Sudoku Multiplayer App

Gleb Sokolovski

4th Year Project Report
Computer Science and Mathematics
School of Informatics
University of Edinburgh

2024

Abstract

This paper presents a novel approach to revitalizing the game of Sudoku through the development of a multiplayer app that infuses competitive elements and interactive features into the traditional puzzle-solving experience. Recognising the game's widespread popularity and the opportunity to enhance player engagement, this study introduces unique multiplayer rules and integrates game theory principles to make Sudoku more accessible and entertaining. By analysing existing applications and identifying their limitations, the research develops an innovative player rating algorithm inspired by machine learning, offering a nuanced assessment of player skill in this context. The paper details the app's design, emphasising the strategic use of cloud technologies and user interface considerations, and discusses my personal twist on the Rapid Solo Software Development methodology tailored for this project. User feedback played a pivotal role in the app's iterative development, focusing on getting the most honest feedback possible by relying on anonymity. The findings demonstrate the app's potential to transform Sudoku's appeal, offering insights into future directions to expand its reach and functionality, thereby broadening the game's demographic and redefining common perceptions associated with it.

Research Ethics Approval

This project obtained approval from the Informatics Research Ethics committee.
Ethics application number: 617018
Date when approval was obtained: 2023-12-29
The participants' information sheet and a consent form are included in the appendix.

Declaration

Acknowledgements

I wish to extend my profound appreciation to my project supervisor, Professor Nigel Topham, for his relentless guidance and expert advice on best practices in the development of this application. His insights have been invaluable to the project's success.

Additionally, I am deeply indebted to all the testers who consented to partake in our study as prospective users. Their contributions have been instrumental in refining the user interface design, enhancing the game's features, and facilitating comprehensive market validation. Their feedback has played a pivotal role in the evolution of this application.

I would also like to convey my gratitude to my family and friends for their unwavering support and empathy during my academic endeavors.

Table of Contents

Chapter 1

Introduction

1.1 Motivation

Sudoku is a game invented in Japan, which gained a lot of popularity in 2005 in the UK and the USA. By latest poles, 56% of the US population had played the game Games and in the UK 54% of the population like the game and a staggering 97% have heard of it UK. The player is given a 9×9 grid divided into 3×3 blocks. Some of these squares are already filled with a number between 1 and 9. The task of the player is to complete each square in a way that each row, column, and 3×3 box contains all the numbers 1 to 9. There are now many different variations of the game, but that is the main premise B. Felgenhauer [2006].

The simplicity of the rules and the difficulty of the puzzles has motivated many to tackle the problem from many different angles, creating personal techniques and constantly wanting to get better and faster. In fact, there are even Sudoku world championships held every year Team [i]. The competition typically consists of 100 or more puzzles which are solved by each competitor over multiple days Federation [2022].

Beyond the competitions, there had been many papers published on the matter, looking at different problems such as how many possible puzzles there are, how many squares need to be filled for a unique solution, AI algorithms, metaheuristics, quantum algorithms, and so on, for solving Sudoku puzzles Lewis [2007] Pal [2020] Simonis [2005].

Sudoku, a puzzle game familiar to many, often sees a divergence in player engagement, with some enthusiasts embracing it while others shy away, perceiving it as an intricate mathematical challenge rather than entertainment. This perspective presents an opportunity for innovation, aiming to transform Sudoku's appeal by integrating game theory principles, fostering social interactions, and enhancing user experience through an attractive interface. This approach seeks to broaden Sudoku's appeal, mirroring the success of platforms like chess.com, which redefined chess from a niche interest to a global phenomenon. By re-imagining Sudoku in this light, the goal is to shift perceptions, inviting a broader audience to appreciate the game's inherent allure and strategic depth.

In this paper, I make the following contributions:

- Introducing novel rules designed to enhance the multiplayer Sudoku experience, detailed below in Section 1.2.
- Delving into existing Sudoku multiplayer applications, identifying their limitations, and outlining my strategy to surpass these by leveraging their foundational work, as discussed in Chapter 2.
- Unveiling a machine learning-based player rating algorithm tailored for Sudoku, capturing various game nuances, detailed in Chapters 2 and 4. This includes a comparison with traditional chess rating systems and insights from interviews with esteemed chess players.
- Proposing the HoPro (Honours Project) methodology, an evolution of the Rapid Solo Software Development approach, with a focus on user feedback and own vision development, outlined in Chapter 3.
- Exploring the design of the application, highlighting the strategic use of cloud technologies, UI considerations, and comprehensive technology choices for front-end and back-end development, discussed in Chapter 4.
- Sharing insights from user interviews and how these shaped the app's iterative improvement, focusing on enhancing the app's scalability and performance, detailed in Chapter 5.
- Reflecting on the project's success in redefining Sudoku's appeal and outlining ambitious future plans to broaden the app's reach and functionality, as presented in Chapter 6.

1.2 Proposed Rules for the Sudoku Multiplayer Game

The full set of rules can be seen below.

1. Sudoku Challenge is a game played on a 9x9 grid divided into smaller 3x3 sections. Some cells are pre-filled with numbers.
2. Select an empty cell to guess which number (1-9) fits, according to Sudoku rules.
3. Each number must be unique in every row, column, and 3x3 section.
4. Clicking a cell ”locks” it for 5 seconds, preventing your opponent from selecting it.
5. Within these 5 seconds, enter your guess. A correct guess scores a point; incorrect or no guess means no point and a 5-second lockout from that cell.
6. Correct guesses are reflected on your opponent’s grid. Incorrect guesses remain unseen.
7. If both players select a cell simultaneously, it turns yellow, signaling a direct contest for that cell.
8. In the rare case of identical timing for answers, the cell is highlighted differently, and both players earn points.
9. Two toggle buttons beneath the grid control the note-taking feature. Left for notes on, meaning guesses are hypothetical and unseen by your opponent. Right for notes off, making all selections final.
10. Victory is achieved by securing more cells than your opponent.

Chapter 2

Background

2.1 Multiplayer Games

Multiplayer games are one of the most popular-played games to date. According to Balasubramanian [2023] as of 2022, ”54% of the most active gamers worldwide play multiplayer games at least once a week, for seven hours on average”.

People love competing against one another in their favourite games. It makes the games a lot more engaging and fun. Sudoku multiplayer is no exception. There is mainly one thing that comes to mind when thinking about Sudoku multiplayer game: two players are given the same puzzle, both timed while they solve, and the one with the lower time wins - a lot of examples of those can be seen in the next section. I propose a different solution, where both players play on the same board at the same time, while seeing each other’s moves with the aim to fill out more squares than your opponent. A lot more gaming tactics could get involved that way, making the otherwise less exciting game of looking at a 9×9 grid more competitive, and give a more of a multiplayer feeling.

2.1.1 Existing Sudoku Multiplayer Games

Here are the notable example of existing solutions that have already gained traction either on the web, or as a mobile application:

- ”**UsDoku**”
  This is a classic example of two players competing against each other by solving the same puzzle on two different boards and timing to see who wins.
  Things done well:
  - User Interface is very calming and inviting. There are clear instruction on how to play and where to go.
  - You can either join an existing game, or create your own game and invite friends via a link.

  Things to exploit:

  - There isn't that multiplayer sense that one gets when playing on the same board.

- "**SudokuZentai**"
  This mobile app takes a different approach to the one before, where by two users play on the same board, but in a more collaborative ways than competitive. So the user can see what the other player has entered into the cell, but the main goal of the game is to complete the puzzle as quickly as possible **together**.
  Things done well:
  - Each player can see what the other player has put on the board.

  Things to exploit:
  - While good colour palette, the User Interface is not super clear, and it's left for user to guess what the rules of many different types of games are, like "Trainee", "Pushover", "Fair", "Zenkai".
  - The game needs to be more competitive to give it a more multiplayer feel.
- "**SudokuMultiplayerChallenge**"
  This mobile app has the right idea regarding multiplayer. The app puts two players against each other on the same board. The person with the most squares filled - wins.
  Things done well:
  - The multiplayer aspect.
  - You can choose an avatar upon signing up, which gives the app a more personal feel.
  - It's clear how to operate the app.

  Things to exploit:
  - Small improvements on the interface are needed.
  - The Same Square Error.

2.1.2 The Same Square Error

The **Same Square Error** (**SSE**) is a term created by me which describes a very specific problem when it comes to multiplayer Sudoku: what happens when the users click on the same square at the same time [1] and they try to fill it out?

As we know from the proposed rules for the Sudoku multiplayer described earlier, when a user clicks on a cell, they "lock" it for 5 seconds, during which time the other user can't "lock" it themselves. During those 5 seconds, the user must answer correctly or they will be locked out from entering anything into that cell for the 5 seconds following that. But let's imagine the scenario where one user clicks on a cell and tries to lock it.

[1]It doesn't have to be at exact same time, the time I am referring to here is the time at which the server receives a message through WebSocket that the user wants to lock that cell.

The message goes to the server, through the WebSocket connection, where the other user's WebSocket connection picks it up and sends it to the user's front-end, where the cell is rendered as taken by other user. In the process of transferring that data, the other user also clicks on the cell to lock it - by that sending a message to the server - and does it before its shown as taken on their end. So now the server has 2 different messages suggesting the cell is locked by 2 different players, and both players' devices are displaying they locked the cell but have also received a message from the server that the other player has also locked it. This is the essence of the **SSE**.

The solution to this is as follows, assign a different colour to the cell when that happens - letting the players know that the square is taken by both players and they both have 5 seconds to answer it correctly before their opponent does. Unfortunately that's not the end of the problem - if someone beats the other person by a clear margin, there is no problem there, however what if another **SSE** happens where the players answer once again at the same time[1]. In that case, we treat the cell as correct for both players, assigning it once again a separate colour, and counting it towards each player's correct cell count.

Of course the main motivation behind the architecture of the app is to prevent that from happening as much as possible by utilising a mixture of protocols, some designed for speed, and some - for safety. I talk more about it in Section 4.2.13.2.

2.2 Sudoku Player Rating

2.2.1 What is a player rating?

Different games and sports have their own meaning behind their rating. The main idea is to put all players on some scale to see how they compare against each other. Football has an overall rating, which consits of rating a player on attributes like speed, agility, skill, etc. Chess ratings are calculated from the player's previous opponents. While games like PubG and CallOfDuty base their ratings on how much the user has played and how successful they were at a certain gameplay Team [2023b]. In most cases, if we put two players against each other, we would expect the player with the higher rating to win. So from this, we can define player rating as an indicator of the likelihood of a player's success against another player.

2.2.2 Existing examples of rating players

As mentioned previously, a lot of sports and games have a rating system. Unfortunately, there isn't a global Sudoku rating system, which means it leaves room to create my own. As a case study, I looked at the current rating system existing in chess right now, and an example of a Sudoku rating from a web based Sudoku player.

1. ”OnSudoku”
   This is one of the very few Sudoku player rating systems I could find, and perhaps the most interesting one. It's not a multiplayer game, so the ranking doesn't depend on the oponent you beat, but rather on three factors:

(a) Your score leading up to the puzzle,

(b) Difficulty of the puzzle,

(c) Time taken to solve the puzzle.

Their proposed formula for calculating the points earned after each puzzle is as follow

$$B = K \times \frac{D}{0.01 + \frac{T}{3600}},$$

where T is the time taken to solve the puzzle, D is the game difficulty coefficient, and K is the player level coefficient. This would appear to give a good distribution among players, however an important thing to note is the the player coefficient K **increases** as the level of the player increases. So that would mean that a player with an already high rating gets an even higher rating from solving puzzles. Which would mean that it's easier to get to really high level rather than an average level, which is not usually how games work.

2. The Chess Rating System
Rating chess players was one of the most important factors in making the game so popular. In the 1950s, Arpad Elo developed the theory for the current rating system for US Chess Federation, English Chess Federation, Chess Scotland, and many many more national chess federations. Most chess tournaments and platforms like chess.com, use the "Elo system" or a variation of it. The Elo rating system calculates a numerical rating - between 0 and 3000 - based on performances in competitions. It can go up and down according to your wins and losses. It's not a straight forward system, and takes into account many factors like when did you last play, and whether you have a rating to begin with, but the most important formulas to take away are:

$$E = \frac{1}{1 + 10^{-\frac{(R_A - R_B)}{400}}} \tag{2.1}$$

$$r_{\text{post}} = r_{\text{pre}} + K(S - S_{\text{exp}}) \tag{2.2}$$

where R_A, R_B are scores strengths of players A and B, E is the expected score of the game for player A, r_{post} is a player's updated post-tournament rating, r_{pre} is a player's pre-tournament rating, S is the player's total score in the tournament, S_{exp} is the expected total score estimated from the player's pre-tournament rating and the player's opponents' pre-tournament ratings, and K is an attenuation factor that determines the weight that should be given to a player's performance relative to his or her pre-tournament rating Glickman and Jones [1999].The paper by Glickman and Jones Glickman and Jones [1999] also displays the distribution of the chess rating, which they described as **bimodal**. In my opinion the most ideal distribution would be the bell curved Gaussian distribution, since most people would be around the average, with fewer being very good and very bad.

I managed to conduct two interviews with two competitive chess players - one is a class E chess player with a rating of 1098, and the other is a class B chess player with a very impressive rating of 1623. The questions I asked can be found in Appendix A. Based on their answers, both expressed that the rating they have is very important to them, that the rating is a very good indicator of who wins the match, and - very interestingly - the disparity in rating has a direct correlation with the anticipated duration until checkmate. They had a slight difference in opinion when talking about choosing opponents. The interviewee with the higher rating said that they heavily favour playing against opponents with higher ratings than themselves, while the player with the lower rating stated that they like to mix it up, and choose opponents with lower ratings when they want a nice, easy game. When asked the final questions, regarding whether the chess rating should incorporate additional metrics such as board possession, remaining pieces at game's conclusion, or the number of moves leading to checkmate, both players said yes, however to do it more carefully than that, as need to take into account the style of gameplay, some people are more aggressive, some are more passive, so the new hypothetical ranking system shouldn't discriminate the players based on styles.

2.2.3 Rating Sudoku Players

Thinking about all the above infomration, I think it would be fair to base the ranking on the following factors:

- Complexity of the puzzle
- Possession of the board at the end of the match
- The duration of the match
- Difference of players' rating before the sudoku clash
- Number of puzzles solved before the clash

Chapter 3

Methodology

3.1 Development Methodology

A software development methodology refers to the framework that is used to structure, plan, and control the process of developing a software system Software [2024]. There are many development methodologies to choose from and it's important to choose one that works well for the project and the team Team [2017].

Since the team consists of only one developer - me, I had to alter aspects of existing development methodologies. After some extensive research I came across a paper on a development methodology called **Rapid Solo Software Development (RSSD)** Methodology based on Agile Kristo RP [2022]. RSSD adapts the same phases from the Agile methodology but adds its own subphases. It is also important to point out that the ”RSSD methodology was tested on 10 total projects, and respondents agreed that it was helpful to streamline the development process, with an average score of 4.19 out of 5.0”. The phases of the methodology are:

1. **Meet Phase**: The initial meeting between the developer and stakeholders involves discussing, creating stickman diagrams to illustrate and confirm the final scope of requested features, and confirming the features before the meeting concludes.
2. **Plan and Pre-Evaluation Phase**: This phase involves re-evaluating the feasibility of the project, creating a detailed project proposal during the chunk subphase, and presenting this proposal along with necessary documentation to the stakeholders.
3. **Design Phase**: In this phase, the developer draws essential diagrams like class diagrams, ERD, or Use Case Diagrams, and prepares function and class declarations with comments. These structures are stored separately for documentation purposes.
4. **Develop Phase**: The developer creates the code body based on the prepared structures and cycles through coding, integrating, and verifying subphases to ensure modular and compatible functions and classes.
5. **Test Phase**: Each function is extensively tested using different cases, primarily through the black box method, and the code is optimized to enhance the program's

performance as the user base grows.

6. **Evaluate and Maintenance Phase**: This phase involves discussing with stakeholders, demonstrating the software, and collecting feedback in multiple meetings while simultaneously performing maintenance to fix bugs and make necessary changes, with the developer bearing a significant portion of the maintenance costs.

This methodology was mainly created for developers that are building out someone else's idea. Since all the specifications are coming from me, I altered some sections and added some sections to arrive at the final methodology. We'll call it the **HoPro** (Honours Project) Methodology:

1. **Idea Specification and Verification**: The main ideas and functionality for the application are drawn out here. This is the time to conduct all the necessary interviews with potential clients, hear their ideas and feedback in order to gain verification for your idea.
2. **UI Design Phase**: This phase is for designing the User Interface for the application. After the design is complete, get back in touch with potential users and get their feedback. If there are uncertainties or constructive criticism in the design, refine and review again.
3. **Plan and Pre-Evaluation Phase**: Similar to what is described above, however make a bigger emphasis on planning out which tech stack to use for this project - clearly outlining the reasons, benefits and drawbacks behind each one.
4. **Develop Phase**: Same as above, this phase is for writing the code. There are two sub-sections for this phase in my case: 1) computing the Sudoku Player Rating algorithm, and 2) coding out the actual application.
5. **Test Phase**: The app is given to potential users and their feedback is received.
6. **Refinement Phase**: This is the phase to use the results of the **test phase** to make adjustments as necessary. This includes fixing of the bugs, alterations in UI, adding/removing any features. The **test** and **refinement** phases are repeated until the feedback is fully positive.
7. **Evaluate and Maintenance Phase**: Using the previous two phases, this is a phase to evaluate how well the app works in comparison to success criteria outlined in the Analysing Successful Requirements of the App section, and also making plans for future improvements and rollouts.

As seen in the HoPro methodology, there is a big emphasis on potential users and getting their feedback. I believe this will be crucial in the app's original success. This way I ensure idea validation Daher [2022] and market fit Taylor [2022].

3.2 Analysing Successful Requirements of the App

Table 3.1 outlines the minimal amount of features that need to be achieved before the roll out of the app. The table is updated throughout every phase outlined in the HoPro

Feature	**Description**
Creating and deleting an account	The user should be able to sign up by providing their email, name, username, password, and a profile picture. The user can also delete their account, which removes user data permanently from the server.
Logging in and logging out	The user should be able to log in and out at any point on any device they prefer.
Settings	The user should be able to change their profile picture, email, username, name, and password that they provided at the sign up stage.
Viewing games played	The user should be able to view all the games they have ever played, the outcome of the game, and the amount of points they gained or lost after the game.
Viewing user statistics	The user should be able to view their rating, games won, lost, and drew, and number of games played.
Starting a new game	The user should be able to start a new game against a human (not a computer).
Playing the game	The user should be able to play the game based on the rules outlined in this section.
Toggle between note mode and play mode	The user should be able to make a note in the square, or to give a final guess for the square - like in most Sudoku games.
End game screen	The end game screen should appear after the game has finished, outlining the difference in possession, time taken, and the impact the outcome had on the rating. The end game screen should also have the ability to challenge the opponent again or return to the home page.

Table 3.1: List of minimal features required for a successful application and their descriptions

development methodology - features are removed or added depending on customer feedback.

While Table 3.2 displays additional features to be added if there is time.

3.3 Potential Users and Feedback Collection

Given that this is a smartphone application, the total available market encompasses all individuals possessing a smartphone and internet access. Within this group, the served available market is narrowed down to those proficient in basic English, as the app is designed in English. From this subset, the target market is further refined to residents of Edinburgh, representing a local and accessible segment for me, which simplifies the process of gathering feedback.

Feature	Description
Google and Apple Authentication	The user should be able to sign up and log in through their Google or Apple accounts.
Pick their own opponent	The user should be able to pick a particular opponent (for example their friend, or someone they've played before) to play by searching up their username.
Single-player mode	The user should be able to practice playing the game against a computer. The outcome of this game will not be recorded on their record - this is just for practice.

Table 3.2: List of additional features to add if there is time and their descriptions

I've decided to split the users from whom I get feedback into three categories:

1. People who know the rules of Sudoku and play the game regularly.
2. People who know the rules of Sudoku but never play the game.
3. People who don't know the rules of Sudoku and never play the game.

I've split it this way in order to get accurate perspectives on the game from different levels of Sudoku players. Group number 1 is important because they will be main drivers of traffic at the start. People who enjoy the game are looking for constant improvements and would be excited to try out a new concept.

As described in the introduction, this app is designed for group number 2 - people who have tried to play and didn't like it. Positive feedback from the second group would prove the concept to be correct. Finally, group number three would describe the performance to a brand new audience.

As outlined earlier, there will be multiple points of interactions with potential users. The two main points of interaction are after completion of the original design, and after the completion of the build phase. As seen in Appendix B , the questions asked after original design had to do mainly with the colour scheme, design flow, understanding of the UX, and so on. While in Appendix H, the questions were focused on the feel of the app, the understanding of the navigation of the application and the game. All the results from the interviews are outlined in the UI Design and Testing sections respectively.

The "mum test" is a term used when trying to validate your idea in business Hornby. It refers to the fact that your mum will always be proud of your idea and will never say anything negative about it. In this context, it's not just the mum, it can be friends, family, even strangers that just don't want to disrespect your idea, or feel uncomfortable confronting how bad it is. Feedback collection becomes extremely difficult when trying to get around this phenomenon. My solution is to have the interviewees give their feedback anonymously, hopefully ensuring that the feedback is more truthful. The users will all receive a link to the form at the same time, and will be asked to complete them based on the things discussed in the interview. The only information I will need from the users is which group they belong to, but no other identifiers are needed.

3.3.1 Participant Data Collection

All the user information is protected by the University of Edinburgh's research ethics and integrity of Edinburgh [2024]. My interviews were approved by the ethics board, and all the information about all the interviews were made very clear to all the participants.

I created a Participant Information Sheet (refer to Appendix J), providing essential insights into the study's motivation, methods, and additional resources for potential participants. This document covers details about the research team, the study, the effects of participation including possible risks and benefits, and data privacy measures. Additionally, I drafted a consent form (found in Appendix K) to secure participants' agreement. This form required participants to acknowledge their understanding of the information sheet, the study's purpose, and the voluntary nature of their involvement. It also sought their consent regarding audio and video recording and the use of their anonymised data in scholarly works and future approved research. Lastly, participants were asked to formally agree to participate in the study.

3.4 Project Timeline

To determine the project's estimated completion time, I evaluated the effort required for each task and factored in the demands of my other university courses. By organising the tasks into monthly increments, I devised the timeline displayed in Fig 3.1.

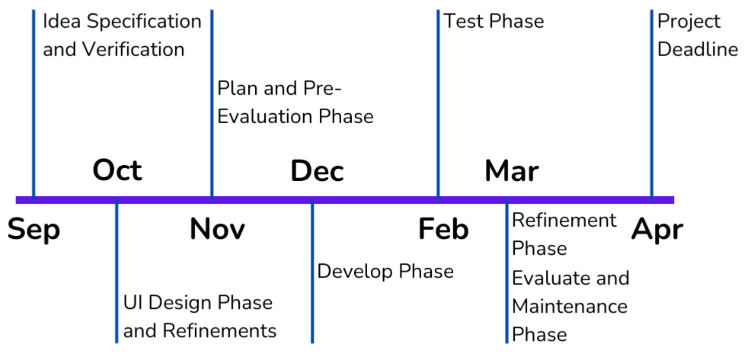

Idea Specification and Verification
Plan and Pre-Evaluation Phase
Test Phase
Project Deadline
Oct
Dec
Mar
Sep
Nov
Feb
Apr
UI Design Phase and Refinements
Develop Phase
Refinement Phase
Evaluate and Maintenance Phase

Figure 3.1: Projected Project Timeline

Chapter 4

App Design and Development

4.1 UI Design

Nowadays, the way the app looks holds as much value as the actual functionality of the app Team [e]. Users expect a certain standard when it comes to looks and feel of the app. There is a certain science behind it even - rules to follow, and concepts to upkeep Staff. The process of designing an application's front-end is its own journey - starting from the software to use and ending with the user feedback.

4.1.1 Figma

Figma is a collaborative software used primarily for design of applications Team [2024h]. I've been using Figma to design my projects for a while now. Its simplicity of use is the primary factor that drove me to use it for this project as well.

4.1.2 Final Design

4.1.2.1 Sign Up and Log In Pages

We start off by reviewing the first pages users will see - the Sign Up and Log In pages. Those are seen in Fig C.1.

The pages contain everything they need. The log in screen has input fields for username and password, the ability to authenticate using Google and Apple accounts, and a redirect to sign up page button. The sign up screen has the input fields for email, name, username, and password, but also a field to repeat password. It's a good practice to do this in case of a spelling error in the password field. The profile icon at the top of the fields is the prompt for profile picture upload. After clicking on it, the user is taken to their images where they select one, after which the prompt turns into the selected picture. Like the log in page, the sign up page also has the ability to sign up with Google or Apple account, or be redirected to the log in page if they already have an account.

4.1.2.2 Home Page

We now move on to the home page, as demonstrated in Fig C.2.

The screen on the left is the first thing the user sees after signing up or logging in. The first box contains the user's information: profile picture, username, number of wins, losses, and draws, and the custom Sudoku rating.

Subsequently, the next section displays the user's last five Sudoku clashes. Each clash's details include the opponent's username, their rating prior to the game, the result of the match, and the points gained or lost. The application showcases only the last five matches to enhance usability and design. Additional matches can be accessed by selecting the "View More" button. This design choice serves two main purposes: firstly, to maintain a clean layout, as accommodating 200 games on the homepage would be impractical. Secondly, it enhances user experience by enabling quicker loading times, ensuring users can swiftly commence a new game without the delay of loading extensive game histories. Another aspect of the design incorporates color coding to intuitively differentiate outcomes: the positive green signifies winning, while the negative red indicates losing.

4.1.2.3 Settings Pages

Now let's review the settings screens, seen in Fig C.3.

As can be seen in Fig C.3(a), there are three buttons: Account Settings, Delete Account, and the Go Home button. I've decided to put the home button at the bottom, instead of an arrow in the top left - like seen in many applications - for ease of user reach. The user would be able to easily reach that button with the thumb without the need to reposition the hand Ingram [2016]. Fig C.3(b) shows what happens when the user clicks on the "Delete Account" button - a prompt appears asking the user to verify their last action. It's a common response for important data altering action - to prevent accidental deletion of the account. Then Fig C.3(c) shows the screen that appears after the user clicks on "Account Settings" button. The user is given the option to change their profile picture, their name, and their email. Clicking on the save icon on the bottom right will save the changed details, while clicking the left arrow in the bottom left, takes the user back to the Settings page without saving any changes.

4.1.2.4 GamePlay Page

We now move on to perhaps the most important part of the app - the Game Page. There is a lot of functionality to cover in a single page, so Fig C.4 shows the games page annotated.

The page contains the following components:

1. Sudoku grid,
2. Game timer,
3. Game score,

4. Timer indicating amount of time left on the locked square - green,
5. Timer indicating amount of time left on the locked out square - red,
6. Squares filled from the start - white background with black numbers,
7. Squares filled by the user - dark purple background with white numbers,
8. Squares that are locked by the user - dark green background with no numbers,
9. Squares that the user is locked out of - red background with no numbers,
10. Squares filled by user's opponent - blue background with white numbers,
11. Squares locked by the user's opponent - red background with the five second timer ticking off in white,
12. Squares that were filled by both the user and the opponent at the same time,
13. Toggle to switch between play mode and note mode,
14. Quit game button - if clicked the user withdraws from the game - making the opponent the automatic winner, and
15. Numbers to click when making a guess or a note into the square.

I used color contrast to differentiate various sections and enhance user awareness of the on-screen components due to the substantial amount of information displayed Kauranen [2023] Patteri [2020].

4.1.2.5 End Game Page

The final page is the page that appears after the game has ended. It contains all the necessary information: the score, the time it took to complete the game, a fun display of possession of the board, and a couple of navigation buttons. From there you can a) go straight into a new game with a different opponent, b) challenge the same opponent again, or c) go back to home page. The page is displayed in Fig C.5.

4.1.3 Interview Results

As planned in our HoPro methodology, after the original design, they needed to be shown to potential customers and feedback to be recorded. The questions for the interviews are seen in Appendix B. The users were given 9 statements, and they had to answer with a number between 1 (completely disagree) to 10 (completely agree). So the maximum score that would suggest the application's designs have fulfilled their purpose is 90. The average score came back as 86, which is definitely very high. The statements where most points were lost were statements (b) and (f) that have to do with the rules and how to play. I have reasonable assumptions to think that they are one of those things that will solve themselves once the users start playing, since grasping the rules of a new game is always complicated and you only get the hang of it once you start actually playing Bycer [2020].

There was also a response from the optional second part of the questions - leaving a comment on the entire application. The comment was a recommendation to include a tutorial in the GamePlay page if this is your first time playing. This is a usual occurrence in games, and the suggestion came from a player from group 1 - they know the rules and play the game regularly - suggesting they must be familiar with these tutorials appearing in other applications. Fig E.6 shows an example of such tutorial from a Sudoku app I use.

Based on this, I have designed my own tutorial screen, shown in Fig C.7. This was not shown to the users as they will see it in the actual application when they try it out.

4.2 Plan and Pre-Evaluation

Now it's time to design the application's architectural design. It's important to design the application to be resilient, but also not to ”overdesign” it to a point where it costs a fortune to maintain and forever to implement. The cost need also be kept in mind - ideally keeping the cost as low as possible to push the app to production.

4.2.1 Mobile Apps versus Web Apps

A study done very recently has revealed that there are now more mobile users than desktop users statcounter Team [2023]. However that brings up the popular debate of whether it's better to develop a web app or a mobile app. There are pros and cons to both - for both the consumer and the developer.

There are reportedly more than seven million apps available on iOS and Android platforms Mansoor [2023].

The mobile apps give advantage over performance, features and user experience. The speed of loading is increased as most of the static code lives on the device instead of having being fetched from the server like in the web apps Mansoor [2023]. That also improves the user experience. Another thing that improves user experience is the ability for apps to work offline and more importantly their ability to work in the background. Offline apps take popularity when the user is on a plane and need to kill some time. However nowadays most of the time, the user will have access to the internet - be it through Wi-Fi or Cellular Data. Working in the background becomes useful when you need to perform some calculations that take more time. For example apps like Instagram use the background running for push notifications, content pre-fetching, location services, task schedulers (like uploading a post, or getting the reactions for your post) Team [2011].

Having spoken about the pros of mobile apps, let's mention some cons. The development time takes much longer than the web app equivalent. Mobile app development is platform specific - different versions are needed for different operating systems. iOS apps are developed with a programming language called Swift, while Android relies on Java/Kotlin and XML Team [2024a] lmf [2023]. However, due to latest technologies, this too is no longer an issue. Programming languages like React Native and Flutter have been created to simply the app development process by programming for all devices with one language - JavaScript for React Native and Dart for Flutter. Of course a clear disadvantage becomes that they become a bit slower due to the extra layer of code that needs to be interpreted into native components. But it's worth mentioning that there have been studies done that came to the conclusion that while React Native apps are not as fast as its native app equivalent, they are still very fast and responsive because they use native components to create app's user interface Deshpande [2023]. Another con of mobile apps is that they take up more space on the local device, compared to the web app that lives fully on the server Shah [2023].

In conclusion, both mobile apps and web apps have their pros and cons, and should be chosen depending on the project. For the Sudoku Multiplayer app, the better choice, in my opinion, would be the mobile app, because of the performance, better and 'smoother' user experience, and device features like ability to run in the background.

4.2.2 React Native

As mentioned before, React Native is a JavaScript framework for writing real, natively rendering mobile applications for iOS and Android. Like all other JavaScript code, it needs a Virtual Machine to be run on, unlike the native code which gets executed directly on the device. It's also important noting that iOS devices have a JavaScript engine called *JavaScriptCore*, while Android devices do not, so our code will have *JavaScriptCore* attached to it. Now since Java/Kotlin (Android) and Objective-C/Swift don't communicate with JavaScript directly, they need to communicate through JSON - the *Bridge* is responsible for that. At runtime, the JavaScript code will run on the JavaScript VM, while the native code will run on the device.

When the user starts the application, the device starts three main threads - Main, JavaScript, and Shadow. Main thread is responsible for handling the front-end UI and user interaction like screen touch, scroll, zoom, and others. JavaScript thread is needed to execute the JavaScript and React code. Finally the shadow thread is for converting flexbox-based layout into a layout system that the device can understand.

Let's review the logic behind the Bridge step-by-step with an example of a button press. The process is very similar to that of a web application, for eaxmple a click on the web app sends an HTTP request to the back-end through JSON, back-end retrieves some data from the database and send it back to the front-end through JSON, and the front-end renders it to make it look nice for the user. The back-end and front-end don't understand each other's logic, but they both understand JSON. Similarly when a user clicks on a button, a message is sent from the Native side to the JavaScript side, does some logic on the JavaScript side, and then sends a response back to the Native side, which updates the view accordingly.

An important thing to note is that the Bridge is asynchronous.This is both an advantage and disadvantage depending on the situation. On the one hand, it makes the app appear quicker without hindering the ongoing view on the screen. However when rendering the view that requires a lot of computation, or dealing with a lot of updates at the same time can cause traffic, which is a problem when trying to create a complex application. Another problem is that each message sent between JavaScript and Native sides needs to be serialized and de-serealized, which is also time consuming. To solve these issues the React Native team introduced the JavaScript Interface (**JSI**) to replace the Bridge.

JSI is a general-purpose layer that's used in the JavaScript engine to make direct connection to the native APIs.This can be done synchronously by working on the same thread, or asynchronously by creating a new thread Kosmal [2022].

Some notable examples of apps coded with React Native are Facebook, Instagram, Pinterest, Airbnb, Skype, UberEats, and games like Hangman, 3D Tetris, Word with Friends Team [2024n]. More complicated games like PubG and Real Racing 3 were developed natively because the lag from communication between JavaScript and the Native sides proves too big for such complex inventions.

In conclusion, React Native is an extremely useful and reliable framework, with a big community behind it - hence I have decided to use it for this project.

4.2.3 Expo Go

Expo Go is an open-source platform that serves as a tool chain built around React Native, simplifying the process of developing and testing applications. It allows developers to write React Native code and preview the changes instantly on mobile devices or simulators. Expo Go is particularly useful for rapid prototyping and testing, offering a range of services and libraries that enhance the development experience Team [c].

When a developer writes code in the Expo environment, they can use the Expo Go app to scan a QR code that instantly loads the application on a device. This feature leverages the concept of live reloading and hot reloading, allowing developers to see changes in real-time without rebuilding the entire app. This streamlines the development process, especially during the initial stages of building and testing the user interface and functionality Team [d].

Expo Go encapsulates a variety of native APIs and provides a unified JavaScript interface to access them, which means developers do not need to write any native code. This abstraction simplifies the use of device features like the camera, geolocation, and push notifications, among others. While this can accelerate development, it also introduces limitations since developers are confined to the APIs that Expo supports. If a project requires more direct access to native APIs or third-party native modules not included in Expo, developers might need to ”eject” from Expo, transitioning to a bare React Native project Team [c].

Testing mobile applications using Expo Go offers a unique advantage as it enables developers to share the project with testers or stakeholders by simply sharing a link or QR code. Testers can then run the application on their devices without the need for building or deploying the app to a store, facilitating a more efficient feedback loop Team [c].

However, while Expo Go provides a robust platform for development and testing, it's essential to acknowledge its limitations in the context of performance and native module access. As the project's complexity grows, developers might encounter performance bottlenecks or the need for more direct interaction with the device's native layer, which might necessitate a move away from Expo Eastwood [2024].

In summary, Expo Go is a powerful ally in the React Native ecosystem, offering developers a streamlined path from development to testing, albeit with certain constraints that need to be considered depending on the project's requirements.

4.2.4 Client-Side Caching

Client-side caching is a technique of storing frequently accessed information on the user's device, so that next time it's needed, the retrieval time is magnitudes faster than fetching it from the server. This also reduces number of requests to the server, meaning lower load on the server and hence faster fetch time for all users. For web applications, we use the browser's local storage, however for mobile applications, we can go a step further and eliminate the middle man by storing information directly on the device. React Native has a very easy way of storing information on the device by calling the 'AsyncStorage' function Team [2024m].

The important thing now is to decide which information is stored in the cache. Once the user logs in, the server returns a JSON Web Token JWT - I talk about it more in Section 4.2.8. We need to store that token in order to keep the user signed in, and fetch more information securely later. So the token clearly needs to be stored locally.

Following that, after the user logs in, we fetch their details to display on the home page. Those include user's name, username, profile picture, clash record, rating, and previous five games. To reduce the loading time when the user opens the app next time (without having to log in), these should also be stored in the cache. However, there is a slight complication regarding the profile picture. As outlined in Section 4.2.11, we store the profile pictures in AWS S3. So it's not the actual image we store on the device, but rather a URI that we use to access the image. Also outlined in the section, is that these URIs have an expiration after which the URI will no longer return the image - for the sake of security. So to store the profile picture in the client-side cache, we need to store both the URI and the expiration date and time.

4.2.5 Back-end

Here I outline all the technologies and processes that are working behind the scenes, namely the programming language and frameworks running on the server, the database management system, server-side caching, and authentication.

4.2.6 Node.js vs Bun

The back-end is responsible for computations that the user doesn't see and usually lives on a server. It comprises the application's structure, data, system, and logic.

Just like on the front-end, I've decided to use JavaScript on the back-end as well because of ease of development and great speeds for frameworks like Node and Bun. Of course, as usual there are also cons - main ones being difficulties working with relational databases, very frequent updates that causes rewriting of the existent code, and possible performance issues for CPU-heavy applications due to JavaScript's single-threaded nature Winter [2023].

Node.js is a server environment that runs on V8 JavaScript engine Team [2023a]. For years now, it has been the go to JavaScript framework for writing code that lives outside of the browser. However in September 2023, a new all-in-one toolkit for running, building, and testing JavaScript, from a single file to full-stack applications, called Bun 1.0 was released J. Sumner [2023]. It was made to replace Node, while still providing the option to run Node files and npm packages in Bun. It has dragged everyone in with its impressive speed - reportedly four times faster than Node J. Sumner [2023]. While Node was built on Google's V8 engine, Bun was built on Apple's WebKit engine, which is used to power Safari - which is used by billions around the world, so we know it's reliable. Bun has made a lot of other improvements that mainly have to do with developer experience, like importing files, built in support for web APIs, hot reloading, improved install speeds, and much more.

Unfortunately because Bun is so new, it's lacking the big community that Node has, and sources believe that it will still be many more years before Node code is fully replaced by Bun. So building a Bun application right now would mean relying on documentation to solve more complicated queries rather than asking the community.

Taking all of this into consideration, I have decided to use Node as the main back-end language for this project.

4.2.7 Express.js: Enhancing Node.js for Web Servers

While Node.js provides a robust environment for running JavaScript on the server side, it often requires additional frameworks for efficient web server development. One such framework is Express.js, a minimal and flexible Node.js web application framework that provides a robust set of features for web and mobile applications.

Express.js acts as a layer built on top of Node.js that simplifies the server creation process. By using Express, developers can quickly set up middleware to respond to HTTP requests, define routing tables which prescribe how to perform different actions based on HTTP Method and URL, and render HTML pages based on passing arguments to templates Team [2024g].

One of the standout features of Express.js is its ability to simplify the process of building APIs. With Express, creating RESTful web services becomes straightforward, enabling developers to easily define routes and send/receive data in various formats like JSON. This simplification is a significant advantage when developing applications that require a back-end API to serve data to the client-side.

Express.js also shines with its middleware framework, allowing developers to perform additional tasks on the request and response objects. Middleware functions can execute code, make changes to the request/response objects, end the request-response cycle, and call the next middleware function in the stack. This feature is particularly useful for adding layers of logic such as authentication, data validation, and logging.

By integrating Express.js with Node.js, I gain access to a powerful combination for building efficient and scalable web servers. The ease of creating APIs and the streamlined handling of HTTP requests and routing makes Express.js an invaluable tool in the Node.js ecosystem, facilitating rapid development and deployment of web applications. Alternatives to Express.js include Nest.js, Fastify, Sails.js, and others. None have any serious advantage over Express, and since I have used Express before and am very familiar with the process, I have decided to use for this project Ekete [2023].

4.2.8 Authentication

As mentioned before, there are three ways of authenticating a user when it comes to my app - Google account, Apple account, and in-app authentications. Google and Apple authentications - also known as OAuth or OAuth2.0 - have become very popular in applications nowadays. That's because it simplifies the process of signing up to an application, by it omitting the longer procedure of entering all your details and verifying your email Morrow [2022]. Let's review how it works on the developer's point of view.

4.2.8.1 OAuth2.0

OAuth2.0 is a protocol that enables a website or application to access information from other web applications on a user's behalf. OAuth1.0 was used up to 2012, when OAuth2.0 replaced it due to improvements in security and performance Zhang by Okta.

OAuth2.0 uses the concept of **Access Tokens**, which are issued by the authorisation servers - Google and Apple - to the clients - users trying to sign up. Access tokens come in different formats defined by the token issuer, but the most widely used format is a JSON Web Token (JWT). I discuss JWT in Section 4.2.8.2. The resource server receives and validates the access token, and returns the necessary information requested.

Here is the outline of what happens when the user decides to sign up using their Google or Apple account:

1. The user clicks on the Google or Apple icons seen in Fig C.1(b).
2. The app requests authorisation from the authorisation server.
3. The user logs in to their Google or Apple account.
4. The authorisation server authenticates the app, and verifies that the information we are asking for about the user is permitted, i.e. the user has granted us permission to have this information.
5. The authorisation server redirects the user back to the app with the access token.
6. The app uses the access token to get the information necessary from the resource server.
7. The information is saved to the database.

The process of logging in would then look like this:

1. The user clicks on the Google or Apple icons seen in Fig C.1(a).

2. The app requests authorisation from the authorisation server.
3. The user logs in to their Google or Apple account.
4. The authorisation server redirects the user back to the app with the access token.
5. The app uses the access token to get the user's email from the resource server.
6. The app uses the email to fetch all of user's other information from the database.

4.2.8.2 In-App Authentication

The app should obviously also support authentication from manual entry of user's details. Here is the outline of the sign up process:

1. The user enters all their details and clicks submit.
2. The information is sent to the back-end using JavaScript native *ajax* event - fetch.
3. The back-end checks for duplicate entries, i.e. if there already exist accounts with that username or email.
4. If all the details are good, the back-end saves them (the password isn't saved in its raw form, rather it gets hashed first) to the database and creates a JWT.
5. The back-end sends the JWT to the front-end with the code 200.
6. The JWT is saved to the users device for client-side caching as explained in Section 4.2.4.

The log in process is as expected:

1. The user enters username and password and clicks submit.
2. The information is sent to the back-end using JavaScript native *ajax* event - fetch.
3. The back-end checks the entries.
4. If the entry exists with this username and password hash, the back-end saves them to the database and creates a JWT.
5. The back-end sends the JWT to the front-end with the code 200.
6. The JWT is saved to the users device as explained above.

4.2.9 Database

Database is an organised collection of data designed to store large amounts. I have already outlined what the app uses the database for, now we break down how to implement the database correctly.

4.2.9.1 Relational vs Non-Relational Databases

First thing to think about when designing the database architecture is to choose between relational and non-relational database.

A relational database organises the data into rows and columns, that together form a table. A database can have multiple tables. These tables can be joined together by the use of primary and foreign keys. Each column has a certain data type that is defined upon creation of the table. Another useful feature is that relational databases guarantee the ACID (Atomicity, Consistency, Isolation, Durability) properties Team [g]. To retrieve data from a relational database, you use SQL programming language. It's important to point out that each Database Management System (DBMS) has their own SQL rules, which - while similar - do differ. Some popular examples of DBMSs that use relational database structure are:

- MySQL,
- PostgreSQL,
- Oracle, and
- Microsoft SQL Server.

On the other hand, a non-relational database does not rely on table structure, but rather on various data models, such as key-value, document, column family, and graph. For our purposes, the most common data model we would use would be the document. Documents store all the information related to a single record Pawlan. Some notable examples of DBMSs that use non-relational database structure are:

- MongoDB,
- Cassandra,
- Google Cloud Firestore, and
- Amazon DynamoDB.

When comparing these two structures, we need to consider the following categories:

1. **Consistency and Integrity**
   Relational databases provide strong data consistency and integrity due to following the ACID rules outlined earlier. On the other hand, non-relational databases don't provide such guarantees, therefore it can contain different values for one key.
2. **Performance**
   Because non-relational database doesn't need to perform all the consistency checks, it can offer faster performance for specific use cases, such as big and real-time data.
3. **Data Schema**
   Relational databases are better for structured data with relationships, while non-relational databases are more flexible.

4. **Development Time and Feel**
   If not configured correctly from the start, relational databases take more time to re-structure and it takes longer to create more complex queries, while non-relational databases are easier to set up.

Pawlan

There is a lot of thoughts that goes into choosing the correct database, however let's consider the data that we need to store:

- User's name, email, username, etc.
- A user plays many games against other users.
- Each game stores 2 users, duration, winner, user rating before and after.

Clearly there is a lot of relationships among the data, and we would need to employ more complex queries that are supported by the SQL programming language. Hence I have decided to use a relational database for this application.

4.2.9.2 Database Structure

There are two main parts to save to the database: the clashes and the users. However since there is a many-to-many relationship between these two, databases 101 tells us that we need to create a third table to act as the middle man Team [h]. The final database schema can be seen in Fig D.1.

Let's review which event triggers which table in the database:

1. **User logs in or signs up:** We use the table 'Users' to either add a row to (if signing up), or read information from (if logging in).
2. **User edits something in the settings:** The 'Users' table is updated.
3. **New clash between two players:** A new row is added to 'Clashes', with the values of a new identifier 'clash_id', 'start_time' is the time at which the clash started, and 'duration' and 'winner_username' are set to *null*.
4. **A clash has ended between two players:** Firstly, we update the 'Clashes' table, setting the 'duration' and 'winner_username' to the appropriate values. Then we add two rows to 'UserClashes' - one for each of the player. There we indicate which clash this row is referring to through the foreign key 'clash_id', and the starting and ending rating of the player. We need to save the starting and ending ratings for when the player wants to see how much they've gained or lost in the Home page. Finally, we update the 'Users' table to reflects the players new ratings, and increments one of 'wins', 'losses' or 'draws'.

4.2.9.3 Database Management System (DBMS)

A simple definition of a DBMS is it's a software system used to store, retrieve, and run queries on data Team [a]. As I explained earlier, each DBMS has their own way to organising the data which results in slightly different way of using them even if they both use the relational or non-relational database structure. Each DBMS has its own advantage that separates them from the rest, for example faster read times, or faster write times, or better multi-threading support, or superior transaction management, and so on and so forth.

Relational databases have been around way longer than non-relational so there is an even bigger pool to choose from. From our discussion in Section 4.2.9.2, we can see that the application is not read nor write heavy application. So the most important thing to pat attention to would be transaction management, so we accidentally don't lose the winner of the game, or the rating change following the outcome of the game. It is for this reason I have decided to use OracleDB as my main database Team [2024j]. Apart from excellent transaction management, OracleDB provides exceptional concurrency, high availability, and comprehensive security features Team [2021a].

OracleDB has its own programming language PL/SQL, which is an extension of SQL, and lets you add Oracle programming extensions to SQL Matters [2022]. The Oracle database runs on a virtual machine, which allows for greater flexibility when it comes to resource allocation, increased security because of isolation from the rest of the application, easy snapshots and cloning - meaning easier disaster recovery process, and better cost-efficiency by allowing multiple VMs to share the same physical resources Team [2024l].

Disaster recovery procedure is one of the most important things to think about when developing an application. The worst thing you can do is let your users know that some or all of their data is gone forever Kreps [2019]. This procedure needs to be well rehearsed so if even the main database does indeed lose data, the backup data is quick to fill in the gaps Nirav [2023]. I have decided to set up an automatic snapshot of the database every day at 3am because that's my predicted time for lowest app usage. I have set up the snapshots at the VM level using RMAN and Shell scripts, and a scheduler Team [2024k]. Now when it comes to the actual recovery using these snapshots, the procedure is not quite as simple. Since the snapshots are at the VM level, I would need to restore the VM disk image, effectively rolling back the entire VM, including the database, to the snapshot state. Luckily there is a lot of support for this online and I was able to complete this procedure fully once Team [2021b].

There are some downsides to only backing up the database using snapshots. If I back it up at 3am, but something happens to production database at 3pm, there will be 12 hours of lost data - a loss that doesn't seem that bad at this stage but if the app was to be getting a new Sudoku clash a second, that would be 43,200 clash scores and rating updates lost, not to count new users information, settings updates, and so on. I think for this stage, snapshots will do the trick but later, I would need to think of a better solution, as I discuss in the Future Work section.

4.2.10 Server-Side Caching

Just like client-side caching, server-side caching is used to lower loading times, thus improving user experience. Server-side caching comes in many forms but mainly it's a database that allows reads and writes at extremely fast speeds, but foregoing certain durability measures - making it unfit for the main database Team [2016]. Cache usually stores the information that is most needed, meaning data from the database that's accessed most frequently. I have decided to store 100 most active users on the application. This number will go up the more traffic the application gets, but for now it will stay 100. I am storing user profiles for the reasons outlined in Section 4.2.13.

4.2.10.1 Cache replacement policies

Just saving all the information to the cache also isn't good, as this would drastically increase the taken storage on the disk. So it's best practice to store the data that's only accessed most frequently. When some data in the cache becomes less frequently accessed and some data outside the cache starts getting accessed more frequently, it's our job to replace them and maintain correct order. For the replacements, we need to utilise one of the cache replacement policies. I have decided to use least frequently used (LFU) as the main one, as we only need most active profiles on there Edirisinghe [2021]. We will base activity of the profile on the number of clashes the user has had. We implement this policy using cache sharding and cache-side logic Team [b].

4.2.10.2 Redis

Redis is open source, in-memory data structure store, which is one of the most popular choices for a cache. Redis offers a lot of advantages like speed and scalability, but the biggest advantage it offers that pushed me towards it is persistence. Unlike some caching solutions, Redis can persist data to disk, ensuring you don't lose your cache entirely during restarts or failures Team [2024q]. Redis is accessed through a command line interface making it easy to implement and maintain Team [2024r].

Redis supports many data structures but I will only need two of them to implement it in the way I need: sorted sets Team [2024t] and hashes Team [2024s]. The visual representation of them is shown in Fig E.1.

There are certain commands that accompany these structure. The command to add a new item to a sorted set - '$zadd$' - and remove an element - '$zrem$' - both are executed in $O(\log(N))$. Since there will be 100 elements at any time, $N = 100$, meaning adding and removing elements will be a fixed time of $O(\log(100)$. The command to set a new hash - '$hset$' - is executed in an impressive $O(1)$, while reading the hash with the command '$hgetall$' is executed in $O(N)$, where N is the fixed size of the hash of 7.

4.2.11 Cloud Storage

Storing pictures - and files in general - is a lot more efficient to do in a file management system, since those are optimised for storing and retrieving files. There are a lot of advantages that come with using cloud file management system, mainly:

- **Scalability and Availability:** cloud storage solutions expand dynamically as data grows.
- **Security:** platforms that provide the storage services develop extremely advanced encryption, versioning, and cross-region replication - far more advanced than

anything I can code out.

- **Cost-Effectiveness:** the platforms enable cost savings by allowing users to choose the most appropriate storage type based on access patterns and data criticality.

Armstrong [2023]

The only main disadvantage of using cloud storage provider instead of saving files directly to the server is speed. Since the files live on some server that isn't the app's native server, it takes extra time to fetch it from there Team [j]. But that is an extremely minor con compared to all the pros.

I will of course be using a cloud storage provider.

4.2.11.1 Amazon Web Services (AWS)

AWS is a cloud storage provider. It has over 200 fully featured services from data centers globally Team [2024f]. However we focus on storage, and it has around 13 services that offer storage, as far as I am aware Team [2024c]. AWS offers 33 regions, each with multiple availability zones. AWS also offers a free tier which is perfect for testing and just getting the application off the ground.

4.2.11.2 AWS Simple Storage Service (S3)

S3 is one of the storage solutions provided by AWS mentioned above. S3 consists of buckets. These buckets are all configured individually, where you choose the location of the server, publicity, security measures, and much more. It is possible to have many buckets in different locations, which is a very useful feature for user-base that's spread around the world. My main bucket is registered in London, which is the closest location to Edinburgh that AWS offers.

S3 is the most commonly used service for many reasons. One of them is that S3 has unlimited storage, so you will never run out Team [2024d]. Another reason is that it is incredibly durable since it provides 99.999999999% durability, and 99.99% availability of objects over a given year. You also only pay for what you use making for cost effectiveness Team [2024e]. Given all these reasons, I have decided to use AWS S3 as the main cloud storage service.

4.2.11.3 Cloud Back-Up

While incredibly durable and provides 99.99% availability of objects - as described in Section 4.2.11.2 - there is still downtime and loss of data Gawroński [2021]. It's for those reasons that companies do like to keep a back-up bucket. This back-up bucket is only used to write data into and only read from in case of a failure of the main bucket Banke [2018]. It is for that reason that the cost of maintaining such a bucket is absolutely minimal - we are talking dozens of pounds for storing terabytes of data Team [2024e]. It is also recommended that the back-up bucket is in a different location to the main one, so I have registered this one in Ireland, which is still close to the user's location. With this bucket up and running, the probability of users not being able to access the data is 0.000001%, and the probability of any data loss drops to $10^{-20}\%$.

4.2.12 GamePlay

Here is the high level overview of what happens when two users with similar ratings try to play by clicking ”New Game” on the Home page.

1. The server has a priority queue to manage player requests running on the back-end. The queue prioritises matching players based on their waiting time, aiming to match those who have been waiting longer.
2. User A sends a request over WebSocket to join the queue, and starts listening for responses over the WebSocket channel from the server.
3. User B sends a request over WebSocket to join the queue, and starts listening for responses over the WebSocket channel from the server.
4. Server code realises that the two players are compatible for a clash and pairs them together. The details of the opponent is sent to each player over WebSocket and they are both removed from the queue.
5. Server then waits for confirmation to come through over the WebSocket channel from both players’ front-end, confirming that they are still there and are ready to clash. The window to send the confirmation is 30 seconds.
6. If the confirmations come through from both, the server fetches the puzzle from an API, inserts a row into the ”Clashes” table to record the match, and sends the puzzle to the players along with the message ”BEGIN” and the id of the clash (that’s used that the id for communication over WebSocket).
   - Players then activate a 10s timer, with the message: ”The clash will start in: ...”. When the timer runs out, the players are redirected to the GamePlay page.
7. If the confirmations, don’t come through, the players are sent a message saying ”CANCEL”, at which point, the players’ front-ends are responsible for reapplying to join the queue again.

That covered the generic 99%-of-the-time case. However there can be times when the user joins the queue but there is no one in the queue with similar rating. Similar rating in this case, is within one standard deviation (300) away - I talk about this more in Section 4.3. In that case:

1. Server fetches the top 100 most active players on the platform from Redis, as discussed in Section 4.2.10.
2. They are arranged from most to least compatible according to rating, and the first 10 are considered.
3. From top to bottom, each one is sent a notification to join the clash, and they have 30 seconds to reply with a positive answer.
4. If they reply positively, the server performs the steps outlined above, and the clash begins.

5. If none of the 10 reply positively, the server returns that they were unable to find an opponent at this time, and they should try again later.

4.2.12.1 GamePlay Protocols

UDP is a lightweight data-transfer protocol used usually for time-sensitive applications like gaming (mainly video gaming) Team [f]. WebSocket is another protocol that is extremely popular. It's used in many different types of applications - most notorious examples are messaging apps. WebSocket is sometimes used over UDP because it's more reliable in many ways Team [2023c]:

- Fail safe mechanisms that allow WebSocket connection to reopen after detecting an unexpected closure.
- It's lossless, i.e. no data is lost in the process.
- It's still very fast - utilising full-duplex communication channels over a single TCP connection.

Due to UDP being blocked by most firewalls Ali [2011], and it takes a lot of effort to bypass that - mainly by integration with other protocols, I have decided to not use UDP, but rather rely solely on WebSocket.

4.2.13 Final Architecture

The above can be concisely summarised in the diagram shown in Fig F.1.

4.3 Rating System

As outlined in Section 2.2.3, the rating change depends on the following five factors: complexity of the puzzle, possession of the board at the end of the clash, the duration of the clash, difference of players' rating before the Sudoku clash, and number of puzzles solved before the clash. So the task is to find the weights **w** - a vector - such that at the end of each clash, each of the factors are multiplied by their respective weight:

$$\mathbf{R}_{new} = w_0 + w_1 \cdot f_1 + w_2 \cdot f_2 + \cdots + w_5 * f_5 = \textstyle\sum_{i=0}^{5} w_i \cdot f_i,$$

where $f_0 = 1$, $\mathbf{R}_{new}$ is the player's new rating, and f_i are the five factors mentioned above. We currently have 6 weights - 5 weights for the factors and a constant term, however as I justify in Section 4.3.3, we will only need 5 weights. The task is to make the player rating distributions as close to the normal distribution as possible.

Reinforcement learning is needed to calculate the weights.

4.3.1 Reinforcement Learning

Reinforcement learning is a machine learning technique. Just like humans learning from consequences of their actions, adapting to achieve better outcomes, reinforcement learning has an agent that learns to make decisions by interacting with an environment of York. The goal is to maximise reward or to minimise punishment. Reinforcement learning usually follows the same five steps:

1. **Observation:** The agent observes the current state of the environment.
2. **Decision:** Based on the observed state, the agent makes a decision or selects an action following a policy (a strategy that maps states to actions).
3. **Action:** The agent performs the action.
4. **Reward and Next State:** The environment responds with a new state and a reward (or penalty).
5. **Learning:** The agent updates its policy based on the received reward and the transition to the new state.

Simplilearn [2023]

4.3.2 Steps

The following are the steps taken to compute the weights of the Sudoku player rating algorithm:

1. Define number of generations, which is the number of times the weights will go through mutation.
2. Generate 5,000 players with **true** ratings from a normal distribution. Initialise their calculated ratings.
3. We define the reward/punishment to be the measure of how closely our model produces ratings to the true ratings for each player after simulating 100 games amongst each other. We call this process the ”evaluation of fitness of the weights”.
4. It’s up to me to choose the mean and standard deviation. I went for mean $\mu = 1700$ and s.d. $\sigma^2 = 300$. The consequences of having standard deviation much smaller than the mean is that 68% of players will be within the 1400 to 2000 range, which is a good situation for this situation.
5. Each generation has its own population. The population is a set of individuals - in this case weights. The first generation starts off with a completely random population. For every generation, we are going to have 5,000 players play 100 games each against random opponents. Each player has a true rating - a rating that comes from a normal distribution - and a game rating - this rating is set to 800 to all new players. This stage is called ”simulation” and I talk about it more here. In the simulation phase, we predict the outcome of the game using the players’ true ratings. We then calculate both players’ new rating using the weights passed in. We repeat this entire process for every set of weights in the population.
6. The population changes with every generation. We evaluate the fitness of every weight set in the population. Then they undergo selection, where we pick the best-performing set of weights based on fitness.

7. The we perform crossover, where we combine weights from pairs of best-performing sets to create new sets.
8. Finally, we introduce mutation, where we slightly modify the weights to introduce variation.
9. Then it's the next generation and we repeat everything.
10. The best performing weights are chosen at the end of all the generations.

4.3.3 Simulation

There are a lot of things to consider when trying to simulate 5,000 players playing each other. Firstly, we need to understand that no matter how hard we try, we'll never be able to mimic the human actions, but we can try. Let's simulate a game between Player1 and Player2. Player1 has true rating of p_1 and game rating of g_1, while Player2 has true rating of p_2 and game rating of g_2.

We begin by creating the concept of a "bad game". Each player can have a bad game - that's part of being human. However we also need to realise that the better you are, the less likely you are to have a bad game. So the probability of a player having a bad game is inversely proportional to their true rating. We use the following formulas to calculate the probability:

$$\mathbb{P}[\text{Player1 has a bad game}] = \tfrac{100}{p_1}$$
$$\mathbb{P}[\text{Player2 has a bad game}] = \tfrac{100}{p_2}$$

Definition (Probability Factor)**.** Before we get further into calculations, we need to define a new term - **probability factor**. Probability factor is a number between 0 and 1, which tells us what portion of the board the player is likely to capture, meaning fill in first.

First scenario is when **both players either have a good or a bad day**. In this case the player with the highest rating will win with 100% probability. Let's say Player1 has the higher rating, then

$$\mathbb{P}[\text{Player1 wins}] = 1$$
$$p1_{pf} = \tfrac{p1}{p1+p2},$$

where $P1_{pf}$ is the Player1 probability factor.

Second scenario is when one player has a bad day and another player has a good day. Without loss of generality, let's assume Player1 has a good day and Player2 has a bad day. Then we break it down into three scenarios:

1. $\mathbf{p_1 > p_2}$:

$$\mathbb{P}[\text{Player1 wins}] = 1$$
$$P1_{pf} = 1.2\tfrac{p_1}{p_1+p_2}$$

2. $\mathbf{p_1 == p_2}$:

$$\mathbb{P}[\text{Player1 wins}] = 1$$

$$P1_{pf} = 1.05\frac{p_1}{p_1+p_2} = 1.05\frac{p_1}{2p_1} = \frac{1.05}{2}$$

3. $\mathbf{p_1 < p_2}$:

$$\mathbb{P}[\text{Player1 wins}] = \begin{cases} \frac{100}{(p_2-p_1)^2} & \text{if } p_2 - p_1 > 10 \\ 1 & \text{otherwise} \end{cases}$$
$$W_{pf} = 1.2\frac{p_w}{p_1+p_2},$$

   where W_{pf} is the winner probability factor and p_w is the rating of the winner (whoever the winner is based on the probability).

The predicted number of squares that the winner will fill in is then calculated as:

$$W_s = \lceil \mathbf{FS} \cdot W_{pf} \rceil,$$

where W_s is the number of winner squares and **FS** is the number of free squares at the start of the clash. For the sake of this computation, we will set this equal to a fixed amount of 64. As mentioned in source, the minimum number of clues required to make the Sudoku puzzle have a unique answer is 17. Hence $81 - 17 = 64$.

The **predicted duration** depends on two things: world average Sudoku solving time and the winner's rating. The average solve time is approximately 20 minutes Green [2021]. So the predicted duration is then calculated as:

$$t_p = \frac{20 \cdot 60 \cdot \mu}{p_w} = \frac{20 \cdot 60 \cdot 1700}{p_w}$$

There are four types of complexities of the puzzles: easy, medium, hard, and expert. We denote these by using 1, 2, 3, and 4 respectively to give them a numerical value. I decided to start from 1 and not 0, so that when it is easy, it still counts for something.

Now we need to think about how to express the function responsible for rewarding/punishing players based on their game ratings and the outcome of the game - let's denote it as $\omega(g_1, g_2, P1_s, P2_s)$. For example, if Player1 has game rating lower than Player2 but fills in more squares, then Player1 should be awarded quite well. Player2 in that scenario should therefore be punished quite harshly for that. On the other hand, if Player1 has a lower game rating and loses, then they shouldn't be punished very badly, and Player2 shouldn't be celebrated too well for that win either. After some trial and error, I have arrived at the following expression for ω:

$$\omega(g_1, g_2, P1_s, P2_s) = 10^{\delta(g_1,g_2)\frac{(g_1-g_2)(p_1-p_2)}{6400}},$$

where

$$\delta(g_1, g_2) = \begin{cases} -1 & \text{if } g_1 < g_2 \\ 1 & \text{if } g_1 > g_2 \end{cases}$$

As we can see, it follows all the constraints we have set on it.

All the results have been predicted. All that is left to do is calculate the players' new ratings:

$$\begin{aligned} \mathbf{R}_{p1_{\text{new}}} &= w_0 + w_1 \cdot \mathbf{PC} + w_2 \cdot t_p + w_3 \cdot \mathbf{GP}_{p1} + w_3 \cdot \omega(g_1, g_2, P1_s, P2_s) \\ \mathbf{R}_{p2_{\text{new}}} &= w_0 + w_1 \cdot \mathbf{PC} + w_2 \cdot t_p + w_3 \cdot \mathbf{GP}_{p2} + w_4 \cdot \omega(g_2, g_1, P2_s, P1_s), \end{aligned}$$

where $\mathbf{PC} \in \{1,2,3,4\}$ is the puzzle complexity, $P1_s, P2_s$ are the number of squares filled in by both players such that $P1_s + P2_s = 64$, and $\mathbf{GP}_{p1}$ and $\mathbf{GP}_{p2}$ are the number of games played before the clash by Player1 and Player2 respectively.

4.3.4 Result

After running Reinforcement Learning using the steps outlined over 100 generations, the weights were as follows:

$$\mathbf{w} = \begin{pmatrix} 0.94660185 \\ 0.75045995 \\ 1.10087091 \\ 0.89018092 \\ 0.80039273 \end{pmatrix}$$

To demonstrate the correctness of the weights, I ran a script that creates 5,000 players from normal distribution as before. They all play 100 games against random opponents and then the game ratings frequency are plotted as a histogram. The graph in Fig G.1 shows 10 of these tournaments, and overlayed in red is the actual distribution that the player ratings were created from. The average mean of the 10 tournaments is 1672 with average standard deviation of 310, which is close to the original values of 1700 mean and 300 stadard deviation. So we have shown that the algorithm performs to a high standard.

Chapter 5

Testing

5.1 User Interview Results

The feedback collection from users was once again conducted anonymously to avoid the feedback being untruthful. The interviewees and I needed to be in the same geographical location for this test since in order to get the application on their devices, my laptop and their phone needed to be connected to the same network Team [2024o]. There was a possibility to run the application on the simulator Team [2024p], but I thought the users would get a better feel if they played on their physical devices. The interview transcript can be seen in Appendix H.

Similar to the previous feedback collection, users were asked to rank the statements from completely disagree (1) to completely agree (10). There were 8 statements so the maximum mark a person could give is 80. The average mark received was 78, which is unbelievably high. The points lost in almost every feedback were statements (d) and (g). The interviewees let me know in the comment section that they would have wanted to know who they were playing against before the clash - mainly to see their rating and understand if the game will be hard or easy, and also in the EndGame screen, they wanted to know how many points they have gained or lost after the clash. Luckily, these changes are easy to implement. Now, before the clash begins, the users get at least ten seconds to see their opponent. A screenshot of this screen in seen in Fig I.1.

Overall, the user testing went very well and the results came back positive with many asking to keep them updated on when the app will be released fully.

Chapter 6

Evaluation, Future Work, and Conclusion

6.1 Evaluation

Table 6.1 shows the feature list from Appendix D but this time indicates whether the feature has been accomplished or not. As can be seen all of the core features have been accomplished, with one feature being accomplished partially. As explained in the table, when a user creates their account by signing up with Google or Apple, there is further planning that needs to happen for when the user wants to change their email. However, if the user has signed up through the app, there is no problem in changing the email there. All the other setting option work as planned. Furthermore, I have managed to implement Google and Apple authentication, which was a feature in the additional feature list. The other two additional features have not been implemented as of yet. This is a good indication that the app was built successfully feature wise.

Now returning back to the main goal I mentioned in Chapter 1, this app was built to raise the awareness of Sudoku and appeal to people who have previously shied away or never tried it because they believed it was boring or difficult. I have mentioned in Chapter 3 that because of this goal, Group 2 is the people who's opinion I want to change about Sudoku, so they have been monitored very carefully. That's why another reason for success of the application was that every person in Group 2, replied wit either 9 or 10 for statement (f): ”The log in and sign up process contained everything I needed and was extremely easy to navigate”, implying they enjoyed the competitive twist the app has provided to an otherwise lonely and long game. Having a competitor made the game have more stakes and made the game last shorter, so it was nice quick bursts of competitive fun.

Whilst making the game more fun for new Sudoku players, it also maintained its popularity among the regular player, which is good since they are the ones that will probably be responsible for most games on the platform. Unsurprisingly, from the feedback collected, Group 1 found the added concept of a rating to be very fun. For this group, I was beaten by every single one and the average completion time for all of them was around 2 minutes, so they found it to be equivalent to a sprint rather than a longer

Feature	**Accomplished**
Creating and deleting an account	Yes.
Logging in and logging out	Yes.
Settings	Partially. If the user created their account through the app, there is no problem changing the email, but if their signed up through Google or Apple, it gets trickier if the user wants to change their email. Further Research required.
Viewing games played	Yes.
Viewing user statistics	Yes.
Starting a new game	Yes.
Playing the game	Yes.
Toggle between note mode and play mode	Yes.
End game screen	Yes.
Google and Apple Authentication	Partially. Only Google authentication was implemented at this time.
Pick their own opponent	No.
Single-player mode	No.

Table 6.1: List of features and whether they have been accomplished or not.

paced marathon.

Group 1 was also very engaged, and some people walked away absolutely loving the game of Sudoku, and are now playing single player Sudoku just for practice.

In conclusion, the application turned out to be a great success. I discuss potential work to still be done before releasing the application to the public in the next section, but for the first product it seems to have validated primary market fit and validated the idea overall.

6.2 Future Work

The main after this project is to make the application ready for release. Due to a high standard that games are held to nowadays, we need to review some limitations of the app.

6.2.1 Limitations

There are a couple of features I believe the app should have in order to bring it to the standard of an application that can capture a market:

1. **Single-Player Feature**: The app should have the ability practice Sudoku solving skills on a single player.

2. **Multiplayer AI Feature**: The user should be able to practice multi-playing without affecting their rating. I think they should be able to pick the ability level of the AI opponent as well.
3. **Choose the ability level of their opponent**: Inspired by chess.com, the user should also be able to choose the rating range of their opponents.
4. **Make it web accessible**: This is not an immediate feature but by making the game accessible online and hence on the laptop too is an important step in ensuring wider user outreach Wondrasek [2021].
5. **Make it more social**: Finally, I think the platform should have a social element to it. You should be able to connect to people, chat about Sudoku, play against your friends. That's what really drives the popularity of applications Motta [2023] Project [2023] Trepte et al. [2011]. This feature is not immediate and will take time to plan out and execute properly, but at the end of the day that's how the game gains a whole new level of traction and importance.

6.2.2 Technical Improvements

Now let's discuss technical improvements on the application. Right now the app works fine and is able to handle a certain amount of users. However, it is not configured to its full capacity. Here are some technical improvements I want to implement in the near future:

1. **Database Connection Pooling**: Database connection pooling is a technique used to manage database connections in a cache, allowing multiple applications or users to share a pool of connections. It's needed to optimise resource usage, reduce the overhead of establishing connections repeatedly, and enhance application performance. Implementing connection pooling typically involves configuring a pool manager to handle connections' lifecycle, including creation, borrowing, return, and expiration. By reusing existing connections, applications can execute database operations more efficiently, improving throughput and reducing latency, especially in high-concurrency environments Custer [2021].
2. **Better database back-up**: While database snapshots are useful for capturing the state of a database at a specific point in time, relying solely on them for backups has limitations. Alternative strategies include using distributed databases, which can offer enhanced resilience and availability. Distributed databases store data across multiple physical locations, reducing the risk of data loss due to hardware failure or catastrophic events. Moreover, implementing regular backup procedures, such as full, differential, and incremental backups, provides a comprehensive approach to data protection, allowing for more flexible and efficient data recovery options. Additionally, leveraging cloud-based backup solutions can offer scalability, reliability, and improved disaster recovery capabilities Bindu [2023] Team [2024i].
3. **Optimising cloud file storage through multiple medians**: To optimise AWS for file storage, we can use S3 lifecycle policies to transition older or less frequently

accessed data to S3 Infrequent Access (IA) or S3 Glacier for lower storage costs. For active EC2 instances requiring less frequent access to certain data, we can utilise Elastic Block Store (EBS) snapshots, storing these snapshots in S3, and retrieving them when necessary. Implement Amazon CloudFront to cache data closer to users, reducing access times and improving efficiency is also a great way to improve user experience Team [2024b].

6.3 Conclusion

During this study, I have identified a problem with the current Sudoku games, by that identifying a gap in the market, designed and executed a solution to this problem, and validated the solution with potential users. The aim of this Sudoku multiplayer application was to engage more people in the game of Sudoku by making it more competitive, quick, and fun, and redefine a lot of myths to do with it. Based on feature criteria and user feedback, the application was successful. However as outlined in the Future Work section, the application still needs improvements to capture the masses necessary for a platform like this to succeed.

Bibliography

Firkhan Ali Bin Hamid Ali. A study of technology in firewall system. In *2011 IEEE Symposium on Business, Engineering and Industrial Applications (ISBEIA)*, pages 232–236, 2011. doi: 10.1109/ISBEIA.2011.6088813.

E Armstrong. File management: more beneficial than you think. 2023. URL `https://nordlocker.com/blog/file-management/`.

F. Jarvis B. Felgenhauer. Maths of sudoku i. 2006.

K. Balasubramanian. The rise of online multiplayer. 2023. URL `https://www.gameopedia.com/online-multiplayer-games`.

F Banke. Amazon s3 backup strategy. 2018. URL `https://medium.com/@frederikbanke/amazon-s3-backup-strategy-45ba510f7850`.

C Bindu. Distributed system: Db backup and restore strategy. 2023. URL `https://medium.com/@bindubc/distributed-system-db-backup-and-restore-strategy-f24c163c8beb`.

Auth0 by Okta. What is oauth 2.0? URL `https://auth0.com/intro-to-iam/what-is-oauth-2`.

J Bycer. The importance of the new player's experience. 2020. URL `https://www.gamedeveloper.com/business/the-importance-of-the-new-player-s-experience/`.

C Custer. What is connection pooling, and why should you care. 2021. URL `https://www.cockroachlabs.com/blog/what-is-connection-pooling/`.

W Daher. The best way to validate your startup idea. 2022. URL `https://waseem.substack.com/p/the-best-way-to-validate-your-startup`.

C. Deshpande. Flutter vs. react native: Which one to choose in 2023? 2023. URL `https://www.simplilearn.com/tutorials/reactjs-tutorial/flutter-vs-react-native`.

Tia Eastwood. Expo vs bare react native...the pros and cons. 2024. URL `https://www.tiaeastwood.com/expo-vs-bare-react-native-the-pros-and-cons`.

K S Edirisinghe. Server-side caching in web applications. 2021. URL `https://medium.com/codex/server-side-caching-in-web-applications-a9145be1cfa0`.

D Ekete. 5 alternatives to express.js, 2023. https://www.makeuseof.com/expressjs-alternatives/.

World Puzzle Federation. World sudoku championships official page. 2022. URL https://www.worldpuzzle.org/championships/wsc/.

Old School Games. What percentage of people play sudoku? URL https://osgamers.com/faq/what-percentage-of-people-play-sudoku.

W Gawroński. The complete history of aws outages. 2021. URL https://awsmaniac.com/aws-outages/.

Mark E Glickman and Albyn C Jones. Rating the chess rating system. *CHANCE-BERLIN THEN NEW YORK-*, 12:21–28, 1999.

M Green. How i used alteryx to solve sudoku puzzles 4,000 times faster. 2021. URL https://www.phdata.io/blog/how-i-used-alteryx-to-solve-sudoku-puzzles/.

P Hornby. The mom test – why you should never ask your mother if your idea is any good. URL https://www.productfocus.com/the-mom-test/.

S Ingram. The thumb zone: Designing for mobile users. 2016. URL https://www.smashingmagazine.com/2016/09/the-thumb-zone-designing-for-mobile-users/.

C. McDonnell J. Sumner, A. Partovi. Bun 1.0. 2023. URL https://bun.sh/blog/bun-v1.0.

A. Kauranen. How colours guide the player in video games. 2023.

J. Kosmal. How does react native work? understanding the architecture. 2022. URL https://medium.com/front-end-weekly/how-does-react-native-work-understanding-the-architecture-d9d714e402e0.

D Kreps. Myspace lost all music uploaded to site prior to 2015. 2019. URL https://www.rollingstone.com/music/music-news/myspace-lost-music-809455/.

Rusyaizila R Kristo RP. A rapid solo software development (rssd) methodology based on agile. 2022.

Rhyd Lewis. Metaheuristics can solve sudoku puzzles. *J. Heuristics*, 13(4):387–401, 2007. doi: 10.1007/s10732-007-9012-8. URL https://doi.org/10.1007/s10732-007-9012-8.

lmf. Build your first android app in java. 2023. URL https://developer.android.com/codelabs/build-your-first-android-app#0.

I. Mansoor. App download data (2023). 2023. URL https://www.businessofapps.com/data/app-statistics/.

Ionos Team Technical Matters. Oracle database: what it is and how it works. 2022. URL https://www.ionos.co.uk/digitalguide/hosting/technical-matters/oracle-database/.

S Morrow. Third-party authentication (oauth): Good or bad for security? 2022. URL https://www.infosecinstitute.com/resources/industry-insights/third-party-authentication-oauth-good-or-bad-for-security/.

M Motta. Social gaming: When games become social networks for gen z and millennials. 2023. URL https://www.adjust.com/blog/social-gaming-gen-z/.

K Nirav. Gitlab dev deletes entire production database. 2023. URL https://medium.com/@kanani-nirav/gitlab-dev-deletes-entire-production-database-719756f4a2ce.

University of Edinburgh. School of informatics - research ethics. 2024. URL https://www.inf.ed.ac.uk/research/ethics/code.html.

University of York. What is reinforcement learning? URL https://online.york.ac.uk/what-is-reinforcement-learning/.

A. Pal. Solving sudoku game using a hybrid classicalquantum algorithm. *Europhysics Letters, Volume 128, Number 4*, 2020.

A. Patteri. Color design in game development. 2020.

D Pawlan. Relational vs. non-relational database: Pros cons. URL https://aloa.co/blog/relational-vs-non-relational-database-pros-cons.

BBC The Life Project. How online gaming has become a social lifeline. 2023. URL https://www.bbc.com/worklife/article/20201215-how-online-gaming-has-become-a-social-lifeline.

M. Shah. Mobile app development process: Step-by-step guide for 2023. 2023. URL https://www.invonto.com/insights/mobile-app-development-process/.

H. Simonis. Sudoku as a constraint problem. *Imperial College London*, 2005.

Simplilearn. What is reinforcement learning: A complete guide. 2023. URL https://www.simplilearn.com/tutorials/machine-learning-tutorial/reinforcement-learning.

Alliance Software. An introduction to software development methodologies. 2024. URL https://www.alliancesoftware.com.au/introduction-software-development-methodologies.

NineTwoThree Staff. Ux/ui design principles in modern mobile applications. URL https://www.ninetwothree.co/blog/ux-ui-design-principles-in-modern-mobile-applications.

statcounter Team. Desktop vs mobile vs tablet market share worldwide. 2023. URL https://gs.statcounter.com/platform-market-share/desktop-mobile-tablet.

T Taylor. How to determine product market fit in your industry. 2022. URL https://blog.hubspot.com/sales/product-market-fit.

Apple Team. Swift. 2024a. URL https://developer.apple.com/swift/.

AWS Team. Amazon s3 storage classes. 2024b. URL https://aws.amazon.com/s3/storage-classes/.

AWS Team. Cloud computing with aws - amazon web services. 2024c. URL https://aws.amazon.com.

AWS Team. Amazon simple storage solution service (s3) faqs. 2024d. URL https://aws.amazon.com/s3/faqs/.

AWS Team. Amazon s3 simple storage solution service pricing. 2024e. URL https://aws.amazon.com/s3/pricing/.

AWS Team. What is aws? - cloud computing with aws - amazon web services. 2024f. URL https://aws.amazon.com/what-is-aws/.

CherryRoad Consultancy Team. What are the top 8 reasons to use an oracle database? 2021a. URL https://www.cherryroad.com/2021/10/22/oracle-database-cloud/.

Cisco Team. What is database management systems (dbms)? a. URL https://www.appdynamics.com/topics/database-management-systems.

Dragonfly Team. How can you build a distributed lru (least recently used) cache? b. URL https://www.dragonflydb.io/faq/how-to-implement-distributed-lru-cache.

Expo Go Team. Expo go documentation. c. URL https://docs.expo.dev/.

Expo Go Team. Expo go documentation faqs. d. URL https://docs.expo.dev/faq/.

Express.js Team. Express. fast, unopinionated, minimalist web framework for node.js. 2024g. URL https://expressjs.com/.

Figma Team. What is figma? 2024h. URL https://www.figma.com/.

Flowmatters Team. Why is user interface design important for user engagement? e. URL https://www.flowmatters.com/blog/why-is-user-interface-design-important-for-user-engagement/.

Fortinet Team. User datagram protocol (udp). f. URL https://www.fortinet.com/resources/cyberglossary/user-datagram-protocol-udp.

Google Cloud Team. About cloud sql backups. 2024i. URL https://cloud.google.com/sql/docs/mysql/backup-recovery/backups.

IBM Team. What is a relational database? g. URL https://www.ibm.com/topics/relational-databases.

IBM Team. Restoring sql databases from virtual machine snapshots. 2021b. URL https://www.ibm.com/docs/es/tsmfve/7.1.1?topic=protection-restoring-sql-databases-from-virtual-machine-snapshots.

Instagram Engineering Team. What powers instagram: Hundreds of instances, dozens of technologies. 2011. URL https://instagram-engineering.com/what-powers-instagram-hundreds-of-instances-dozens-of-technologies-adf2e22da2

Microsoft Team. Create many-to-many relationships. h. URL https://support.microsoft.com/en-gb/office/video-create-many-to-many-relationships-e65bcc53-8e1c-444a-b4fb-1c0b8c1f5653.

Node.js Team. About node.js. 2023a. URL https://nodejs.org/en/about.

Oracle Team. Oracle database. 2024j. URL https://www.oracle.com/uk/database/.

Oracle Team. Scripting oracle rman commands. 2024k. URL https://blogs.oracle.com/connect/post/scripting-oracle-rman-commands.

Oracle Team. About virtual machine db systems. 2024l. URL https://docs.oracle.com/en-us/iaas/base-database/doc/virtual-machine-db-systems.html.

PubG Team. Frequently asked questions about ranked mode. 2023b. URL https://support.pubg.com/hc/en-us/articles/360048004774-Frequently-Asked-Questions-about-Ranked-Mode.

React Native Team. Asyncstorage - react native. 2024m. URL https://reactnative.dev/docs/asyncstorage.

React Native Team. Who is using react native? 2024n. URL https://reactnative.dev/showcase.

React Native Team. Running on device. 2024o. URL https://reactnative.dev/docs/running-on-device.

React Native Team. Running on simulator. 2024p. URL https://reactnative.dev/docs/running-on-simulator-ios.

Redis Team. Introduction to redis. 2024q. URL https://redis.io/docs/about/.

Redis Team. Redis cli. 2024r. URL https://redis.io/docs/connect/cli/.

Redis Team. Redis hashes. 2024s. URL https://redis.io/docs/data-types/hashes/.

Redis Team. Redis sorted sets. 2024t. URL https://redis.io/docs/data-types/sorted-sets/.

Roelof Jan Elsigna Team. The importance of server-side caching. 2016. URL https://roelofjanelsinga.com/articles/the-importance-of-server-side-caching/.

Sudoku.com Team. What actually happens at a sudoku world championship? i. URL https://sudoku.com/how-to-play/what-actually-happens-at-a-sudoku-world-championship/.

Suite Team. Why cloud storage downloads are slow. j. URL https://www.suitestudios.io/post/why-cloud-storage-downloads-are-slow.

Synopsys Editorial Team. Top 4 software development methodologies. 2017. URL https://www.synopsys.com/blogs/software-security/top-4-software-development-methodologies.html.

Wallarm Team. Websocket protocol. 2023c. URL https://www.wallarm.com/what/a-simple-explanation-of-what-a-websocket-is.

Sabine Trepte, Leonard Reinecke, and Keno Juechems. The social side of gaming: How playing online computer games creates online and offline social support. *Computers in Human Behavior*, 28:832–839, 12 2011. doi: 10.1016/j.chb.2011.12.003.

YouGov UK. Sudoku activity. URL https://yougov.co.uk/topics/society/explore/activity/Sudoku.

R. Winter. Javascript for backend development: An introduction. 2023. URL https://blog.hubspot.com/website/java-backend.

J Wondrasek. Should you build an app or a website? 2021. URL https://www.softwareseni.com/should-you-build-an-app-or-a-website/.

T Zhang. Oauth 1.0 vs oauth 2.0. URL https://www.loginradius.com/blog/engineering/what-is-the-difference-between-oauth1-and-oauth2/.

Appendix A

Questions asked competitive chess players

1. How do you perceive the significance of your chess rating in your competitive journey?
2. Do you strategically select opponents based on their chess rating?
3. Based on your observations, when competing against opponents with a lower rating, is the outcome predominantly in your favour?
4. In your opinion, does a player's rating directly correlate with the anticipated duration until checkmate, and the extent of dominance exerted by the higher-rated player?
5. Would you favour competing against lower-rated players for a higher probability of victory, or challenging higher-rated players for potential rating improvement despite a decreased win probability?
6. Do you believe that the chess rating system should incorporate additional metrics such as board possession, remaining pieces at game's conclusion, or the number of moves leading to checkmate?

Appendix B

Questions asked during the UI desgn interviews

Before asking any questions, the interviewee was given the designs and asked to look at them for as long as they needed. After they were done, the questioning began.

1. Please rank the following statements on a scale from 1 to 10, 1 - being I completely disagree with the statement, and 10 - being I completely agree with the statement.
   (a) The app design has met my expectations.
   (b) The rules of the game are easy to grasp.
   (c) The tone of the app is friendly.
   (d) I could understand how to navigate the app very easily.
   (e) The app has a very nice visual appeal to it.
   (f) The actual game page is easy to understand.
   (g) The app's appeal aligns perfectly with its purpose.
   (h) This design encourages me to use the app regularly.
   (i) I would recommend this app to others purely based on design.
2. Are there any other comments you would like to leave regarding any part of the application - not just the design.

Appendix C

UI Design

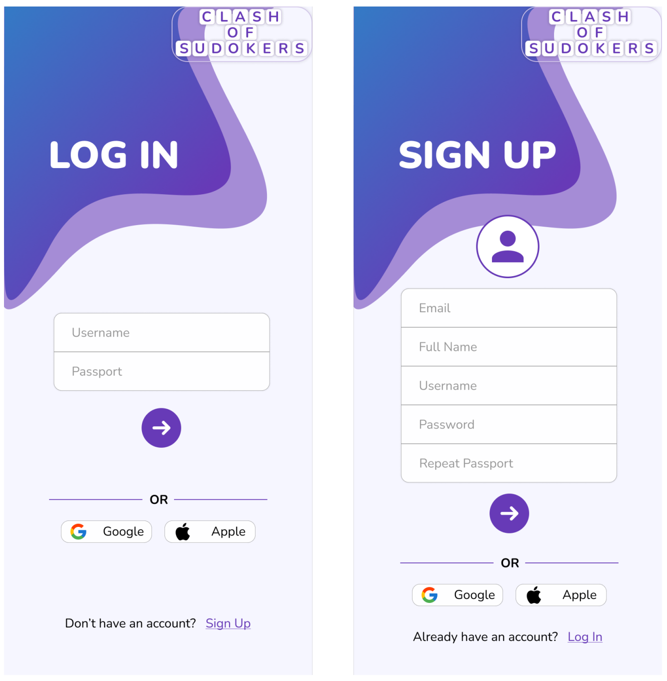
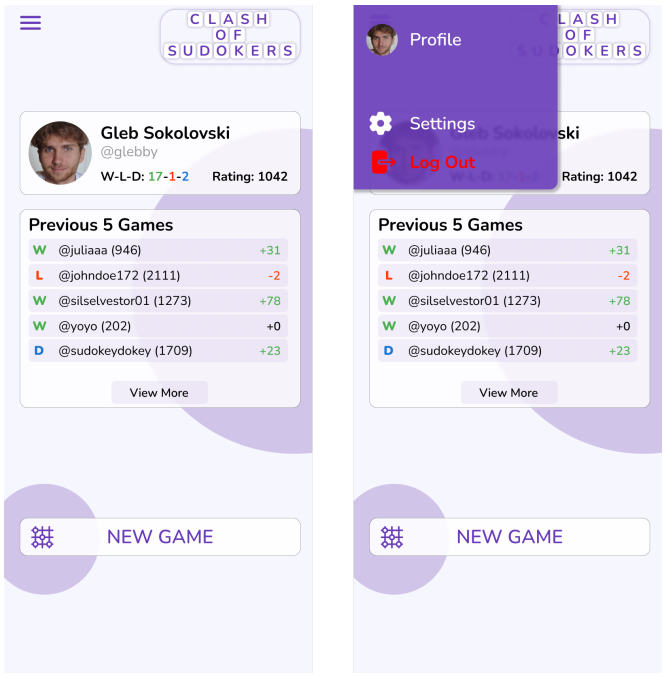


Figure C.1: (a) Log In Screen and (b) Sign Up Screen

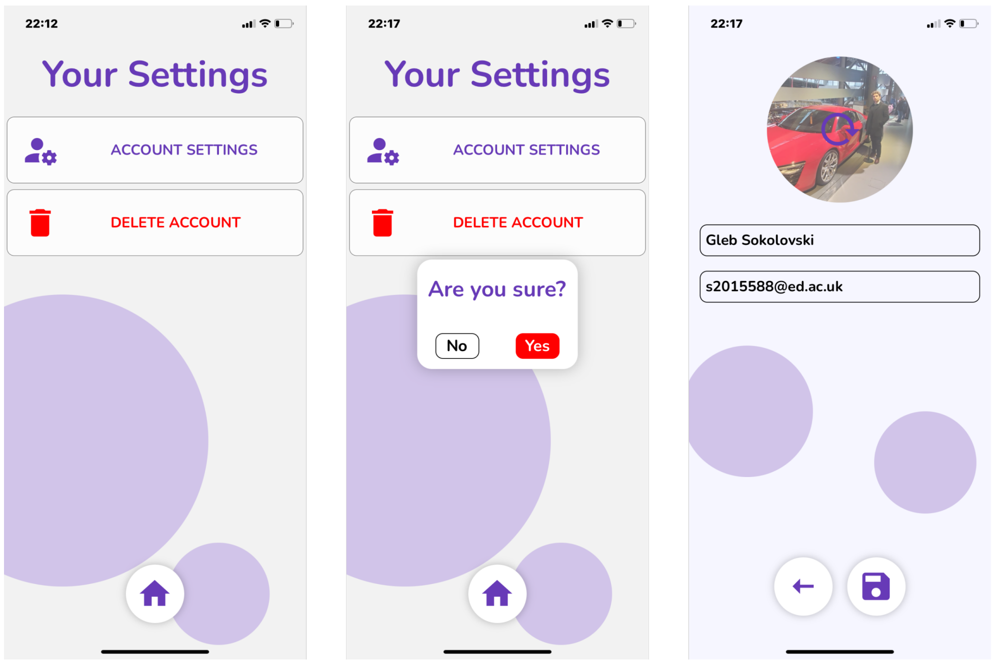


Gleb Sokolovski
@glebby
W-L-D: 17-1-2 Rating: 1042

Previous 5 Games

W	@juliaaa (946)	+31
L	@johndoe172 (2111)	-2
W	@silselvestor01 (1273)	+78
W	@yoyo (202)	+0
D	@sudokeydokey (1709)	+23

Figure C.2: Home Screen: The screen on the right demonstrates the menu bar.



Gleb Sokolovski
s2015588@ed.ac.uk

Figure C.3: (a) Settings Page (b) Delete Account Clicked (c) Account Settings Clicked

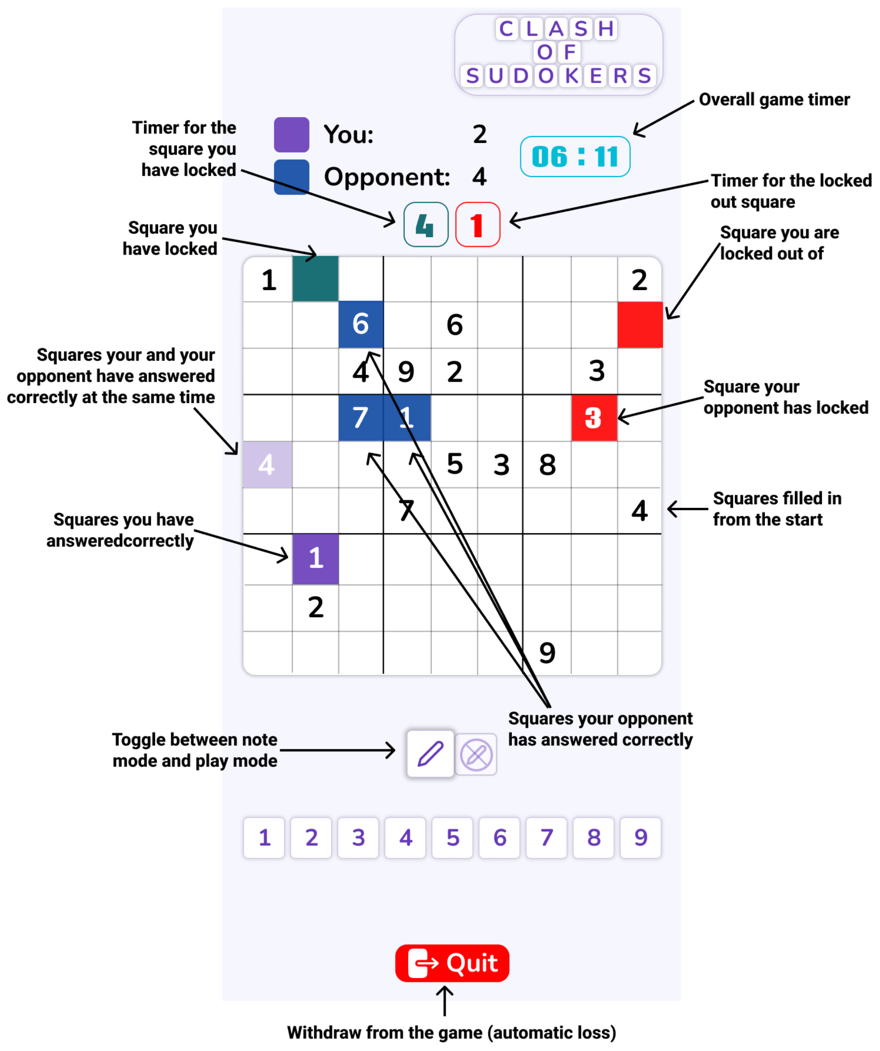

CLASH OF SUDOKERS
You: 2
Opponent: 4
06 : 11
4
1
Overall game timer
Timer for the square you have locked
Timer for the locked out square
Square you have locked
Square you are locked out of
Squares your and your opponent have answered correctly at the same time
Square your opponent has locked
Squares filled in from the start
Squares you have answeredcorrectly
Squares your opponent has answered correctly
Toggle between note mode and play mode
1 2 3 4 5 6 7 8 9
Quit
Withdraw from the game (automatic loss)

Figure C.4: Game Page annotated

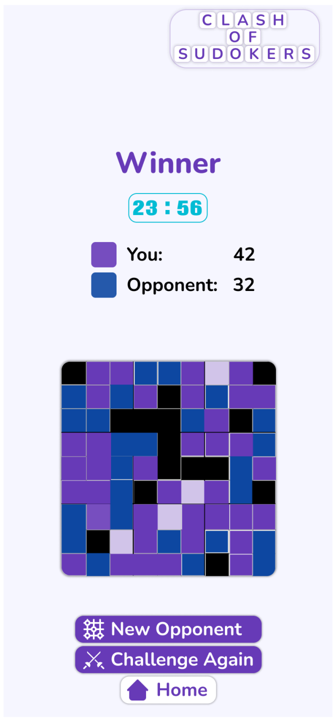
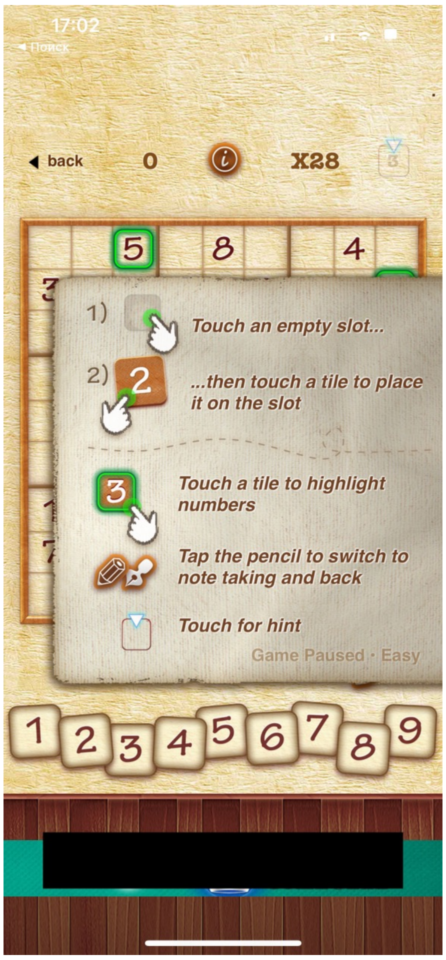


Figure C.5: End Game Page



1) Touch an empty slot...

2) ...then touch a tile to place it on the slot

Touch a tile to highlight numbers

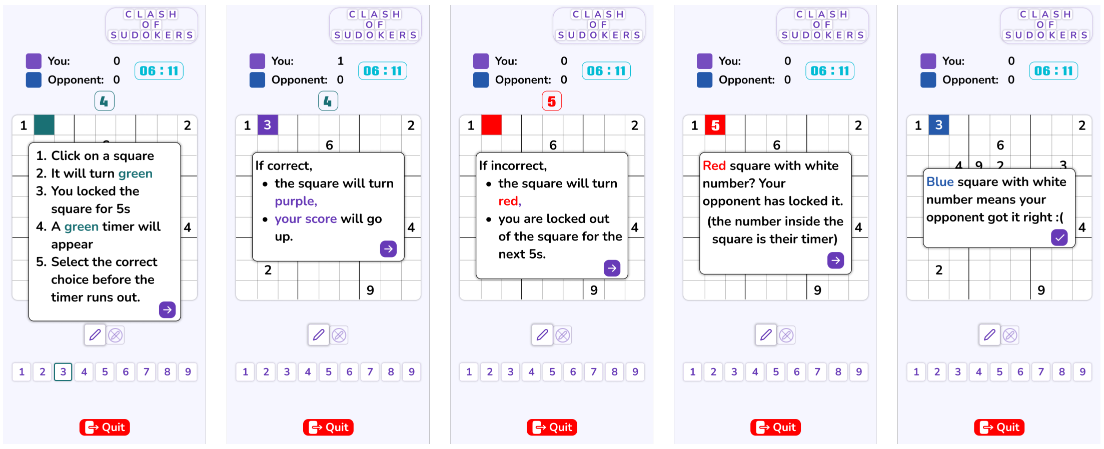
Tap the pencil to switch to note taking and back

Touch for hint

Game Paused · Easy

[redacted]

Figure C.6: Example of a tutorial in a Sudoku app



1. Click on a square
2. It will turn green
3. You locked the square for 5s
4. A green timer will appear
5. Select the correct choice before the timer runs out.

If correct,
- the square will turn purple,
- your score will go up.

If incorrect,
- the square will turn red,
- you are locked out of the square for the next 5s.

Red square with white number? Your opponent has locked it.
(the number inside the square is their timer)

Blue square with white number means your opponent got it right :(

Figure C.7: Design of the Tutorial for newcomers

Appendix D

Database Schema

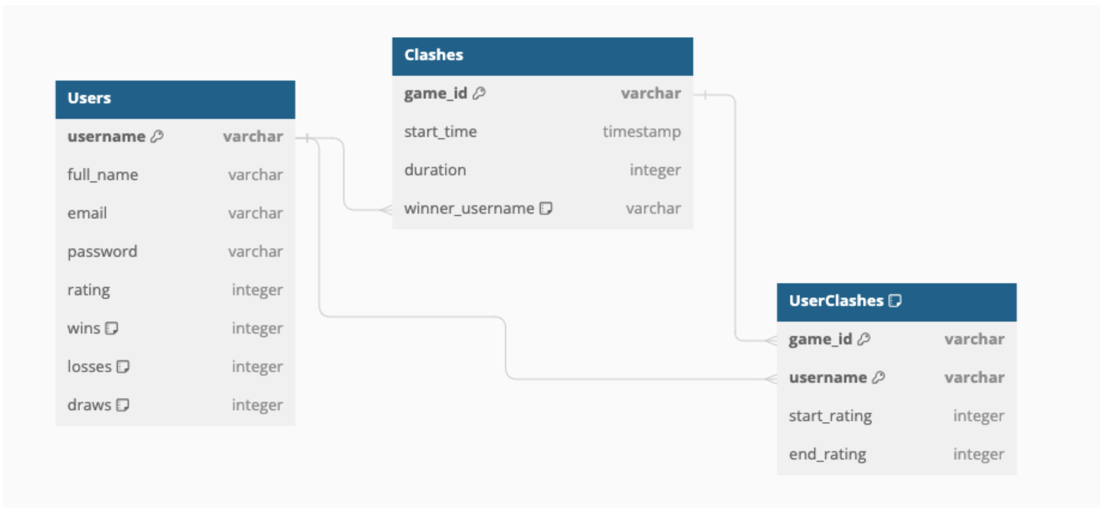


Figure D.1: Database Schema

Appendix E

Visualisation of Redis Data Structures

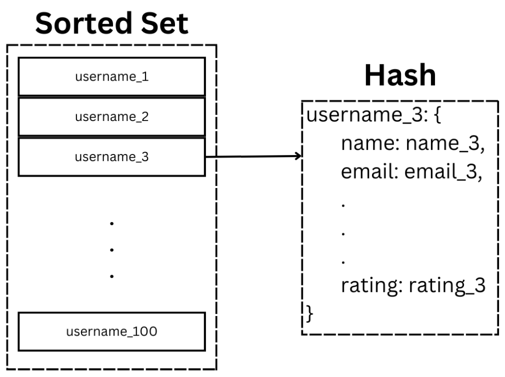


Figure E.1: Redis data structures

Appendix F

Final Architecture

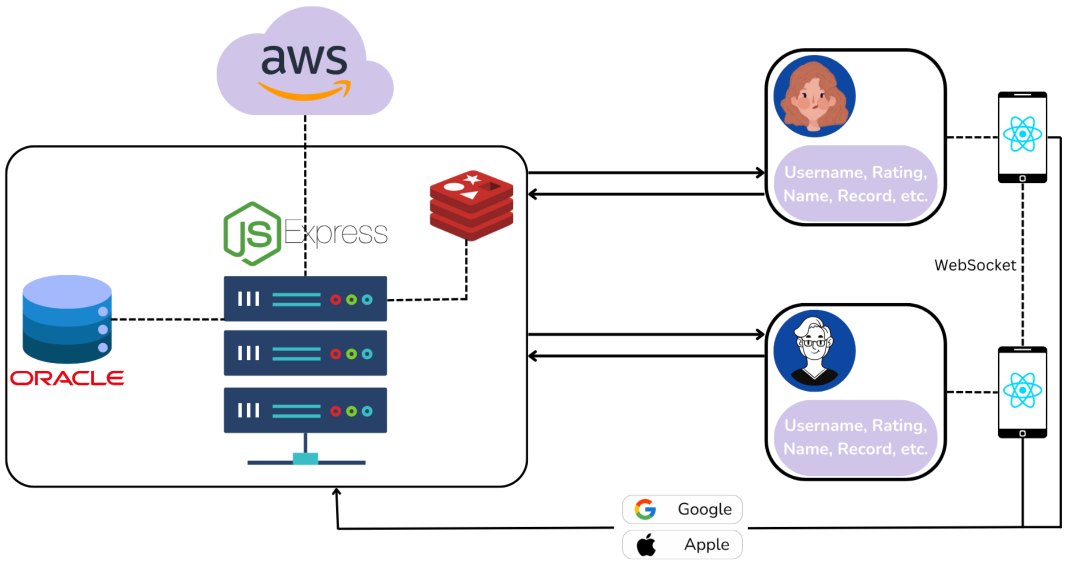

aws
Express
ORACLE
Username, Rating, Name, Record, etc.
WebSocket
Username, Rating, Name, Record, etc.
Google
Apple

Figure F.1: Final App Architecture

Appendix G

Sudoku Player Rating Algorithm Correctness

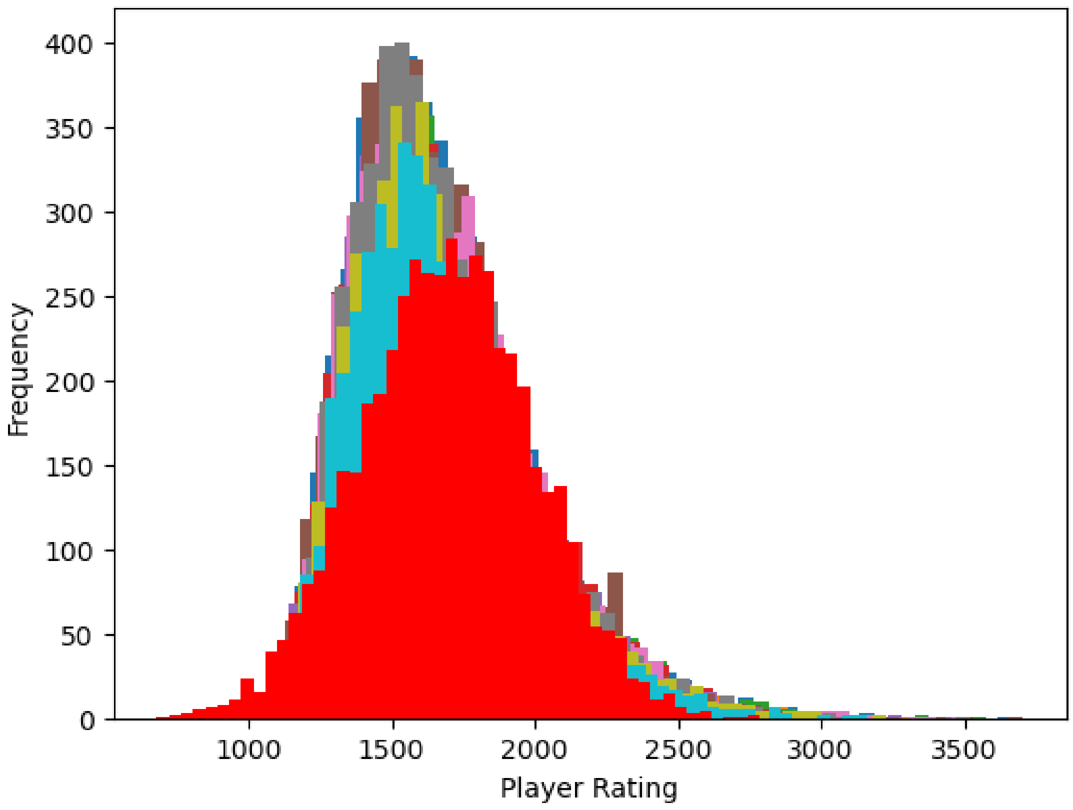


Figure G.1: 10 Sudoku tournaments of 5,000 players playing 100 games each against randomly selected opponents. Their end game rating is plotted as a histogram do demonstrate distribution. The histogram in red is the true rating distribution assigned to players at the start.

Appendix H

Application Testing User Interview Transcript

Part 1: The setup
First, set up the application on the user's device. Inform them that it will require downloading an application from the App Store. The application is lightweight - taking up around 90MB, and it can be deleted straight after the interview. It is your job to download the application, connect it to the laptop, and delete it afterwards. The users don't have any input during this stage.

Part 2: Running the Application and Familiarisation
Start the application through Expo Go and hand the phone to the user. They should be greeted with the Sign Up page. Nudge them to create an account. Really highlight that their details and their account will be deleted straight after the experiment and they will be able to re-register when the app is live on the App Store.

After the registration, ask them to explore different features of the app. Ask them to execute some actions like:

- Try to change your profile picture.
- Try to log out and log back in.
- Try to return to home page.

These actions are to see that the navigation is very clear and to make the user familiarise themselves with the app.

Part 3: Clash
Now ask the player to start a new game. They will join the queue to be paired up with an opponent. Join the queue as well so you are paired up together. After getting paired

up, the clash should begin. Play the game with the user. At the end, let them examine the EndGame page and the changes made to the Home page, like a game has been added to their history, the rating and their game statistics have been changed. This is the end of the interview. Thank the user for their time and give them a chocolate bar.

Part 4: Feedback Form After all the users tried out the app, make the feedback form active on the website. The questions that are asked are as follows:

1. Please rank the following statements on a scale from 1 to 10, 1 - being I completely disagree with the statement, and 10 - being I completely agree with the statement.
   (a) The actual application was the same/very similar to the design I was shown earlier.
   (b) The log in and sign up process contained everything I needed and was extremely easy to navigate.
   (c) The navigation of the app was simple and very clear.
   (d) The app had all the functionality I expected it to have.
   (e) It became very clear how to play the game and how to win.
   (f) The game itself was fun and engaging.
   (g) The EndGame page was clear and contained all the information I needed.
   (h) I would recommend this application to my friends.
2. Are there any other comments you would like to leave?

Appendix I

Clash Screen

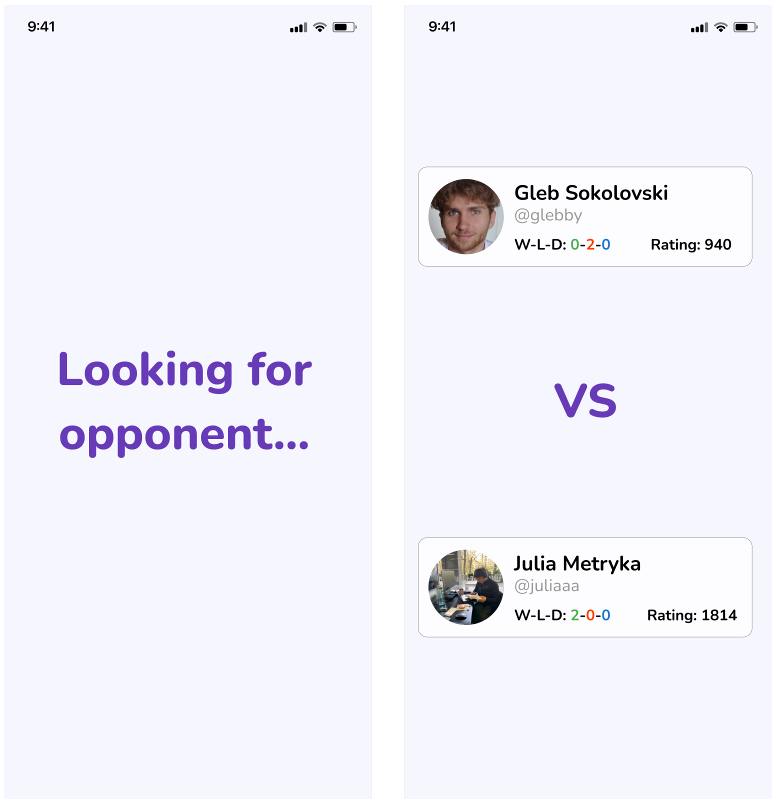


Figure I.1: Clash screen the users see before the clash is about to begin.

Appendix J

Participant Information Sheet

Project Title:	Sudoku Multiplayer App
Principal Investigator:	Nigel Topham
Researcher Collecting Data:	Gleb Sokolovski
Funder (if applicable):	No

This study was certified according to the Informatics Research Ethics Process, reference number 617018. Please take time to read the following information carefully. You should keep this page for your records.

Who are the researchers?
Gleb Sokolovski – University of Edinburgh UG4 Computer Science and Mathematics S2015488.

What is the purpose of the study?
The purpose of the study for my Sudoku Multiplayer App project is to evaluate and enhance the user experience and functionality of the app across different user groups. By involving participants with varying levels of familiarity with Sudoku (experts, occasional players, and novices), the study aims to gather diverse insights and feedback at multiple development stages (initial design, first prototype, and subsequent prototypes). This iterative feedback process is intended to optimise the app's design, usability, and appeal to a broad audience, ensuring it meets the needs and expectations of all user categories.

Why have I been asked to take part?
You have been asked to take part in this study because your experience and perspective as a player (or non-player) of Sudoku are valuable for the development of the Sudoku Multiplayer App. Your participation, whether as an experienced Sudoku player, an occasional player, or a complete novice, will provide essential insights during various stages of the app's development. Your feedback will contribute significantly to improving the design, functionality, and overall user experience of the app, making it more appealing and accessible to a diverse range of users.

Do I have to take part?
No – participation in this study is entirely up to you. You can withdraw from the study at any time, up until the study is published, without giving a reason. After this point, personal data will be deleted and anonymised data will be combined such that it is impossible to remove individual information from the analysis. Your rights will not be affected. If you wish to withdraw, contact the PI. We will keep copies of your original consent, and of your withdrawal request.

What will happen if I decide to take part?

Data Collection: You will be asked questions related to your experiences and opinions on various aspects of the Sudoku Multiplayer App. This may include your thoughts on the app's design, usability, features, and overall enjoyment.

Methods of Collection: The data will be collected through a combination of methods, including questionnaires and individual interviews. The questionnaires will be structured to gather specific information, while the interviews will allow for more in-depth discussions.

Duration: Each individual interview is expected to last approximately 30 minutes to 1 hour. The time commitment for filling out questionnaires will vary, but they are designed to be concise and straightforward.

Recording: There will be no recording present during interviews.

Frequency and Scheduling: You will be asked to participate in sessions at key stages of the app's development – after the initial design, the first prototype, and subsequent prototypes. The exact dates and times will be scheduled in advance and communicated to you, providing flexibility to accommodate your availability.

Location: All interactions will take place either online or at a comfortable place for both parties, allowing you to participate from any location convenient to you.

Are there any risks associated with taking part?
There are no significant risks associated with participation.

Are there any benefits associated with taking part?
You get to be a part of the building process of one of a kind Sudoku app and upon completion, you may receive a thank you note.

What will happen to the results of this study?
The results of this study may be summarised in published articles, reports and presentations. Quotes or key findings will be anonymised: We will remove any information that could, in our assessment, allow anyone to identify you. With your consent, information

can also be used for future research. Your data may be archived for a maximum of 4 years. All potentially identifiable data will be deleted within this timeframe if it has not already been deleted as part of anonymisation.

Data protection and confidentiality.
Your data will be processed in accordance with Data Protection Law. All information collected about you will be kept strictly confidential. Your data will be referred to by a unique participant number rather than by name. Your data will only be viewed by the researcher/research team: Gleb Sokolovski (researcher), Nigel Topham (supervisor), and Rob van Glabbeek (second marker).

All electronic data will be stored on a password-protected encrypted computer, on the School of Informatics' secure file servers, or on the University's secure encrypted cloud storage services (DataShare, ownCloud, or Sharepoint) and all paper records will be stored in a locked filing cabinet in the PI's office. Your consent information will be kept separately from your responses in order to minimise risk.

What are my data protection rights?
The University of Edinburgh is a Data Controller for the information you provide. You have the right to access information held about you. Your right of access can be exercised in accordance Data Protection Law. You also have other rights including rights of correction, erasure and objection. For more details, including the right to lodge a complaint with the Information Commissioner's Office, please visit www.ico.org.uk. Questions, comments and requests about your personal data can also be sent to the University Data Protection Officer at dpo@ed.ac.uk.

Who can I contact?
If you have any further questions about the study, please contact the lead researcher, Gleb Sokolovski, on his email: s2015488@ed.ac.uk, or by telephone: +447774161703. If you wish to make a complaint about the study, please contact inf-ethics@inf.ed.ac.uk. When you contact us, please provide the study title and detail the nature of your complaint.

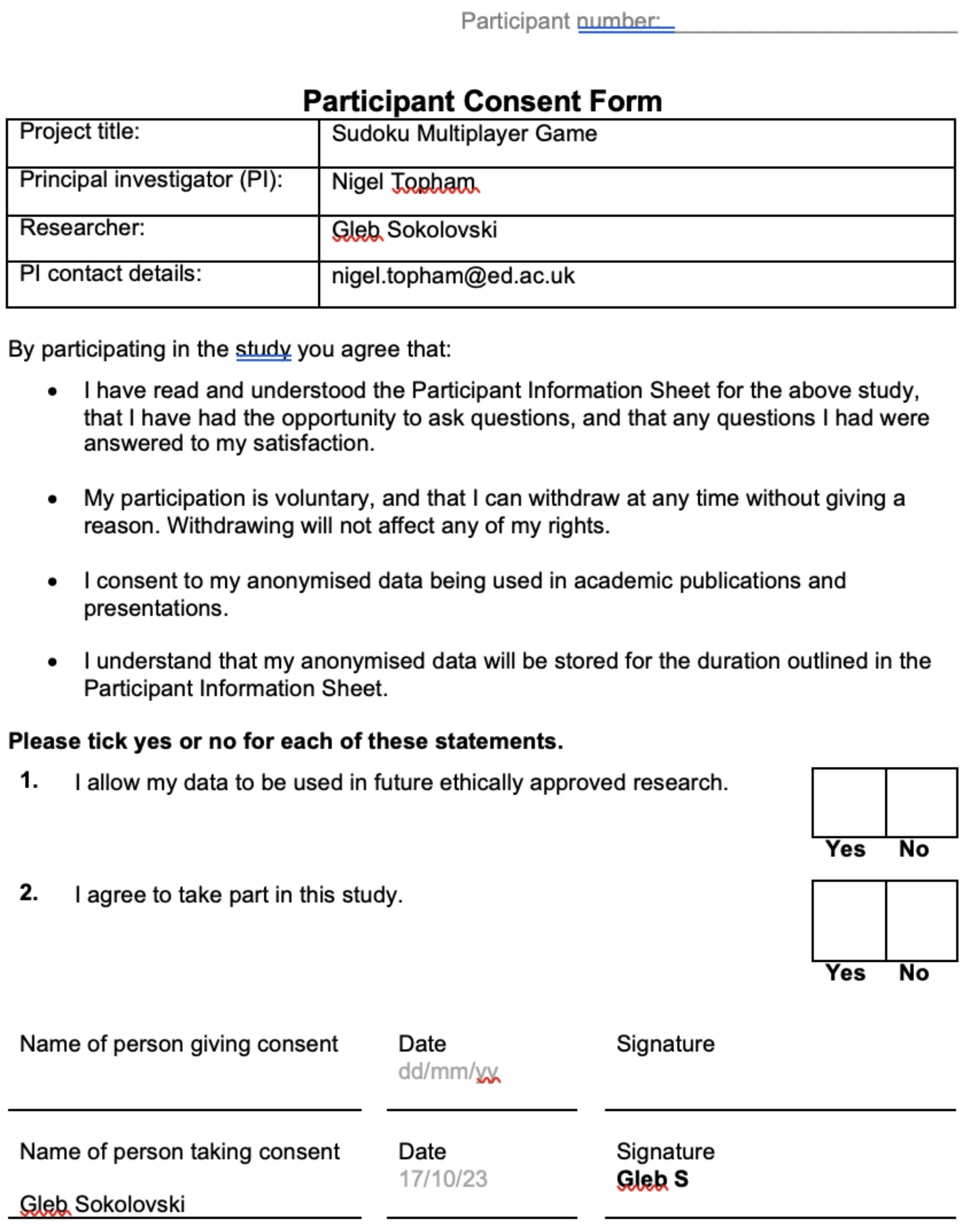
Updated information.
If the research project changes in any way, an updated Participant Information Sheet will be made available on `http://web.inf.ed.ac.uk/infweb/research/study-updates`.

Alternative formats.
To request this document in an alternative format, such as large print or on coloured paper, please contact the lead researcher, Gleb Sokolovski, on his email: s2015488@ed.ac.uk, or by telephone: +447774161703.

General information.
For general information about how we use your data, go to: `edin.ac/privacy-research`

Appendix K

Participant Consent Form



Participant number:

Participant Consent Form

Project title:	Sudoku Multiplayer Game
Principal investigator (PI):	Nigel Topham
Researcher:	Gleb Sokolovski
PI contact details:	nigel.topham@ed.ac.uk

By participating in the study you agree that:

- I have read and understood the Participant Information Sheet for the above study, that I have had the opportunity to ask questions, and that any questions I had were answered to my satisfaction.
- My participation is voluntary, and that I can withdraw at any time without giving a reason. Withdrawing will not affect any of my rights.
- I consent to my anonymised data being used in academic publications and presentations.
- I understand that my anonymised data will be stored for the duration outlined in the Participant Information Sheet.

Please tick yes or no for each of these statements.

1.	I allow my data to be used in future ethically approved research.	**Yes**	**No**
2.	I agree to take part in this study.	**Yes**	**No**

Name of person giving consent	Date dd/mm/yy	Signature
Name of person taking consent Gleb Sokolovski	Date 17/10/23	Signature **Gleb S**

Figure K.1: A form every participant had to fill in before participating in my study.