VILNIUS UNIVERSITY
FACULTY OF MATHEMATICS AND INFORMATICS
INSTITUTE OF COMPUTER SCIENCE
DEPARTMENT OF COMPUTATIONAL AND DATA MODELING

Software architecture project

Informational system for online chess play

Done by:

Eimantas Ramonas signature

Justas Lučiūnas signature

Supervisor:
Lect. Andrius Vytautas Misiukas
Misiūnas

Vilnius
2021

Contents

1 Introduction

The goal of this project is to create an online system where users can play chess.

2 Development iterations

1.
   - A way for users to register their accounts
   - Ability for users to log in
   - Password recovery system
   - Ability for users to play unrated games
2.
   - Implement rating system
   - Ability for users to play rated games
   - Ability for users to play against friends
   - Ability for users to play against AI
3.
   - Ability for users to view their match history on their profiles
   - Ability for users to view rating ladder
   - Ability for users to view other profiles
   - Ability for users to view a finished game (PGN)
4.
   - Ability for users to report other players
   - Ability for users to spectate games
   - Ability for administrators to view reports
   - Ability for administrators to ban players

3 Technologies/tools used

We used Express.js for our application back-end, React.js for the front-end and MYSQL for our database. We want to use these tools because they are quite popular at the moment and we think it is a good idea to learn them. We also planned to use GitHub to help organize our work process, but ended up working directly on the Apache webserver which is hosted in VU OpenNebula.

4 Non-functional requirements

The system will have an intuitive and fluid GUI to make the user experience pleasurable. To ensure security, we will encrypt the passwords of registered users. Our application should be quite lightweight and run fast on most browsers.

5 System design

5.1 Use cases of system functions

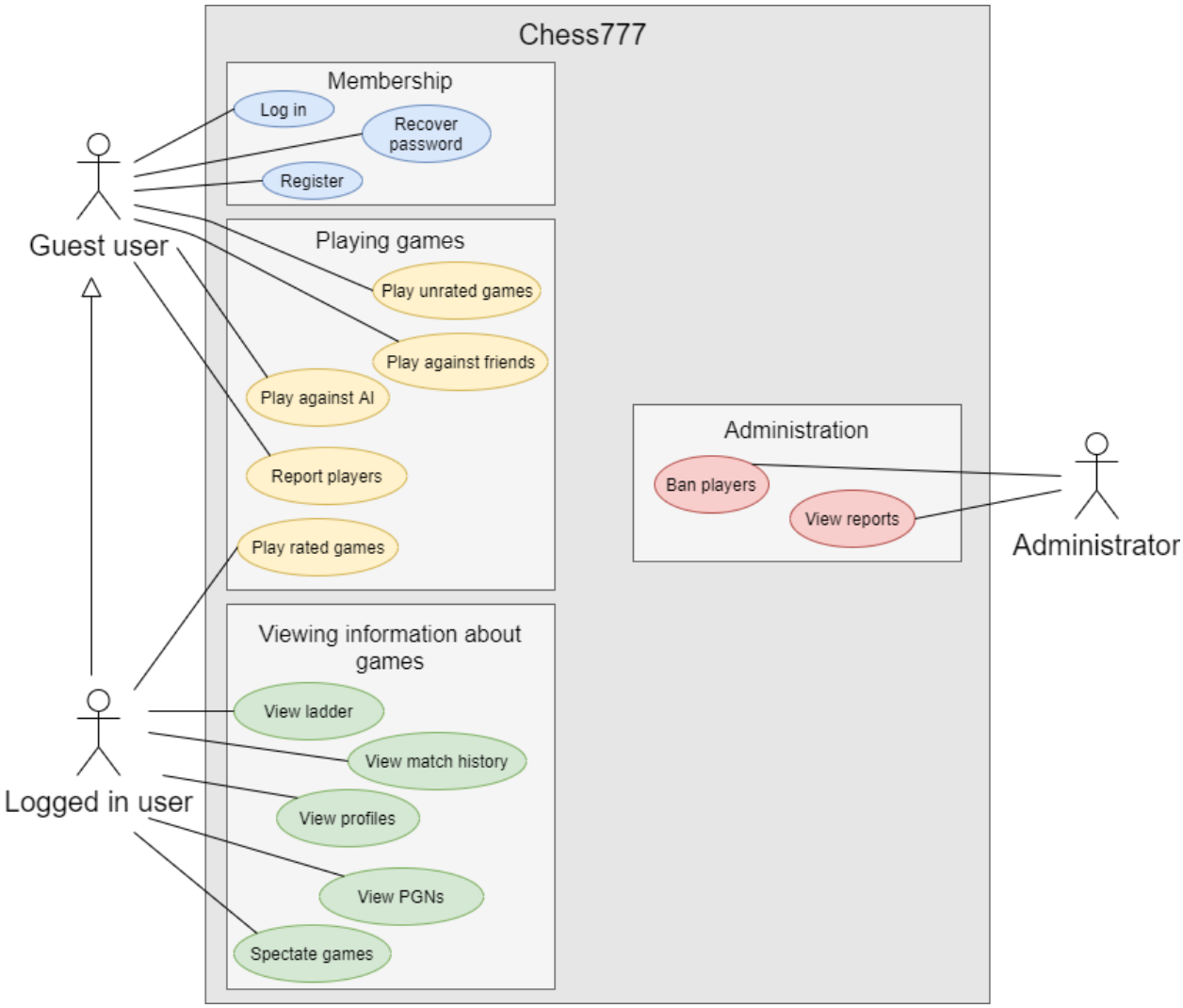

Figure 1. Use case diagram

This diagram describes the behaviour of our system from an external point of view. It shows what functions are available to different actors that use our system. There are three different types of actors: guests, logged in users and administrators.

5.2 Database model

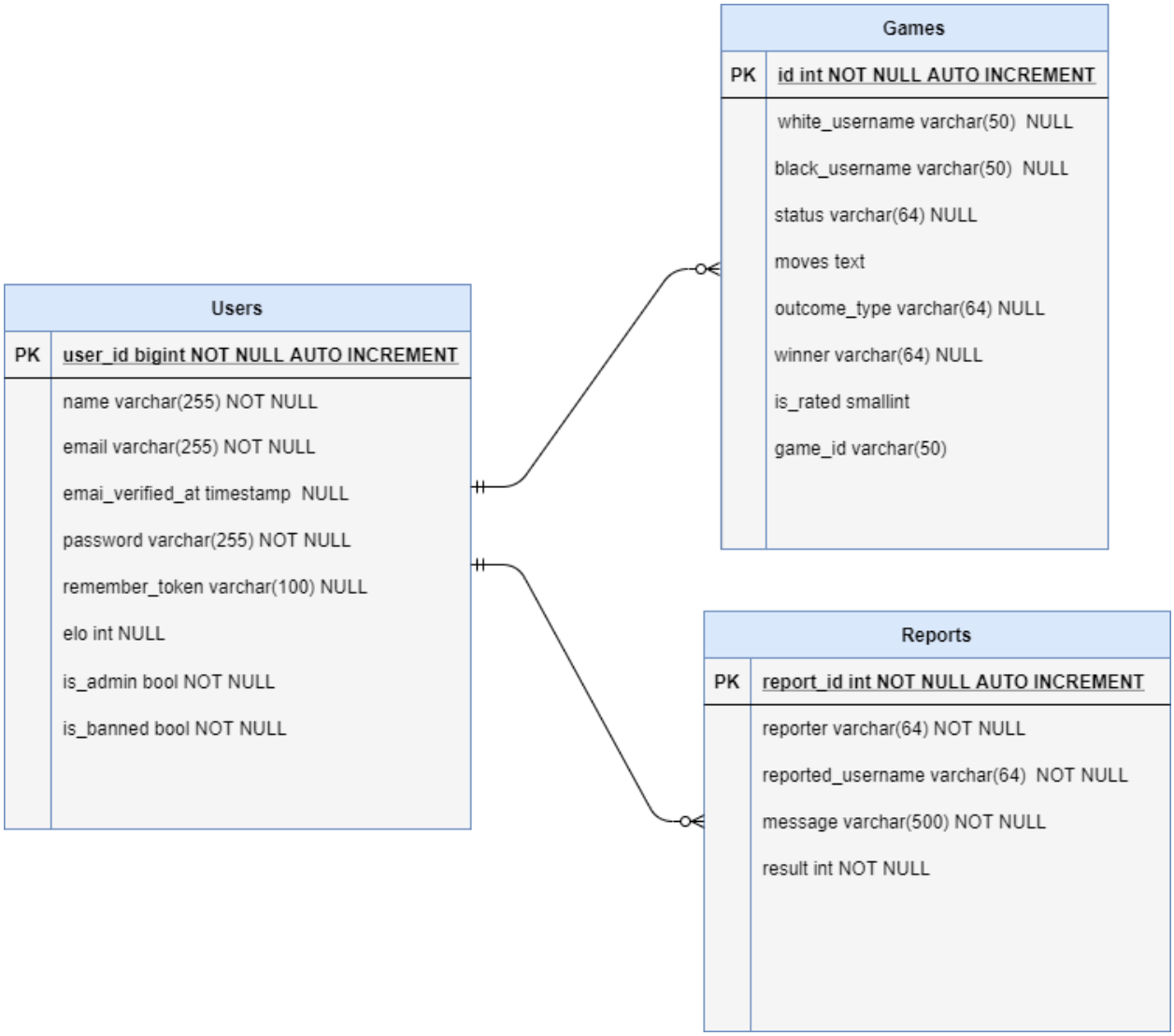

Figure 2. Database diagram

The diagram above describes the planned model of our database. It has three tables, each for information about users, matches and reports. The users table holds information about existing users. The matches table holds information about ongoing and finished matches. Finally, the reports table holds information about submitted reports, it will be used by administrators to decide whether or not to ban a player.

5.3 Plan for password recovery

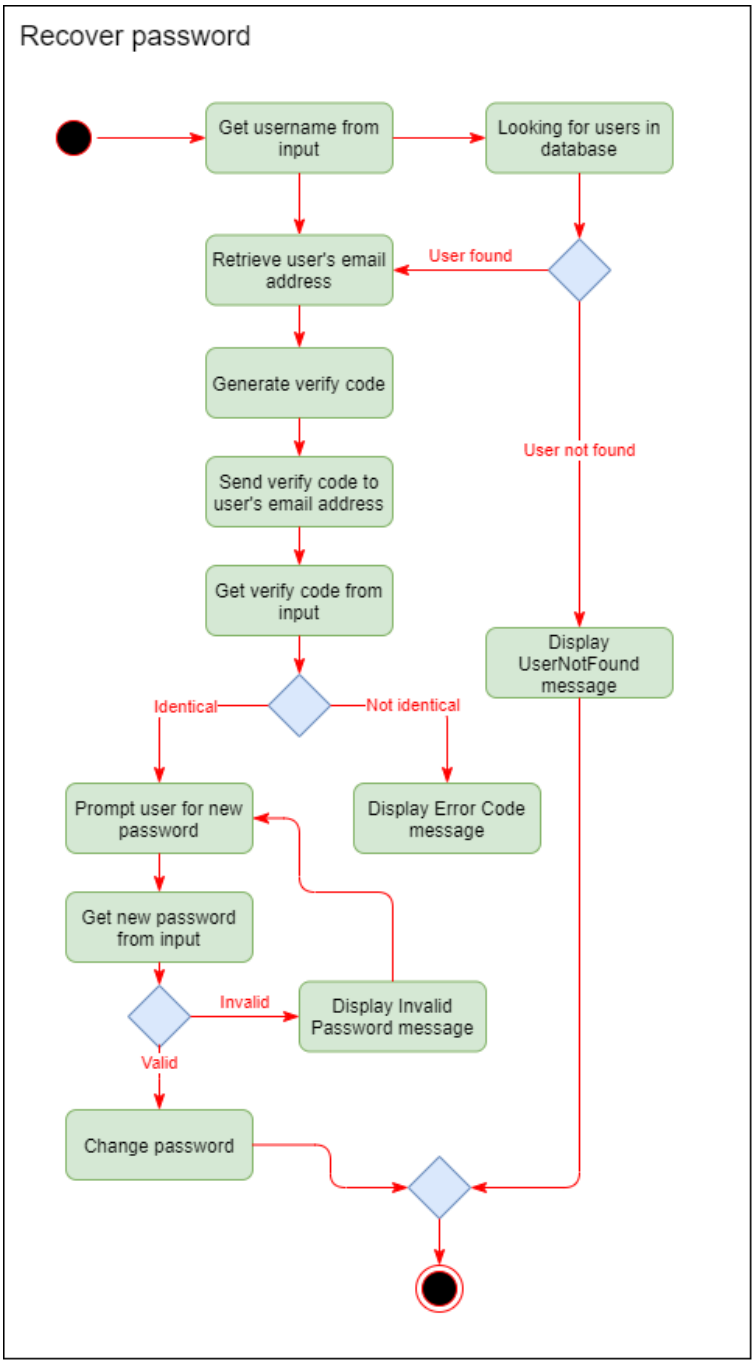

Figure 3. Activity diagram

The activity diagram above shows the flow of our password recovery functionality. First, the user will have to input his username into a form. If the given username is in our database, an e-mail with a verification code will be sent, otherwise an error message will appear. Then the user will be asked to enter the verification code. If the entered code is correct, the user will have an option to change his current password, otherwise an error message will appear. If the entered password matches all the requirements, account password will be changed, otherwise an error message will appear.

6 Deviations from initial plan

6.1 Technologies used

Initially we attempted to use Laravel as the back-end of our system, but we ran into too many problems and decided to use Express.js instead.

6.2 Database model

We changed data types of most columns in our tables for convenience sake and also added a few columns. In users table we added email_verified_at and remember_token columns when we were trying to work with Laravel as our backend. Also, we changed the "Matches" table name to "Games", and replaced the columns "first_user_id" and "second_user_id" to "white_username" and "black_username", because we believe that in a chess database this makes more sense. We also added a column named "is_rated", so we could differentiate between game types, as well as "game_id" where we store the ID from the URL of the game.

7 Installation instructions

Prerequisites:

- Node.js 14.17.0 or newer
- Web-server (locally hosted server should be fine)
- MySQL database

To install the project:

1. Clone this project to the public directory of your server:
   https://github.com/EimantasRamonas/chess777
2. Run **npm install** via terminal from backend and react-frontend directories (delete package.lock if it outputs error and retry)
3. Create database for the system manually
4. Open ports for your webserver and for sockets if needed
5. These files: **react-frontend/src/config.js** and **backend/create_tables.js** contain values which you need to change according to your system setup. **backend/server.js** also contains values that you might have to change:
   - line 16 - information about your database
   - line 32 - port which listens to sockets
   - line 623 - port on which the app listens
6. Run this command from your terminal: **node create_tables.js** from your backend directory
7. Run the backend via **npm start** (backend has to be running for the application to function)

8. To build the front-end part of the application use the following command:**npm run build**
9. Configure your webserver to point to this location: **react-frontend/build**