questions and needs to publish the messages to the backend service, it also uses this module for this particular process.

4.2.3. Persistent Data Management

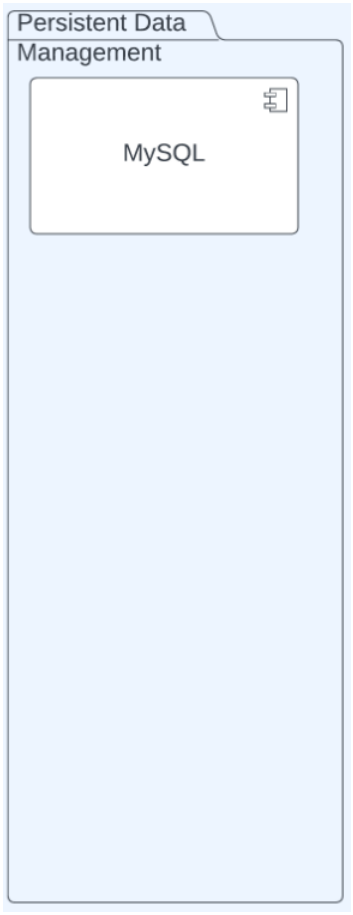

Figure 7: Persistent Data Management Service of Mutrivia

The stored data of the whole project system will be handled by this subpackage.

- **MySQL Module:** A MySQL database that will store museum, session, and user information. The Data Management module from Backend Service will have permission to access the database only.

5. Test Cases

Test ID	Test Type/Category	Summary/Title	Procedure	Expected Results	Priority/ Severity	Date Tested/ Test Results
1.1	Functional	Score increases when you answer correctly on the quiz page	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. The score of the user has checked whether it increased or not.	The score is increased after the right answer is chosen	Major	
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	
1.3	Functional	Score changes according if the question is answered correctly depending on the time left	Session is started by the host. The user is shown a question and the answer is selected right. The score is	Score is computed according to the remaining time	Major	

			checked if it is computed according to the 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	
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	
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	The Score is the same after wrong answers are chosen	Major	

			whether it increased or not.			
1.7	Functional	All question choices can be clicked only once	Session is started 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.	The buttons are unavailable	Major	
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	
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	
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	All users in the session started their quiz	Major	

			button. All users in the session check whether their quiz started or not.			
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 is based on the location.	The museum shown is based on the location	Major	
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	
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	
1.14	Functional	Checking when the session end, new scores are added	User opened the app. Username is asked for by	Users can see the scores on leader board	Major	

		to the leaderboard if there are any all-time high scores.	the app and entered by the user. Checks whether users can see the scores in the leaderboard.			
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 scoreboard.	Attendees are able to see their session scoreboard.	Major	
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	
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.	Moderate	

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	
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	
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 score while playing.	The score is shown to the user while playing.	Major	
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	The leaderboard is shown with the user's place.	Major	

			users can see the session leaderboard with his name included if he is a high scorer.			
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	
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	
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	
1.25	Functional	User can change music setting to on and off via the	User opens the app. Enters the settings menu	Music setting is modified	Moderate	

		setting menu	on the top right of the screen. Modifies the music setting. Checks whether it is changed to choice.			
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	
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	
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	

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	
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	
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	Moderate	

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	
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	
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	

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 staying away for a long time to play the game	Moderate	
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	
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	

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	
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	
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	
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	
4.42	Performance	Showing scoreboard less than 2 seconds	User will click the show scoreboard button.	Scoreboard should be displayed in less than 2 seconds.	Moderate	
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	

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	
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	
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	
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	
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	
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	
8.50	Installation	App should be available to be installed on Google Play Store	User will try to install the app on Play Store	App should be installed	Moderate	

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	
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	

Table 1: Table of test cases

6. 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.

6.1. Public Health

Mutrivia is an application aiming to enhance the museum trip experience of 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.

6.2. Safety

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

6.3. Welfare

Mutrivia is an application aiming 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.

6.4. 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. The application 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.

6.5. 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 application 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.

6.6. Social

In the app, 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.

6.7. 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.

6.8. Economic

The application 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. Teamwork Details

7.1. Contributing and Functioning Effectively on Team

We mostly distribute the work equally among each other while considering the load we have in school as well. Most of us have almost the same 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 detailed design report. After finishing our own part, we always ask for 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. Frontend development is performed by Gökhan Taş and Lara Fenercioğlu. Database implementation and integrations are done by Zeynep Ziyagil. Back-end development is done by Sebahattin Utku Sezer and Bedirhan Sakinoğlu. All the members of the project contribute to the project demo.

7.2. Helping 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. Taking a Lead Role and Sharing Leadership on the Team

Team of Mutrivia does not have a specific leader who leads the whole team. We mostly 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.

8. 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.

9. 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].