Project plan

Quiz Game

Individual Project

Fontys, Eindhoven

Date	**:**	**22.03.23**
Version	**:**	**1.5**
Status	**:**	**For review – Sprint 2**
Author	**:**	**Veronika Valeva**

Version

Version	Date	Author	Amendments	Status
1.0	08.03.23	Veronika Valeva	Created Document	Draft – Sprint 1
1.1	16.03.23	Veronika Valeva	Refactored the research questions after a conversation with Timo Hermans.	Draft – Sprint 1
1.2	17.03.23	Veronika Valeva	The project context and research questions were changed, and also specified the right research methods for every research question after Monique Vissers' feedback.	Draft – Sprint 1
1.3	19.03. 23	Veronika Valeva	Added Business Processes, Data Architecture, Architectural Principles and Non-Functional Requirements sections to the plan.	For review – Sprint 1
1.4	22.03.23	Veronika Valeva	Added context to the FR's and NFR's. Changed the goal of the project from passing the semester – to a more business-oriented perspective. Changed the wording of the research questions.	For review – Sprint 2

Contents

1. Project Assignment

1.1 Context

This individual project is being carried out for a Fontys' Advanced Software Semester 6. The focus of this project is exploring one or more topics by defining research goals, performing research activities, and applying the gained knowledge to the project.

For the case of my individual project, I chose to develop a quiz game. While the project is done for the purposes of a university project, the quiz game can also be considered as an effective tool for training and education purposes. There are various similar quiz game products available in the market such as Quizizz, Kahoot, and Socrative.

The quiz game lets users answer trivia questions and compare scores via a scoreboard. Users customize their session by selecting question category, difficulty, type, and number of questions. Scoreboards display overall and category scores, with one point awarded for each correct answer.

Target Audience

The target audience will be expected to be people playing the quiz by themselves. The web-application design should be user-friendly, allowing for easy game setting-up and playing, as well as accommodating various screen sizes (such as phones, tablets, and monitors) to make the game accessible to the widest possible audience.

1.2 Goal of the project

The goal of this project is to create a fun and engaging trivia quiz game that will entertain and educate players of all ages. The game will feature a variety of categories and questions that will challenge players' knowledge and keep them coming back for more. The game will also be easy to use and accessible on multiple platforms, including desktop and mobile devices. Ultimately, the goal is to create a high-quality trivia quiz game that will become a go-to source of entertainment for trivia lovers everywhere.

1.3 The assignment

The assignment is to develop a quiz game, which aims to demonstrate the ability to produce scalable software independently. The requirements are to develop a functional quiz game that showcases good architectural planning, technical knowledge, and capabilities.

1.4 Functional Requirements

The user should be able to create a quiz session by selecting options such as the number of questions, question category, question difficulty, and question type.

Context: After logging in the user is going to be able to create a quiz session for himself. The quiz session is created according to a set of options. The set of options consists of number of questions, question category, question difficulty, question difficulty and question type. Number of questions are going to be between 10 (minimum) and 50 (maximum). The question categories are going to be, but not limited to Arts & Literature, Movies & TV, Food & Drink, General Knowledge, Geography, History, Music, Science, Society & Culture and Sport & Leisure. The question difficulty being easy, medium, and hard. The question type being either multiple choice (where a question has several answers and one of them is correct) or true/ false (where the question has two answers – "true" or "false" and one of them is correct.)

The quiz game should have access to a bank of questions with different categories.

Context: The questions are going to be generated before the start of every quiz session. The generation of questions will be initiated with gathering Q&A data from a database with Q&A data. The categories that split the Q&A data are the same as the ones mention in the previous functional requirement.

The quiz game should have a scoring system where the user earns one point for every correctly answered question in a specific category. If the answer is incorrect, the user should not receive a point.

Context: During the gameplay, the user answers questions. Whether the users answer the questions correctly or not is taken into account. If the answers is answered correctly the score of the player is increased by one point. Data about how the user answer the questions is saved in order to build components like scoreboard and game statistics. If the answer is answered wrong then the score of the user does not change.

The quiz game should allow users to generate a quiz session, answers the generated questions and save their score progress.

Context: After the user completes (answers all of the questions from) the quiz. The score he/she made during the game is saved. Data about their generated game, their played game, their right and wrong answers, and their score is saved. Later this data will be used for populating the scoreboard components and the game statistics.

The quiz game should be able to save users information – including game preferences.

Context: When a user is registered his data is saved. The data saved being basic account information like username, password (not in raw form), email, gender. When setting up the game session the user will be able to build the quiz games according to his/her preferences. The user will be able to save those preferences so that they could easily create the quiz again using the same options. The game preferences will consist of pre-defined quiz options (number of questions, Q&A category etc.).

The quiz game should have a scoreboard that displays the overall scores of all users for all categories and a scoreboard for each question category.

Context: After logging in user will be able to see his score (considering he/she had played at least one game) compared to the scores of the other players (considering there is at least one other player). The score will be displayed as an overview of the scores for all questions (no matter question category, type, or any other Q&A option), and the scores are also going to be able to be divided in a specific category – meaning the user can choose a specific Q&A category and see his score and compare with other users for a specific question category (considering both the user and at least one other user have played a whole quiz game in that specific chosen Q&A category).

The quiz game should be accessible to users on various devices such as phones, tablets, and computers.

Context: The user should be able to interact with the quiz on different devices like a standard computer monitor, phone, or tablet. This would mean that the user interface should be scalable to multiple screens. The user should be able to easily perform every function provided by the system on any device. Every component of the user interface should be responsive. The web-app should be accessible by multiple browsers as well.

The quiz game should have measures in place to ensure the security of user data and prevent unauthorized access to the game.

Context: The user will have to register/login in order to participate in any quiz game. There will be an authentication system implemented in the quiz game. The need for an authentication in this particular quiz game is to personalize the user's experience and to protect their data. Having scoreboards is based on users having protected access to the game. Having an authentication system would ensure that users would be able to access only their own data and do that only after they have authenticated themselves.

The quiz game should provide feedback to users on their performance, such as correct and incorrect answers, and overall score.

Context: After logging in and having played at least one game the user is going to be able to see statistics about their played games. Those statistics would consist of amount of played games, amount of played games in a particular category, amount of correct/ and wrong answers given.

1.5 Non-Functional Requirements

The quiz game should have a deployment pipeline for automated Continuous Integration and Continuous Deployment to ensure the game is always up-to-date and free of errors.

Context: The quiz game is going to be spread upon three environments. One would be for development, the other would be used for testing, and the third one would be used for as production. For this particular project integration environment will not be needed as there won't be multiple active branches at one time. Also staging environment is not going to be made use of as There would be no need at this point of the project to make a simulation of the production environment. A deployment pipeline ensures that every change made to the game is thoroughly tested and deployed in a consistent manner. This means that users will have a consistent and reliable experience when playing the game, without encountering any unexpected errors. As the user base grows, it becomes more important to have an automated deployment pipeline in place to handle the increased load. By implementing a deployment pipeline early on, it will be easier to scale up the game in the future.

The quiz game should be designed to handle a large number of concurrent users, with the ability to scale up or down based on demand.

Context: The quiz game is likely to have a large number of users playing at the same time. If the game cannot handle this level of traffic, users may experience slow loading times, delayed responses, all leading to poor user experience. By designing the game to handle a large number of users playing the quiz at the same time, developers can ensure that the game remains responsive and enjoyable for everyone.

The quiz game should be secure, with appropriate measures in place to protect user data, prevent cheating, and prevent unauthorized access to the game.

Context: The quiz game is going to have access to and save user's data therefore it is important to implement security measures to protect this data from unauthorized access

The quiz game should be easy to maintain and update, with clear documentation and well-structured code.

Context: As the quiz game grows in popularity, it may require new features or infrastructure changes. Clear documentation and well-structured code make it easier to scale up the game and make changes without introducing new bugs or errors.

1.6 Business Processes

The business processes that will be implemented in the architecture of the quiz game have already been decided. These processes include game design, Q&A data generation, user registration, user authentication, game play. The game design process involves creating quiz sessions using a set of different options. The content creation involves the handling of questions and answers data. User registration is the process of allowing users to sign up for the game, while user authentication verifies their identity and authorizes them to have a personalized experience playing the game as well as to see and compare their game score. Game play involves answering questions, tracking scores, and providing feedback.

1.7 Data Architecture

It is essential to consider the data architecture that will be required for the game. The quiz game will generate and consume data such as user data, quiz responses, and game analytics. User data comprises the user's name, age, gender, and email address, and game preferences which will be utilized to create a user account and personalize the user's experience. Quiz responses encompass the user's answers to each quiz question, including both correct and incorrect responses, to calculate the user's score and provide feedback on their performance. Game analytics encompass data on user engagement and performance, such as the number of quizzes taken, the average score, and the time spent on each quiz. To manage this data, storage and management need to be considered. Firstly, it needs to be determined how the data will be stored, such as in a database or cloud storage, while also considering data privacy and security regulations. Secondly, it needs to be decided how the data will be managed, such as how it will be organized and accessed, and also consider data backup and recovery procedures.

1.8 Architectural Principles

Ensuring scalability, maintainability, security, and performance in this quiz game individual project is essential for providing a smooth and secure user experience. It also facilitates future growth and protects user data. A well-designed quiz game that can handle an increasing number of users and data without degrading performance can save time and resources in the long run.

To ensure scalability, the quiz game architecture has to be designed in such a way that handles a large volume of users and data. This might involve using a cloud-based infrastructure, such as Amazon Web Services (AWS) or Microsoft Azure, to scale the resources up or down as needed. Caching mechanisms like Redis or Memcached can also be considered to reduce database load and improve performance.

Separating different concerns, such as data storage, user interface, and business logic, would ensure maintainability and it would make it easier to maintain and modify the code over time.

To ensure security, the quiz game should be developed using secure coding practices like input validation, encryption, and access control. Tools like HTTPS (Hypertext Transfer Protocol Secure) and SSL (Secure Sockets Layer) certificates to protect data in transit should also be considered as well as secure authentication mechanisms to protect user data.

All of those decisions are yet to be made after researching every possibility.

1.9 Architecture

Front-end

The front-end of the applications will include the user interfaces that users will use to interact with the system. These interfaces have been divided into two distinct components: the game – including game creation and the game itself, and a separate UI for the scoreboard. The reasoning behind the separation of the two components is that they provide different information to the user.

The Game UI will provide features related to gameplay, while the Scoreboard UI will focus on displaying the player's scores and rankings. Keeping the scoreboard in a separate service allows us to scale and maintain it independently from the rest of the game UI. It also helps to ensure that the scoreboard is always available, even if there are issues with other parts of the game UI. This separation of concerns makes it easier to develop and maintain each part of the game, ultimately leading to a better user experience.

Game UI

The Game UI will be an interface used by the users when they authenticate themselves and enter the game. The players are going to be able to set their game preferences and start the game. The technology that is going to be used for this component is yet to be decided.

Scoreboard UI

The Game UI will be an interface used by the users to see their score in all or in a specific question category. Every user will be able to compare his/her score to the ones of the other players. This will be happening in real time and due to this the UI will be rapidly changing every time a player's score changes (gains point). The technology that is going to be used for this component is yet to be decided.

Back-end

The game system is going to be divided into separate services, including a Game Logic Service, Questions & Answer Generator, Scoreboard Service, and Authentication & Authorization Service. This will make the system easier to manage, more secure, and better suited to handle a large number of users. By dividing the system into separate services, each part can be updated or changed without affecting the others. This will make it easier to fix problems and add new features. Thus, making possible for users to play the quiz game even if the scoreboard service is down. The same goes for the situation where the game logic service goes down (or is under construction) – still the users can use the scoreboard. Overall, splitting the game system into separate parts is a good idea because it will make the system work better and be easier to use.

Game Logic

This back-end server will be responsible for managing the game logic. It will be responsible for the generation of quizzes, the actual game, the answering of questions, and keeping the users' preferences – part of the quiz generation. Initially, the players will play by themselves, however, the server would allow the game logic to expand to multiple people playing together. This would require a real time connection between the back-end and the front-end. For this Client-Server communication the usage of SignalR might be considered. ASP.NET will be used as this back-end framework.

Scoreboard

This back-end server will be responsible for managing the scoreboard logic. It will be responsible for the generation scoreboard information based on different options. The user will be able to see the constantly growing scores of the active players. This will require a real-time connection between the back-end and the front-end. SignalR provides real-time communication between the server and clients, allowing for instant updates to the scoreboard as the game progresses. SignalR can handle large numbers of simultaneous connections, making it suitable for (multiplayer) games with many players. SignalR is tightly integrated with ASP.NET thus ASP.NET is going to be used to build the Scoreboard back-end.

Questions & Answer Generator

The Questions & Answers Generator will be a back-end server that gathers questions and answers of a specific set of options. Instead of using a custom-made database filled with questions and answers, "The Trivia API" is going to be used. The trivia API can be updated frequently, providing the game with the latest and most relevant questions. In contrast, maintaining a database of questions requires constant updating and management. "The Trivia API" provides easy integration through a REST-full API. The API and all data returned from "The Trivia API" are licensed under a Creative Commons Attribution-NonCommercial 4.0 International License. This means that the API and the questions it returns are free for non-commercial use – which fits the scope of this project.

FastAPI will be used as a middleware service for separating concerns between the Game Logic and "The Trivia API". FastAPI is a framework optimized for high performance and speed, making it a great choice for building a service that needs to communicate with a third-party API like "The Trivia API".

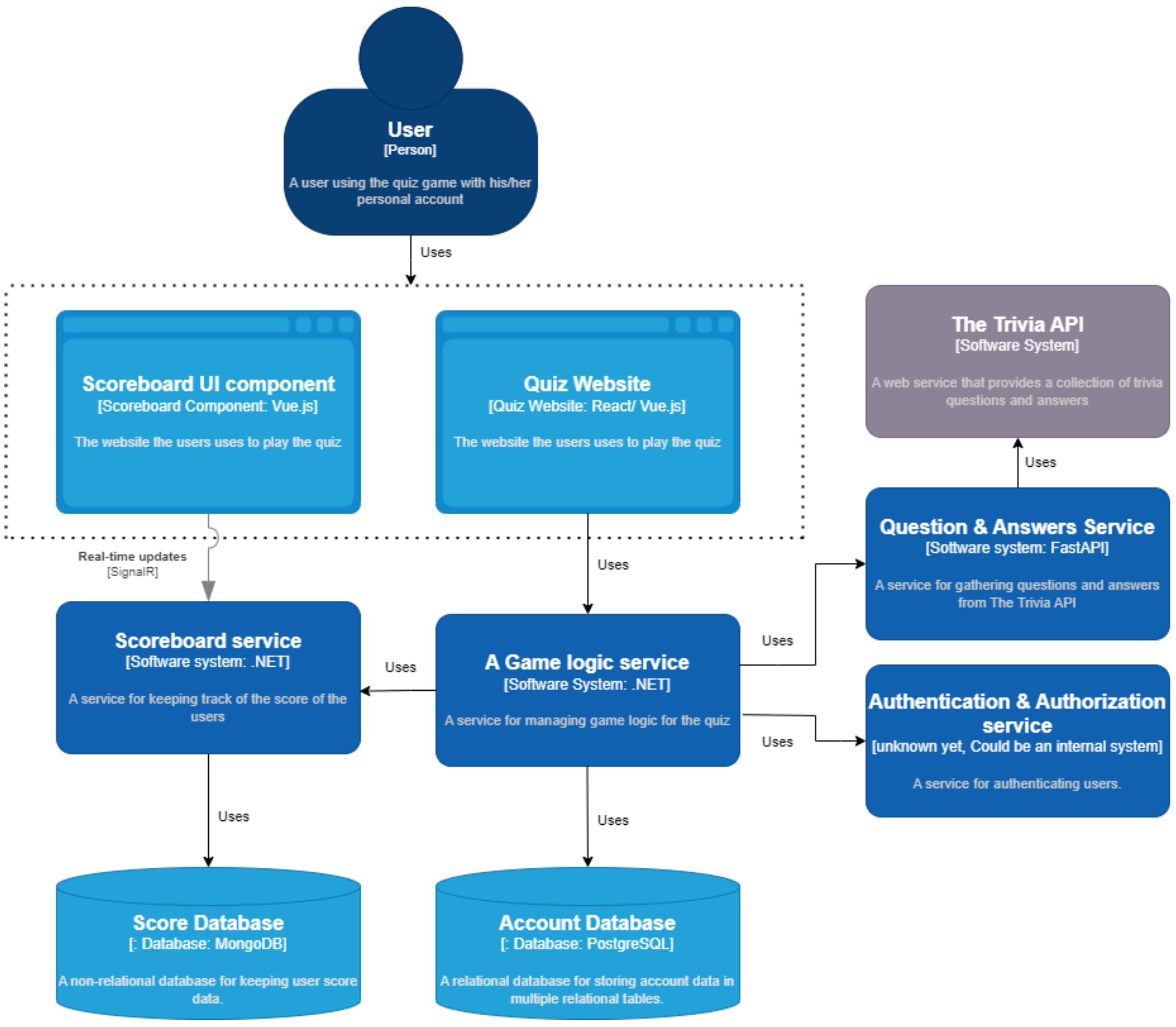
Authentication & Authorization

Users are going to have to authenticate themselves before being authorized to play the quiz. By having users login to a game, the game can store user data and preferences, such as their progress in the game, achievements, and settings, and also ensures the security of the game and the user's personal data. It is to be decided how the authentication & authorization is going to be implemented.

Data storing

I have decided to split the game data storing into two separate databases: an Account Database and a Scoreboard NoSQL Database. This decision was made to improve the overall performance and scalability of the game. The Account Database will store user account information, such as usernames and passwords, game preferences, while the Scoreboard NoSQL Database will store game scores and rankings. Keeping these two types of data separate, ensures that there are no performance issues that may impact the game experience. Additionally, using a NoSQL database for the scoreboard data allows for greater flexibility in handling large amounts of data, making it easier to scale and maintain the database as the game grows in popularity.

Diagram


User
[Person]
A user using the quiz game with his/her personal account
Uses
Scoreboard UI component
[Scoreboard Component: Vue.js]
The website the users uses to play the quiz
Quiz Website
[Quiz Website: React/ Vue.js]
The website the users uses to play the quiz
The Trivia API
[Software System]
A web service that provides a collection of trivia questions and answers
Uses
Question & Answers Service
[Software system: FastAPI]
A service for gathering questions and answers from The Trivia API
Real-time updates
[SignalR]
Uses
Scoreboard service
[Software system: .NET]
A service for keeping track of the score of the users
Uses
A Game logic service
[Software System: .NET]
A service for managing game logic for the quiz
Uses
Uses
Authentication & Authorization service
[unknown yet, Could be an internal system]
A service for authenticating users.
Uses
Uses
Score Database
[: Database: MongoDB]
A non-relational database for keeping user score data.
Account Database
[: Database: PostgreSQL]
A relational database for storing account data in multiple relational tables.

1.10 Research questions

How to build a secure quiz game system that can support growing amounts of data and can be expanded in the future?

Sub-questions

1. What makes up a scalable architecture?
2. What is the best way to keep the different aspects of the system separated and why?
3. What is the best way to structure the system architecture to best fit the quiz game project?

4. Which technology would be most appropriate to align with the requirements of the project?
5. What are the different ways to generate questions and answers data, and which one fits the project?
6. What authentication system would best fit the project?
7. Is data storage required for this project and if so, what are data are they going to store what are the requirements for them?
8. How to keep real-time connection and show continuously changing data without the need of constantly refreshing the scoreboard user interface?
9. What are the essential software environments required for setting up this project?
10. What would be the optimal approach for communication between the different components of the application?
11. What are the best practices for implementing and maintaining a CI/CD pipeline in a software system, and how can these practices be adapted to fit the specific needs of this project?

1.11 Research methods

For research conducted during this project, the DOT framework will be used, while all 5 research strategies will be utilized, the primary focus will be on Library and Field research as a large part of the assignment is something I am doing for the first time. Other than these research methods, the approach will be centred on experimentation and testing before integrating every idea into the project.

- Use library and internet research to identify factors that make an application fit the scope of enterprise architecture.
- Experiment with different approaches to keep different aspects of the system separated and find options for structuring the system architecture for the quiz game project.
- Perform library and internet research to identify different technologies that could potentially meet the project's requirements.
- Perform library and internet research to identify different methods for generating question and answer data to choose the most suitable one.
- Perform library and internet research to find an authentication system that best fits the project's requirements.
- Experiment with different storage options and determine if data storage is necessary for the project.
- Experiment with different approaches to achieve real-time connection and data updates on the scoreboard user interface to find the most effective one.
- Perform library and internet research to identify essential software environments required for setting up the project by reviewing academic literature and industry best practices.
- Experiment with different approaches to communication between different components of the application to find the most effective one.
- Perform library and internet research to find best practices for implementing and maintaining a CI/CD pipeline and adapt them to fit the project's specific needs.

1.12 Research results

The results from every research are going to be applied to the project. The main driver of every research is going to be gathering new knowledge in order to move the project forward. Each research question adds a valuable insights and guidance to the project. For example, which technology to use for each service can help in selecting the appropriate technology stack. How to build a Question & Answer service can provide guidance on designing and implementing this service. What authentication system to use can help in selecting a secure and efficient system. Whether databases are required, and what data they will store, can assist in designing the database system. How to implement a real-time Scoreboard Service can provide information on displaying continuously changing data. The essential software environments needed can help identify the software requirements. The optimal approach for communication between the different components can ensure efficient communication. Lastly, best practices for CI/CD pipeline implementation can guide in the implementation of an efficient and reliable pipeline. In summary, the answers to these research questions can help in making better decisions and designing an effective project.