Project Plan

VocabVersus[[1]](#footnote-1)

Thomas van der Molen

|  |  |
| --- | --- |
| **Project Information** | |
| Project members | Thomas van der Molen |
| Project Name | VocabVersus |
| Version | 1.1 |

|  |  |  |
| --- | --- | --- |
| **Version** | **Date** | **Change** |
| 1.0 | 12-02-2023 | Created Document |
| 1.1 | 16-02-2023 | Changed Project |

Table of Contents

[Prerequisite 3](#_Toc127454621)

[Function Learning Outcomes 3](#_Toc127454622)

[Introduction 4](#_Toc127454623)

[Similar Contexts 4](#_Toc127454624)

[Target Audience 4](#_Toc127454625)

[Functional Components 4](#_Toc127454626)

[Project 5](#_Toc127454627)

[Considerations 5](#_Toc127454628)

[Architectural Decisions 6](#_Toc127454629)

[Game 6](#_Toc127454630)

[Data 6](#_Toc127454631)

[User 6](#_Toc127454632)

[Decisions 6](#_Toc127454633)

[Architecture 7](#_Toc127454634)

# Prerequisite

While the project is done for educational purposes, it is also meant to prove the learning outcomes set by Fontys for Advanced Software semester 6.

The setup and execution of this project will be heavily driven by the learning outcomes set by Fontys.

## Function Learning Outcomes

For Advanced Software semester 6, there are 9 Learning Outcomes, below is a list of the learning outcomes that can be functionally proven in the process of this project:

**Learning outcome 1 – Future-oriented Organization**

Develop and deploy scalable software in accordance to the project goals, and design such solution with the ability for future further development.

**Learning outcome 2 – Investigative Problem-Solving**

Deliver professional products based on a structured problem-solving and methodical planning in a critical/professional manner.

**Learning outcome 5 – Scalable Architectures**

Develop architecture of scalable software, considering attributes related to enterprise contexts with high volume data and/or events.

The architecture should also be future expandable and allow for independent monitoring and deployment.

**Learning outcome 6 – Development and Operations (DevOps)**

Set up software development environments allowing for as much automation as possible, enabling short release times and high software quality.

**Learning outcome 7 – Cloud Services**

Integrate cloud services and deploy (parts of) an application to a cloud platform.

**Learning outcome 8 – Security by Design**

Incorporate best practices and minimize security risks.

**Learning outcome 9 – Distributed Data**

Consider legal and ethical issues alongside specific data requirements for enterprise systems.

# Introduction

VocabVersus is a web-based multiplayer vocabulary game, in this game multiple players will compete to think of a word containing given letters. Points will be given based on player’s speed and complexity of the word given.

User’s will have the ability to play in a competitive environment to try and become the top player on the leaderboards, or create their own game types (word lists, game rules) and play for fun.

The game’s point distribution will change dynamically based on the habits of players, for example: Words used very often will become worth less points.

## Similar Contexts

Similar games to this already exist such as [Scrabble](https://playscrabble.com/) or [Wordle](https://www.nytimes.com/games/wordle/index.html).

## Target Audience

The target audience will be expected to be people playing for a short period (e.g. while waiting for the train or on the toilet) or a group of friends wanting to play a simple party game.

To make the game as approachable as possible for these two major groups, the design of the web-application should make it deliberately easy to join a game or to create your own and invite people, along with supporting many different screen-sizes (e.g. phone, tablet, monitor) .

## Functional Components

|  |
| --- |
| **Component** |
| Webpage(s) allowing users to navigate to the different parts of the game |
| Webpage used by a player during the game |
| Game system for receiving game input and returning game information |
| Player connection system handling the communication between players and game instances |
| Word system for handling logic related to handling the letter-word functionality |
| Account system for allowing users to create and authenticate player accounts |
| Deployment pipeline for automated CI/CD |

# Project

## Considerations

As shortly discussed in the [Target Audience](#_Target_Audience), there are several considerations that are and will be made to create an optimal application, these considerations will be researched separately as architectural decisions.

After the considerations based on architectural decisions have been made, an architecture for the application can be created.

Below is a list of subjects that are of interest during this project and will have to be taken into consideration when creating the application architecture.

|  |  |
| --- | --- |
| **Considerations** | |
| **CG** | **Game** |
| CG-1 | Game-User connection |
| CG-2 | Game latency |
| CG-3 | Game events |
| CG-4 | Game instance load |
| **CD** | **Data** |
| CD-1 | Data management |
| CD-2 | Data privacy |
| CD-3 | Game data aggregation |
| CD-4 | Game event persistence |
| **CU** | **User** |
| CU-1 | Webpage design |
| CU-2 | Player engagement |

## Architectural Decisions

### Game

Players will generally only be playing a single instance of a game at once with other players, this allows player connection to be linked directly with the single game instance that is relevant to them (CG-1). By grouping player connections, instance specific events can much more easily be distributed to the relevant players (CG-2), and it allows the game load to be distributed among smaller game instances, allowing for game instances to be scaled up based on the amount of impact a game instance might have (e.g. a 4 player game vs a 100 player game) (CG-4).

During the game, several players will be playing against one-another in real-time, this makes it extremely important to reduce the latency for actions taken by players that will affect the game state of other players.

Thankfully, players will only know of other player’s actions via events inside the game, due to this a larger connection delay margin can be acceptable as long as it is not disruptive to the game work-flow, for example; slight delay margins can be added to event handling allowing player actions to still be evaluated even if it is technically outside of the relevant game state time (e.g. answers submitted at the end of a round will still be evaluated if it is within the connection delay margin) (CG-4).

### Data

A lot of persistent data from different domains should be stored for the web-application, this includes account information, game instance information and general data used for game instances. Game instance data will often have to be stored temporarily and deleted later, while aggregated game data will be mostly static after being added, with frequent evaluations being made based on the stored data (CD-1).

The data used by the game will be aggregated either by system administrators and users alike, allowing for new data sets to be created for different games, extra care will have to be taken by allowing users to input their own data (CD-3).

Some data, such as the inputs made by users during a game, or personal information such as passwords, have to be stored securely (e.g. properly hashing passwords) (CD-2).

Situations might come up where a game instance might be removed, causing any non-persistent data to be lost (e.g. connection loss causing all players to disconnect), to ensure that players are able to continue their games, important game state data should be stored in a short-term persistent manner and be able to happen asynchronously from the progress of the game (CD-4).

### User

The game is expected to be played on many different devices, such as phones, tablets and computers, this makes it important that the user interface for the game can support these different screens well to improve user experience (CU-1).

Furthermore, to help keep player engagement high, it should be made as easy as possible for users to access core features of the web-application such as joining or creating a game, and to help keep players engaged in playing a more competitive mode should be created with a public leaderboard, allowing players who prefer more challenge to also be engaged (CU-2).

### Decisions

Maybe use database functions for game data evaluation (are these asynchronous?), EFCore required synchronous data, so for persistent game data might not be the best (can be used for accounts however)

## Architecture

1. Subject to change [↑](#footnote-ref-1)