Milestone 2 Report

Software Praktikum FS 21

MAP GUESSЯ

26.03.2021

Group Members:

Jérôme Hadorn - jerome.hadorn@uzh.ch - 19-731-193 (Project Leader)

Hoàng Ben Lê Giang - hoangben.legiang@uzh.ch - 19-751-270

Claudio Gebbia - claudio.gebbia@uzh.ch - 19-712-173

David Diener - david.diener@uzh.ch - 19-733-179

Philip Giryes - philip.giryes@uzh.ch - 19-752-799

Links

Scope of Sprint 1
- Front End Sprint 1 Client - https://github.com/sopra-fs21-group-24/client/milestone/1
- Front End Sprint 1 Server - https://github.com/sopra-fs21-group-24/server/milestone/1

Diagrams
- Class Diagram
- Component Diagram
- Activity Diagram

Diagram 3 - Activity Diagram

Specification of the REST Interface
- UserController
- GameController
- LobbyController
- LeaderBoardController

User Interface Design – Mockups
- The Landing Screen
- The Registration Screen
- The Home Screen
- The Gamemode Screen
- The Create/Join Screen
- The JoinRoom Screen
- The Lobby Screen
- The Game Screen
- The Enlarged Minimap Screen
- The Round Results Screen
- The Final Results Screens
- The Edit Profile Screen
- The Error Screen

Links

Deployed WebApp - https://sopra-fs21-group-24-client.herokuapp.com/

GitHub (Server) - https://github.com/sopra-fs21-group-24/server
GitHub (Client) - https://github.com/sopra-fs21-group-24/client

SonarCloud (Server) - https://sonarcloud.io/dashboard?id=sopra-fs21-group-24_client
SonarCloud (Client) - https://sonarcloud.io/dashboard?id=sopra-fs21-group-24_server

Heroku (Server) - https://dashboard.heroku.com/apps/sopra-fs21-group-24-client
Heroku (Client) - https://dashboard.heroku.com/apps/sopra-fs21-group-24-server

Scope of Sprint 1

For the first sprint we want to have a playable game prototype. We are going to implement two of our three planned game modes (Time & Pixelation). The Cloud Gamemode will be part of our second sprint. We have also excluded the leaderboard on the homepage from our first sprint as it is not essential to the core functionality of the game experience. So our first sprint contains user authentication and management, a working game lobby, a functioning singleplayer & multiplayer game experience and the two game modes Pixelation and Time. In general, our first sprint consists of 85% of all user stories.

Front End Sprint 1 Client - https://github.com/sopra-fs21-group-24/client/milestone/1

- Excluding all issues/user stories regarding the LeaderBoardComponent on the HomeScreen
- Excluding all issues/user stories regarding the “Clouds” game mode

Front End Sprint 1 Server - https://github.com/sopra-fs21-group-24/server/milestone/1

- Excluding all issues/user stories regarding the LeaderboardController, LeaderboardService & LeaderboardRepository
- Excluding all issues/user stories regarding the “Clouds” game mode - CloudsGameMode Subclass in specific

Diagrams

Class Diagram

The class diagram was structured as predefined by our user stories. As our application is more front end orientated, our backend is modeled quite simply. The game class is the spine of our backend. All the important functionalities are implemented in there. We favored using abstract methods over interfaces for the User- and GameMode classes, as the subclasses share a lot of common functionalities.

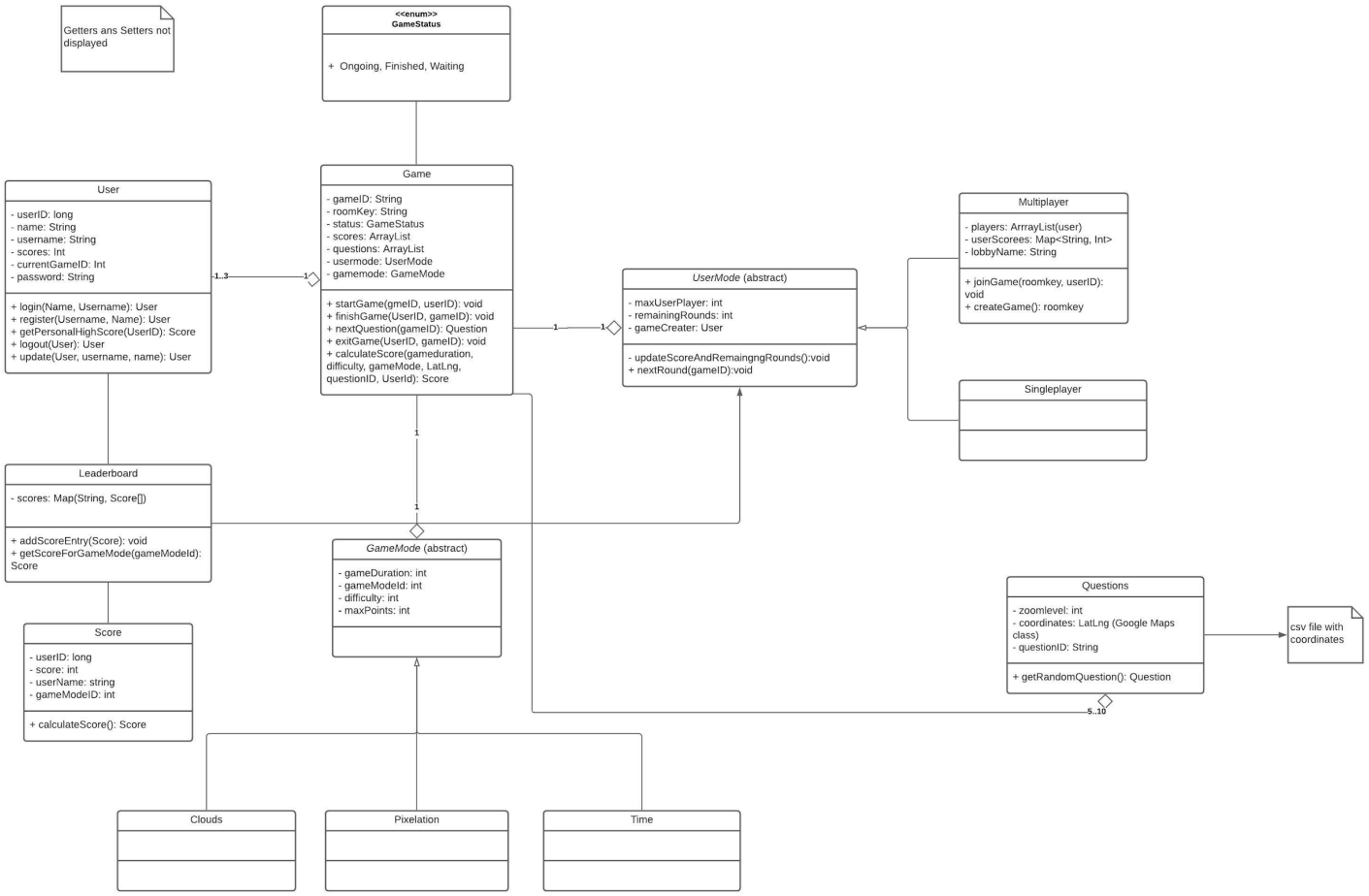

Diagram 1 - Class Diagram

Component Diagram

The component diagram consists of three main components (Client, Server & Database) that interact with each other through API calls. The user's input from the client side arrives on the server where it gets handled by the correct controller and returns the desired output. Relevant objects are stored in repositories which then are mapped by the Java Persistence API (JPA) to a relational database for persistence. Since we do not store our lobbies in a repository, the LobbyService component is responsible for sending the list of users from the lobby to the game object. The Home Screen component gets re-rendered according to its state.

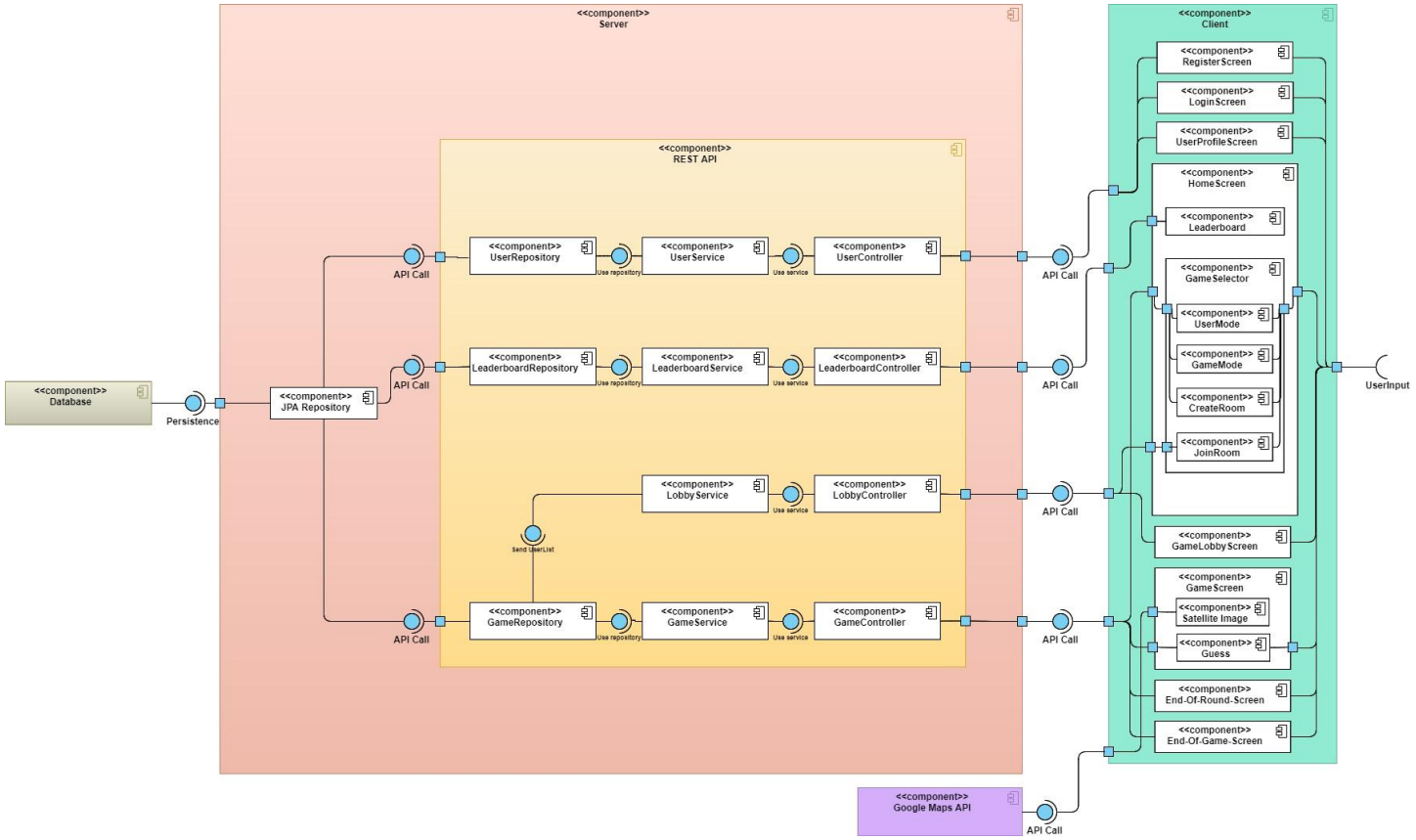

Diagram 2 - Component Diagram

Activity Diagram

For the activity diagram, we chose to show one multiplayer run since the single-player mode is just a simplified version of the multiplayer mode. This diagram is supposed to show the basic flow of the application on the client as well as the server side. Starting with the login process of a user, continuing with setting up a game, playing it and ending after answering all the questions given. This should give you a basic understanding of the way the application works in regards to client, server interactions.

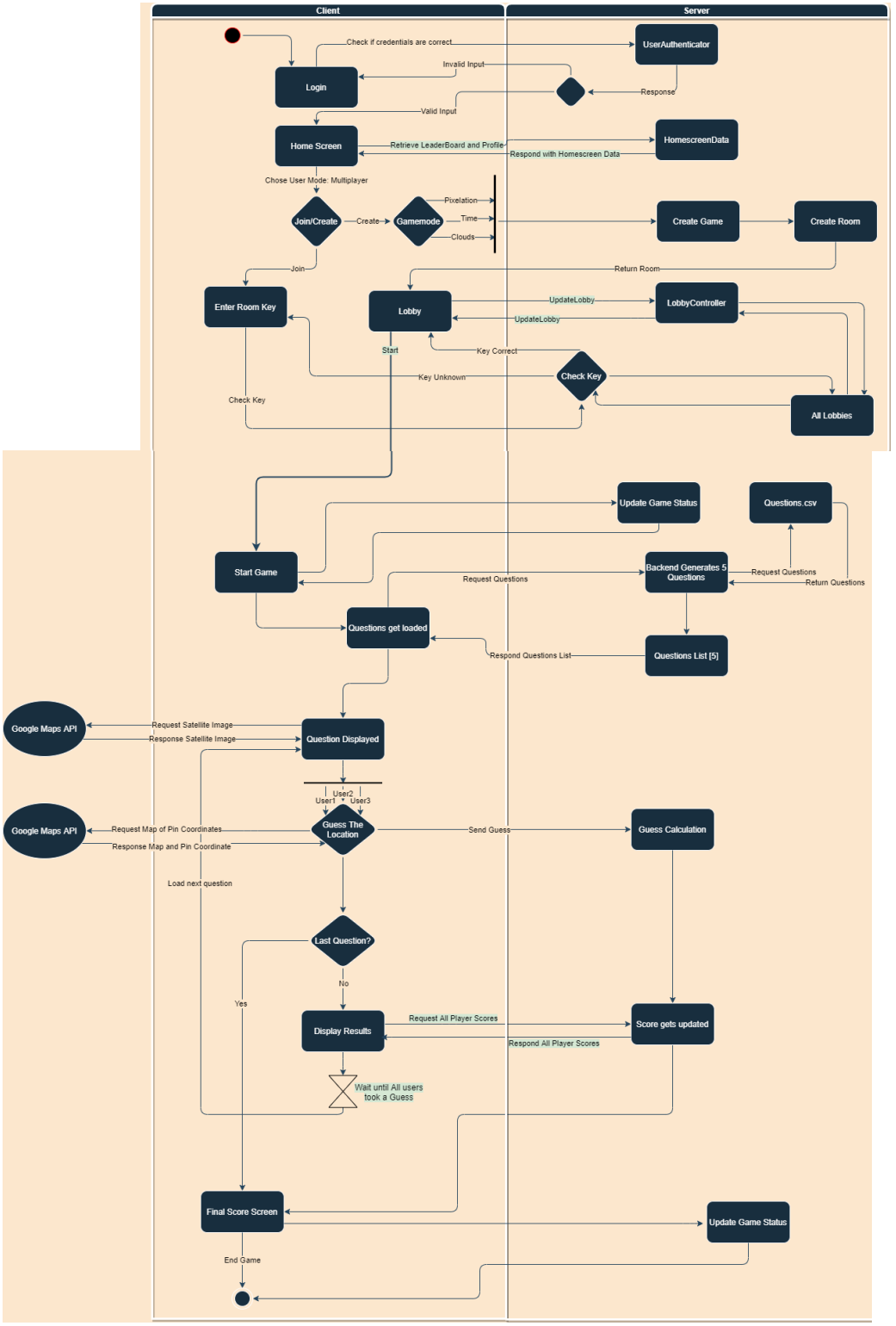

Diagram 3 - Activity Diagram

Specification of the REST Interface

The tables 2-5 outline the REST specifications for our application. Each individual table covers the endpoints for one APIController. For our use case we choose to not use HATEOAS principles. We've added an additional Table Legend that should help to simplify the authentication requirements for the different types of requests.

Authentication Type	Explanation
-	No Authentication needed - any user browsing the web
Authenticated User	Requires the userToken of a registered user
Matching Authentication (<Role>)	Requires the userToken of the user declared in the <Role> or to be contained a list of <Roles>

Table 1 - Authentication Types Legend

When a request's authentication type is "Authenticated User" or "Matching Authentication (<Role>)", it is implied that the userToken of the requesting user is sent as part of the request header to the server where it will be used for authentication checks.

UserController

This controller is here to handle the user authentication and management. It provides proper login, register & logout methods to gain access to the game and then allows users to fetch data of other users and update their own data.

Mapping	Method	Parameter	Parameter Type	Authentication	Status Code	Returned Value	Description
/users/register	POST	username <string>, password <string>	Body	-	201	token <string>	Creates a new user and return a user token for authentication
					409	Error: reason <string>	Creating user failed because of empty username or password
					409	Error: reason <string>	Creating user failed because of non unique username
/users/login	POST	username <string>, password <string>	Body	-	200	token <string>	Authentication succeeded return user token for authentication
					403	Error: reason <string>	Login failed due to incorrect password
					403	Error: reason <string>	Login failed due to incorrect password & username combination or non existent username
/users/logout	PUT	-	Body	Matching Authentication (User)	204	-	User was logged out - User is determined through the userToken in the request header
					400	Error: reason <string>	User couldn't be logged out
/users/{userId}	GET	userId <long>	Query	Any Authenticated User	200	User	retrieve user profile by userId
					400	Error: reason <string>	retrieve user profile by userId failed
					403	Error: reason <string>	no userToken of a registered user was provided in the header
/users/{userId}	PUT	User	Body	Matching Authentication (User)	204	-	User was updated
					400	Error: reason <string>	User couldn't be updated
					403	Error: reason <string>	userToken provided isn't belonging to the user you're trying to update

Table 2 - UserController Rest Specifications

GameController

The GameController handles the entire setup flow leading up to the game, the game itself with all the scoring, questions fetching and game status operations.

Mapping	Method	Parameter	Parameter Type	Authentication	Status Code	Returned Value	Description
/games/{gameId}	GET	gameId <string>	Query	Record Matching Authentication (GamePlayers)	200	Game	retrieve game by gameId
					400	Error: reason <string>	game by gameId couldn't be retrieved
					401	Error: reason <string>	Denied access - userToken doesn't belong to a game participating user
/games/	POST	userMode <string>, gameMode <string>	Body	Any Authenticated User	200	Game	create a game
					403	Error: reason <string>	no userToken of a registered user was provided in the header
					400	Error: reason <string>	game couldn't be created
/games/game{Id}/guess	POST	xCoordinate <float>, yCoordinate <float>, time <float>, questionId <long>	Body	Record Matching Authentication (GamePlayers)	200	Score <int>	User's guess was received and scored, score returned to the user
					400	Error: reason <string>	User's guess couldn't be processed and scored
					401	Error: reason <string>	Denied access - userToken doesn't belong to a game participating user
/games/{gameId}/	PUT	gameId <long>, action 'start'	Query	Any Authenticated User	204	-	Game status changed to 'ongoing'
					403	Error: reason <string>	no userToken of a registered user was provided in the header
					400	Error: reason <string>	User couldn't start the game and its status to 'ongoing'
/games/{gameId}/	PUT	gameId <long>, action 'removeUser'	Query	Record Matching Authentication (GamePlayers)	204	-	User is no longer part of ongoing game
					400	Error: reason <string>	User couldn't leave ongoing game
					401	Error: reason <string>	Denied access - userToken doesn't belong to a game participating user
/games/{gameId}/	PUT	gameId <long>, action 'finish'	Query	Record Matching Authentication (GameCreator)	204	-	Game status changed to 'finished'
					400	Error: reason <string>	Game status couldn't be changed to 'finished'
					409	Error: reason <string>	Denied access - userToken doesn't belong to game creator
/games/{gameId}/questions/{questionId}	GET	gameId <string>, questionId <string>	Query	Record Matching Authentication (GamePlayers)	200	xCoordinate <float>, yCoordinate <float>, zoomFactor <int>	Satellite Image Information for question retrieved
					400	Error: reason <string>	Satellite Image Information for question couldn't be retrieved
					409	Error: reason <string>	Denied access - userToken doesn't belong to a game participating user

Table 3 - GameController Rest Specifications

LobbyController

The LobbyController is here for joining a lobby of a game and giving users already in the lobby a list of all users currently in the lobby.
(It seemed odd to have a POST request without a request body but due to the operation not being idempotent we still felt that we need to make it a POST request)

Mapping	Method	Parameter	Parameter Type	Authentication	Status Code	Returned Value	Description
/lobbies/	Post	gameId <string>	Body	Authenticated User	200	Lobby	Create a game room lobby for your game
					403	Error: reason <string>	no userToken of a registered user was provided in the header
					400	Error: reason <string>	Lobby for game with gameId couldn't be created
/lobbies/{roomKey}	Post	roomKey <string>	Query	Authenticated User	200	-	Join a game room lobby
					403	Error: reason <string>	no userToken of a registered user was provided in the header
					400	Error: reason <string>	Game room lobby couldn't be joined
/lobbies/{roomKey}	GET	roomKey <string>	Query	Record Matching Authentication (LobbyAttendees)	200	User[]	Retrieve users that are in lobby
					400	Error: reason <string>	users in lobby couldn't be fetched
					409	Error: reason <string>	Denied access - userToken doesn't belong to a game participating user

Table 4 - LobbyController Rest Specifications

LeaderBoardController

LeaderboardController provides one method to retrieve all scores on the leaderboard categorized by a game mode. We do not provide a method to create a leaderboard because there are only three instances of a leaderboard and those are hardcoded into our application. Clients can just contribute scores to those predefined leaderboards.

Mapping	Method	Parameter	Parameter Type	Authentication	Status Code	Returned Value	Description
/leaderboards/{type}	GET	type <string>	Query	Authenticated User	200	scores[]	Array of scores could be fetched
					403	Error: reason <string>	no userToken of a registered user was provided in the header
					400	Error: reason <string>	Scoreboard couldn't be fetched

Table 5 - LeaderBoardController Rest Specifications

User Interface Design – Mockups

Some functionalities are available on multiple screens, but for simplicity sake will be just mentioned the first time they occur. (e.g Back-button functionality)

The Landing Screen

This the first page an unauthenticated user sees of Mapguesser. From this site the user can read about the application or move to the “Login” or “Register” page, if the user clicks on the corresponding button.

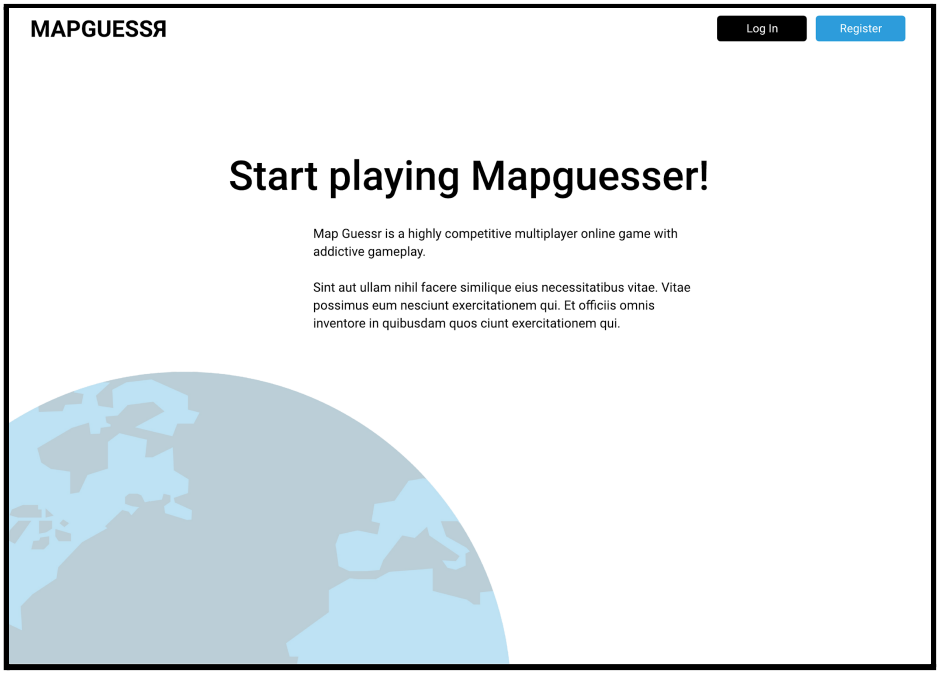

Image 1 - Landing Screen

The Registration Screen

This page is a classic registration page. We omitted the mockup of a login page, because the login is visually similar.
After a successful login/registration the user will be automatically redirected to the HomeScreen.

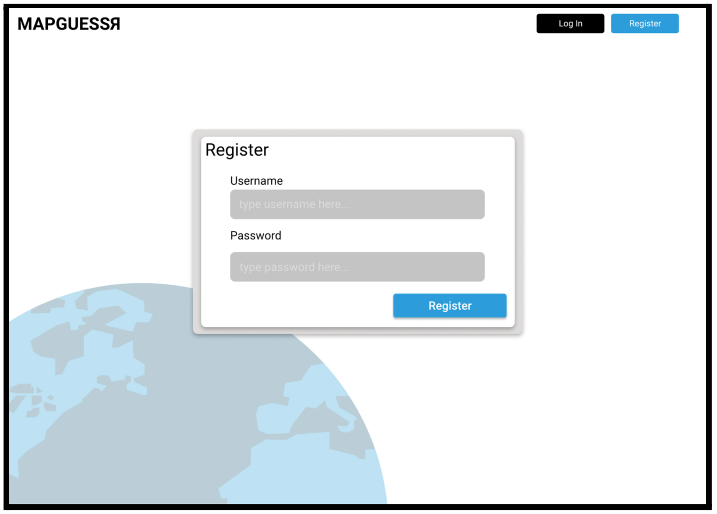

Image 2 - Register Screen

The Home Screen

This is the first screen that is displayed to an authenticated user. When the user clicks on the “Logout” button under the profile name, they will be redirected to the “landing page”. If the user clicks on “Edit”, the “edit page” gets rendered.
Furthermore, the user can click on the categories of game modes (above the scoreboard) and see the scoreboard for the respective gamemode.
To start a game, a user can choose to play on single or multiplayer mode and in either case a Game-mode-Component will be rendered.

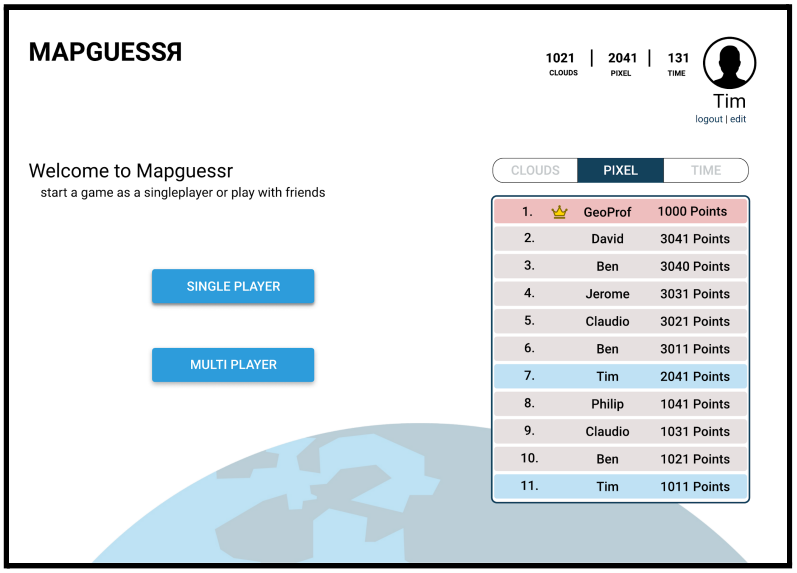

Image 3 - Home Screen

The GameMode Screen

On the right side of the screen, the functionality stays the same as on the Home screen.
On the left side, a user can pick between the game modes or go back to the screen before.
If the user chooses a game mode, he will be redirected depending on the “Usermode” he chose on the screen before, so for single-player a game will be started directly (on the “Game Screen”) and on multiplayer the “Create/Join Screen” will be shown.

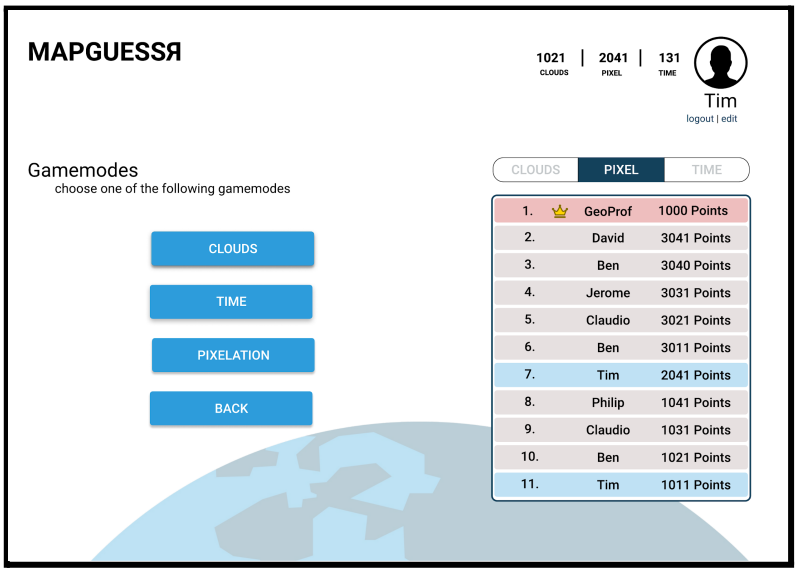

Image 4 - Game-mode Screen

The Create/Join Screen

On this screen, the user has the option to click on “Create” and be redirected to the “Lobby Screen” or on “Join” to be redirected to the “JoinRoom Screen”.

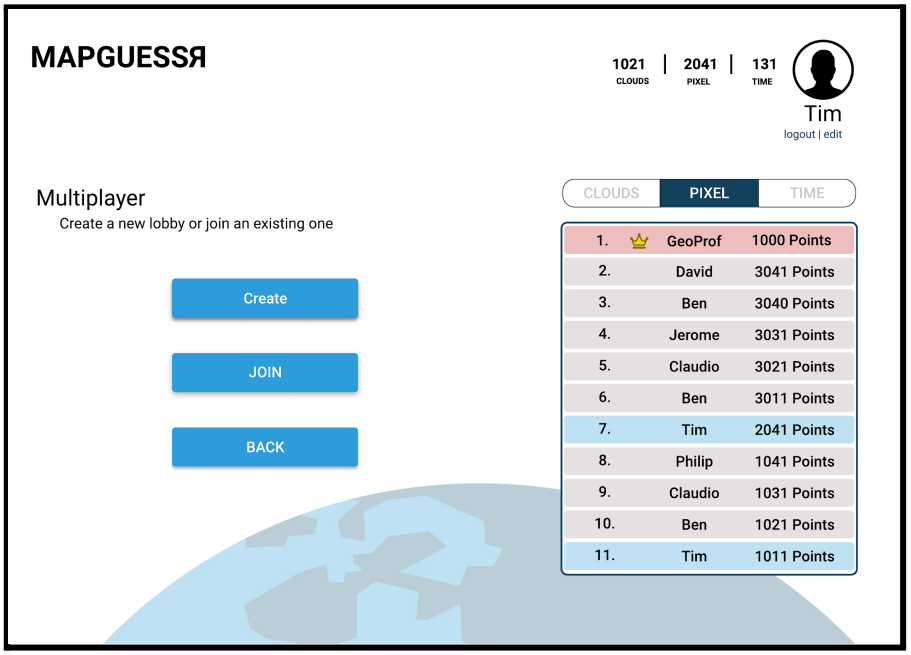

Image 5 - Create/Join Screen

The JoinRoom Screen

On this page the user can type the roomkey of a friend into the input field and join the room. On success, the user gets redirected to the “Lobby Screen”. Upon failure, an error is shown similar to this screen.

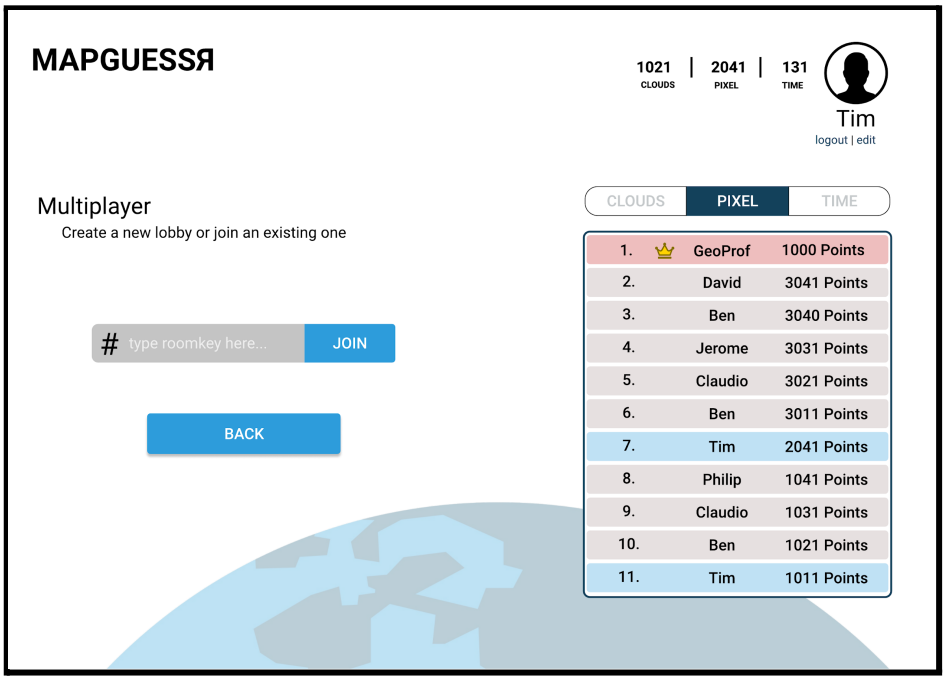

Image 6 - JoinRoom Screen

The Lobby Screen

Depending on the user, if they are the game host or not, the “Start Game” button is visible/clickable. If there are enough users to start a game, the host (the one who created the game) can click on the “Start Game” button, and then all users get redirected to the game. Each user in the lobby has the option to copy the room key (click on symbol next to invite Code) to the clipboard.

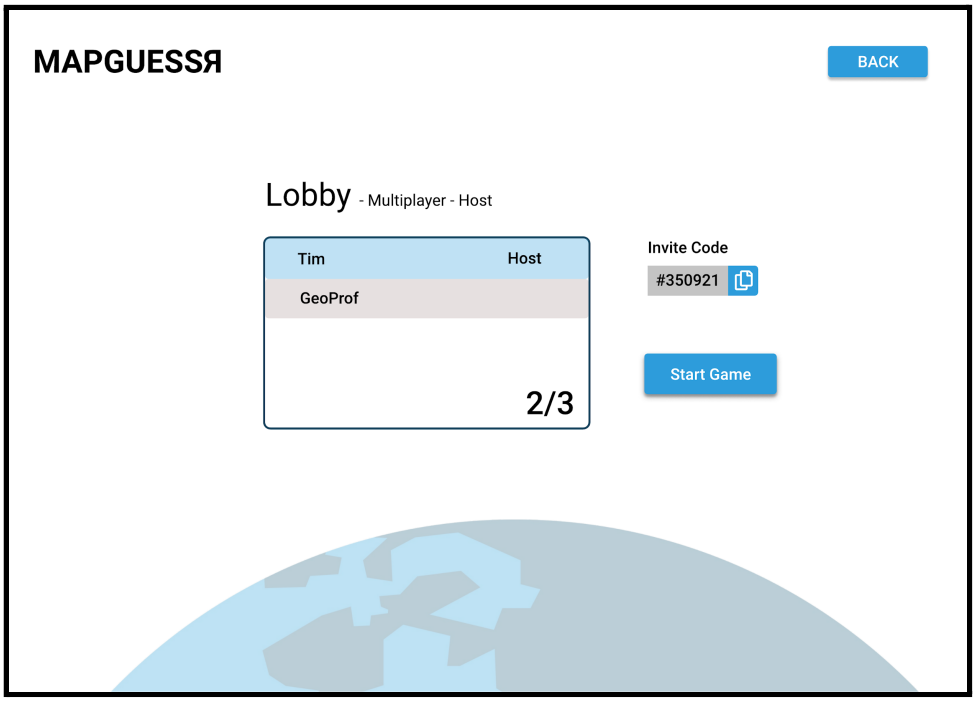

Image 7 - Lobby Screen

The Game Screen

During the game there is the option to exit the game early, when the user clicks on “Leave Game”, they will be redirected to the homescreen. In order to make a guess, the user needs to click on the minimap to enlarge it (enlarged Minimap).

Image 8 - Game Screen

The Enlarged Minimap Screen

On the enlarged map the user can place the pin on any location they want. After pinning a location and clicking on “Submit Answer” the user will see either the “Round Results” or the “Final results” screen, depending on the round number (after the last round, the user goes to “Final results”).
If the user clicks on a location different than the first, then the pin will move to the respective location. On the right corner of the screen is a symbol for “resizing”, which has the same functionality as a “Back” button (screen before).

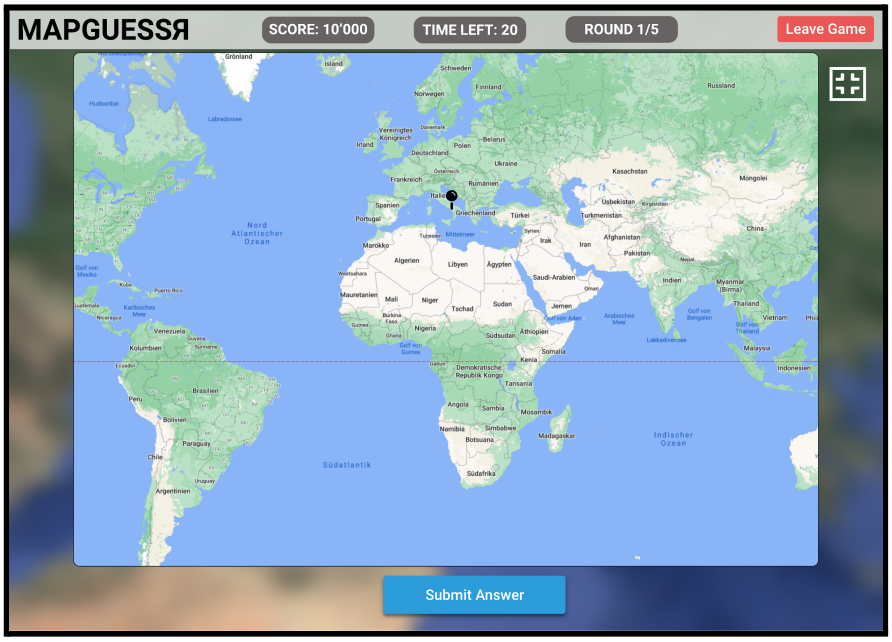

Image 9 - Enlarged Minimap Screen

The End-Of-Round-Screen

The user can see their score, and on multiplayer also the score of his opponents. Each user waits until all players are in the “End-Of-Round-Screen”, in order to start the next round synchronously. After a countdown of 5 seconds , the user will be redirected to the next round (screen).

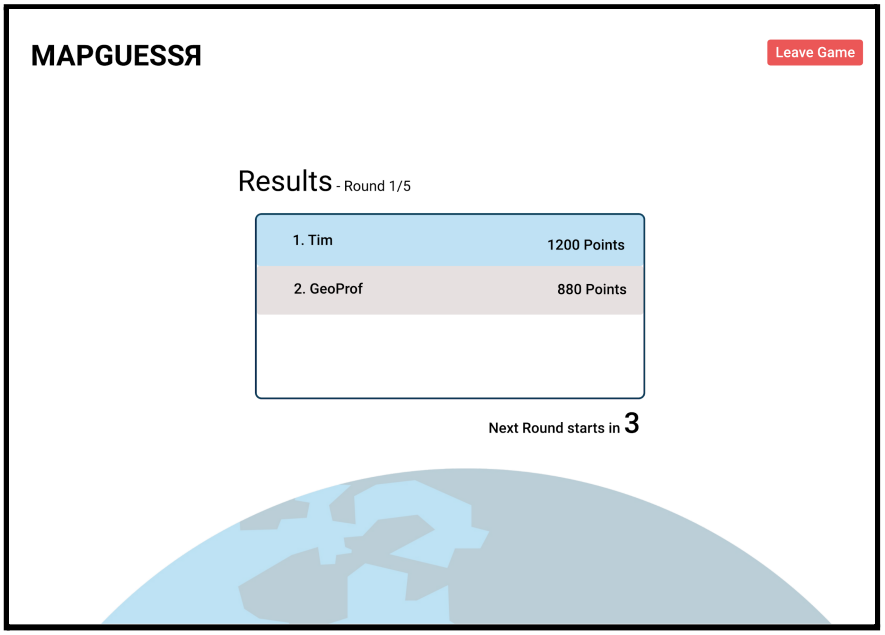

Image 10 - Round Results Screen

The End-Of-Game-Screen

The users can view their score and click on “Finish” to be redirected to the “Home Screen”.

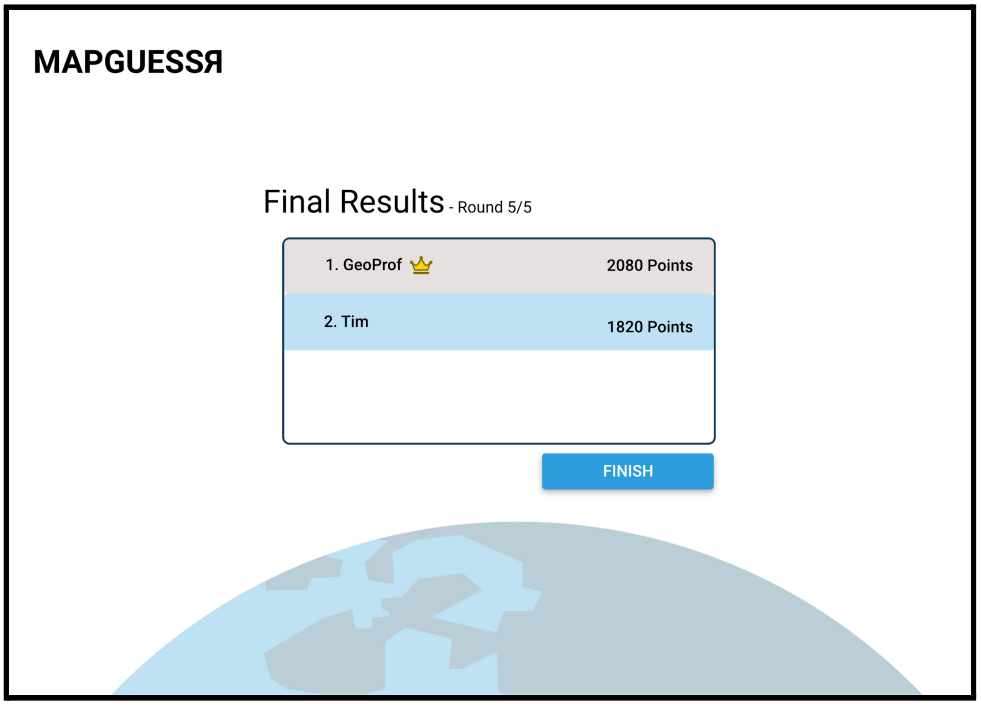

Image 11 - Final Results Screen

The Edit Profile Screen

The user can change with the fields of his profile and click on “save”. On success, they will get back to the screen they were before (one of the screens, where the profile is seen). On error, the popup will be shown.

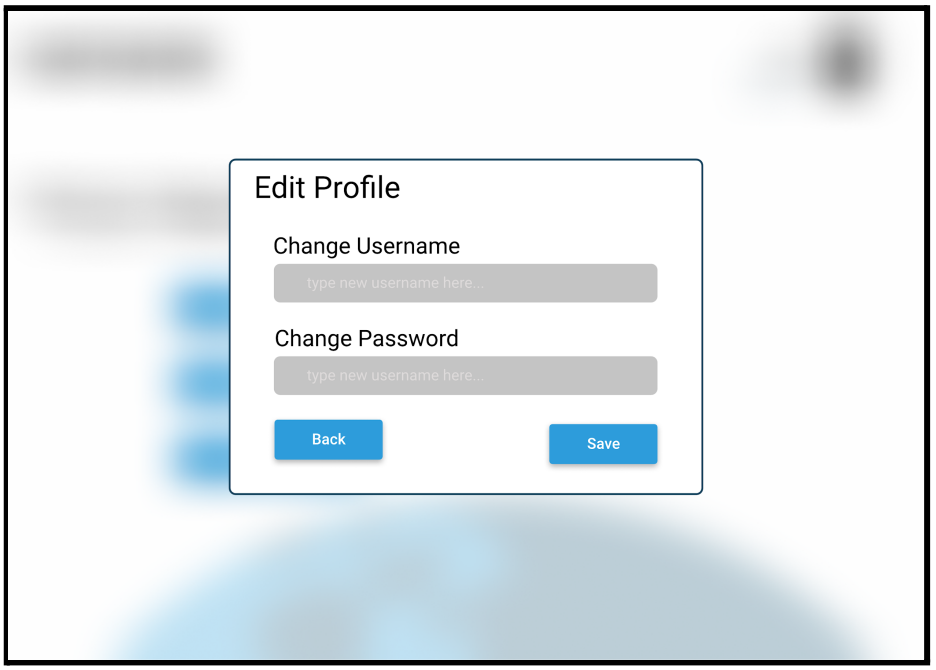

Image 12 - Edit Profile Screen

The Error Screen

This screen will be shown, if needed. The content of this screen will be tailored to the need.

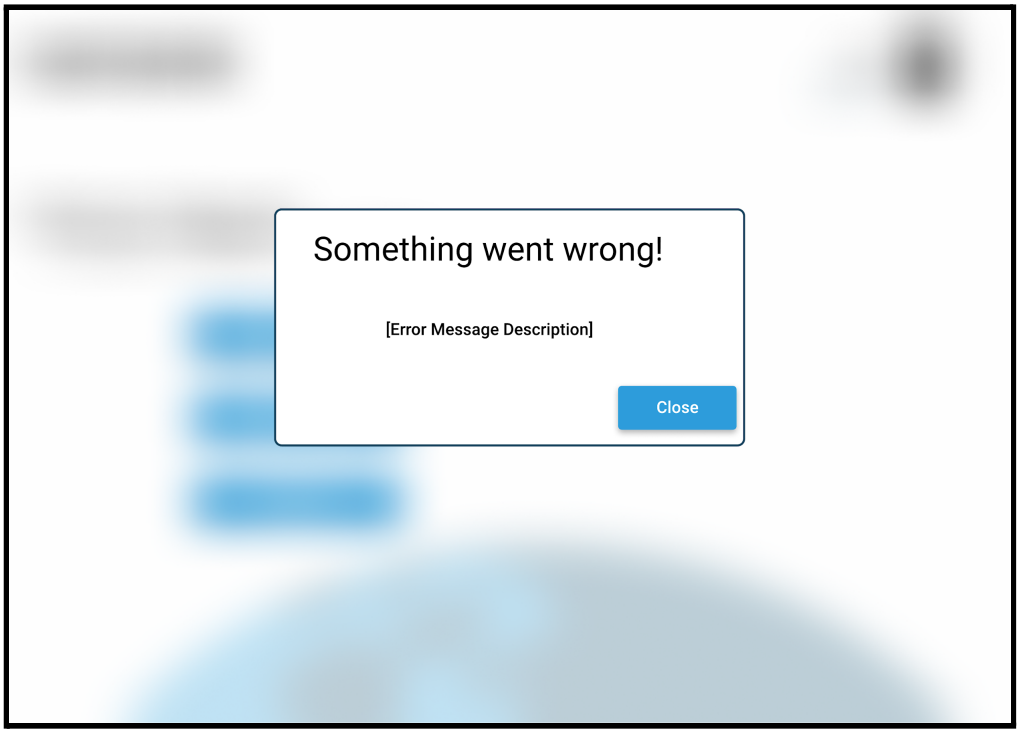

Image 13 - Error Screen