UNIVERSITY OF STAVANGER, NORWAY

DAT240 Project Report

Multiplayer Image Guessing Game

Ardijan Rexhaj, Arkadiy Stepanov, Daniel Alvsåker,
Tommy Fivelstad and Sindre Mosand

November 26, 2021

CONTENTS

1 Introduction

1.1 The Group

The group consisted of 5 people with a varied background and experience level in programming and web development. Two of the group members are working part time, with software development, while the rest are full time students.

1.2 The Game

The image guessing game created by the group generally follows the criteria set by the project description.

The game supports two modes, single-player and multi-player. The role of the proposer can be filled by the creator of the game or an oracle bot system. Multi-player games will start as soon as at least one additional player has joined and the creator decides to start the game. If a human proposer leaves a game, the oracle will automatically take over as proposer.

The game logic was built with proper abstraction in mind, thus it is not dependent on the game mode.

All images provided were categorized, and a new game uses random images from selected categories. Functionality for both manual and automatic slicing of uploaded images were implemented. Uploaded images can be placed in any category, including a 'Uploaded Images' category, and the images are tagged with the users ID.

1.3 Project Setup

For the project a VUE frontend and a C# backend was used. Continuous integration and deployment was setup with GitHub actions using a docker container and a web server.

1.4 GitHub Repository and Web Server

Links for the Github repository and website for the Image Quizzer application:

`https://github.com/dat240-2021/Alvs-kerRexhajMosandStepanovFivelstad`

`https://devmode.app/`

2 Initial Planning

An initial planning meeting was held, with all group members participating. The project description was discussed and group goals for the project were set. The group agreed to attempt to implement all the advanced functionality in the project description, keeping in mind that the some functionality could be dropped due to time constraints. Hence, all functionality and logic implemented should be done in such a way that it supported all the advanced features of the game without any large changes to the code base being needed.

During the initial meeting foundational decisions were made such as technologies, individual responsibilities and communication methods.

An initial domain design was agreed upon, as shown in figure 3.1, and a page map for the application was also created, as shown in figure 3.2.

3 Design

3.1 Design Decisions

The decision was made to generalize the core logic as much as possible resulting in simpler and more structured code. Each context was abstracted as much as possible which enables interchangeable logic and modules.

The lobby context responsibility is to handle inter-game logic meaning the creation, initiation and recording of finished games. However, due to time limitations game recording was not implemented. Once a game has been created it is yielded to the game context.

The game context enforces game logic according to the initial game settings. Once the game context receives a game it acts as it's manager while it is active, distributing events and progressing the game.

Finally, the image context does image processing such as slicing, resizing and categorizing.

3.2 Domain Driven Design

The application was built following Domain Driven Design principles. Backend and frontend logic is clearly separated. The backend code was split into 4 main domains, as can be seen in figure 3.1, with clearly defined responsibilities for each domain.

Communication between contexts is done through pipelines. Additionally, events are used to trigger context specific logic or communication.

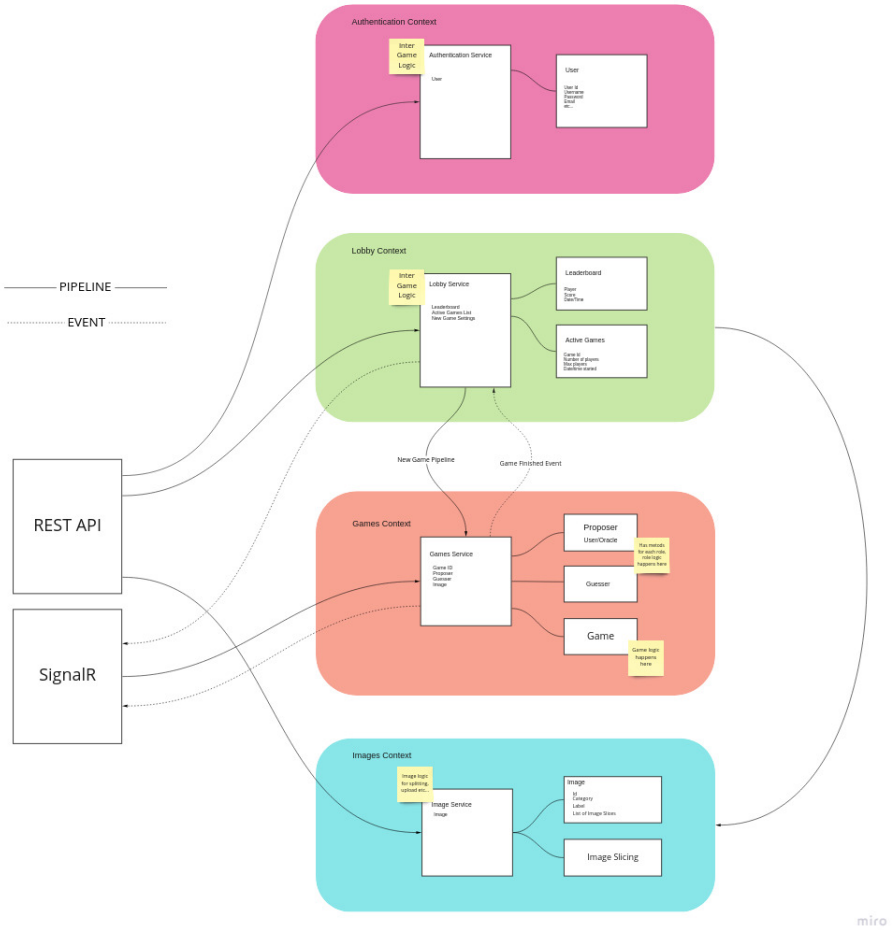

Figure 3.1: Block diagram for the domains.

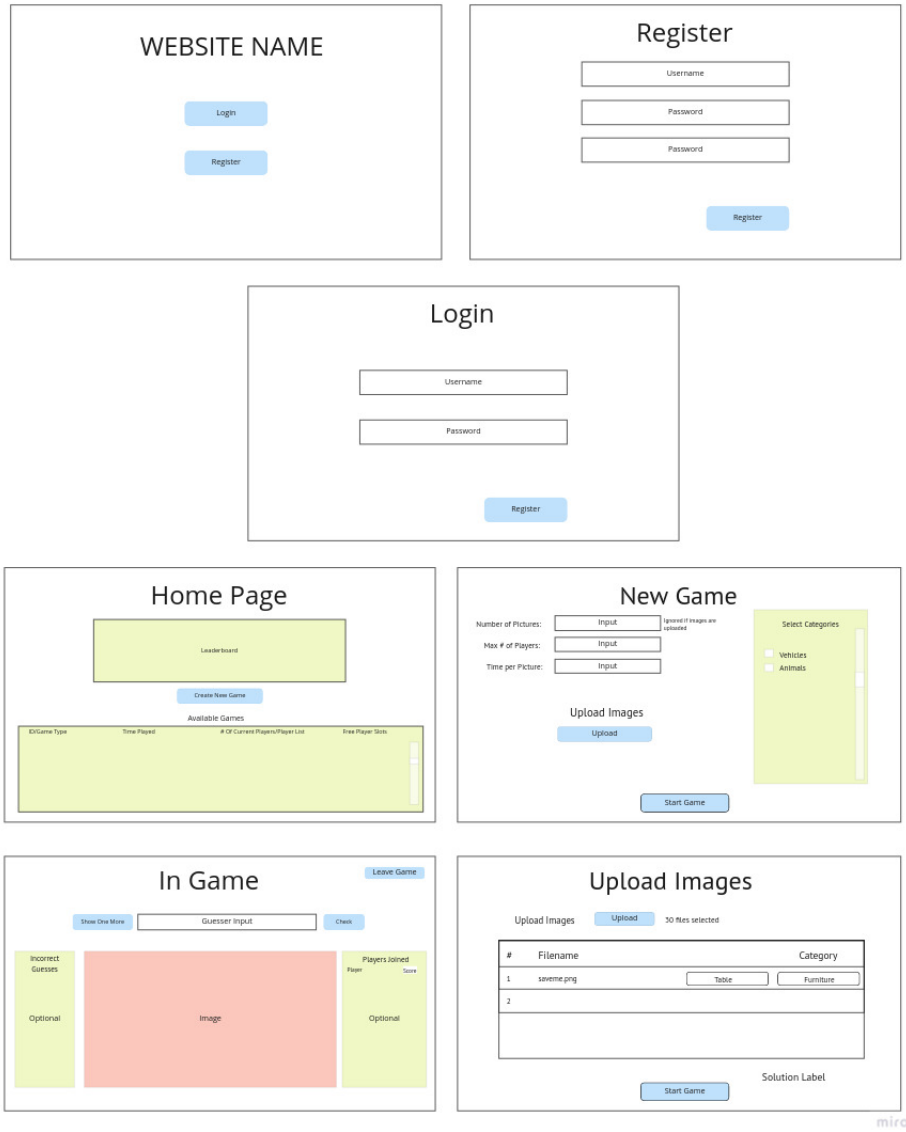

Figure 3.2: Page Map

4 Technical Decisions

The first decision that was made were that the application should have a SPA frontend with a HTTP API backend.

4.1 Frameworks and languages

For the frontend Vue and Typescript were the main technologies. The reasoning behind Vue selection was due to previous experience and easier implementation of SPA and Typescript to provide consistency with the static type nature of backend.

The .NET framework and C# were used due to familiarity, and a wide variety of packages and features. In particular SignalR, Identity, EF Core, ImageSharp and MediatR were used.

4.2 Separation of Concerns

There were several instances where decisions had to be made as to whether frontend or backend is responsible for a particular feature. The logic behind these decisions was in identifying what the project was about. The responsibility of frontend is to provide an interface for humans, while the backend's responsibility was to provide a simple API for different use cases such as AI training. Therefore the decision was made to put image selection based on mouse position in frontend, since this is a feature that is only required by a human user. On the other hand, the image uploading and slicing was done in the backend because it exposes a simple interface for any system to upload images.

4.3 Infrastructure

Communication between frontend and backend happens through an API, using HTTP and WebSockets where applicable. MediatR was used for inter context communication through events and pipelines. The database framework used was EF Core with SQLite.

4.4 Continuous integration and deployment

GitHub's branch protection features were used to protect the main branch, and GitHub actions to run tests on incoming pull requests. A PR merge would trigger automatic deployment to the server.

5 Communication and Workflow

5.1 Communication

Due to the group's additional commitments, the main communication method was a Discord channel for the group. Once a week, the entire group had the ability to meet physically at the university. Otherwise group members met at the university as frequently as possible during the project. The quality of code reviews could have been improved by reviewing changes and additions together physically.

5.2 Workflow

"Frontend", "Backend" and "Other" projects were created for the GitHub repository. Projects started off with tasks that were created and later converted to issues, but this was later changed and issues were instead created in order to be tracked on projects. This had the benefit of simplifying the workflow, while still tracking the work using labels.

The development workflow was a resemblance of a Pull-Request Trunk-Based workflow. This included:

- Creation of short-lived feature branches out of the main branch.
- Draft pull request to the main branch.
- Iterations of code review and resolution of change requests.
- Merge and deletion of the feature branch.

Three automatic tests were done, before merge into main was available for each pull request:

- A .NET build check and running of unit tests.
- Building and run the Docker container.
- Building the Vue frontend with npm and lint warnings for formatting.

6 Improvements

6.1 Testing

Domain logic was tested using unit tests and some integration tests were written but due to time constraints not all domains were covered. The ambition was to completely cover frontend and backend with end-to-end and feature testing. Testing should have a higher initial priority in potential future development.

6.2 Workflow

A trunk based workflow dictates short lived feature branches and relatively small iterations, but this was not adhered to enough initially in the development process. Later on, this workflow was improved by creating smaller and more specific pull requests and issues.

Communication was a good experience, but more frequent meetings would have prevented minor workflow issues.

6.3 Features

- More frontend polish and linting.
- Game history for each player.
- More configurable automatic image slicer, such as number of slices etc.
- Error logging.
- Error handling middleware.
- Bug fixes.
- Guess parsing.
- Game naming, and private games.
- Oracle training.

7 Summary

The project has implemented all base and advanced features as interpreted by the group. At its current state the project is stable, deployed on a server and is mobile compatible. The group has overall functioned well and overcame most difficulties encountered. Most of these difficulties were encountered when pull requests became too big, and hard to handle, the group made significant changes to the workflow by making use of draft PR's such that everyone knew what was being done, and tests were run for each commit. The end result of the project aligns well with the initial expectations of the group.