Development of a Full Stack Chess Tournament Web Application

Tesfaye Asfaw

Haaga-Helia University of Applied Sciences

Bachelor of Business Information Technology

Bachelor's Thesis

2022

Abstract

Author(s)
Tesfaye Asfaw

Degree
Bachelor of Business Information Technology

Report/Thesis Title
Development of a Full Stack Chess Tournament Web Application

Number of pages and appendix pages
50 + 1

Abstract
The goal of this thesis project is to develop and deploy an open-source full stack chess tournament web application. This web application consists of three components: frontend, backend, and a database. The frontend is the web page on the internet that users directly interact with. The backend is hidden from the user. It acts like a middleman between the frontend and the database. The database is where all the users' data is stored. When the application is deployed, it would enable users to organize chess tournaments.

This thesis report starts by discussing some of the best theoretical software development methodologies. It then describes the various tools that the author used in developing this web application. The justifications of choosing these tools over their alternatives were also one of the points of the theoretical discussions.

In the empirical section, the author shows the development process. First, the initial stage of this project is clearly stated. Then the general interactions among the user, frontend, backend, and the database are outlined. After this, the author discusses in depth about each component's development processes.
In the end, the web application was deployed on the internet where any users can sign up and start using it free of charge.

Key words
Full stack, web application, React, Spring boot, Redux Toolkit

Contents

1 Introduction

My previous work began as a simple experiment to explore pairing players in a Swiss-system chess tournament, but over time it evolved into a full-stack web application. In 2021, I incorporated ideas learned from a course at Haaga Helia University of Applied Sciences (UAS), Server Programming, to add a data presentation mechanism for the backend of the project. Further details about this previous work can be found in section 3.1.

During this thesis project period, I have implemented all the tasks listed in the requirements outlined in section 1.2. As a result, the website application can now be accessed by any user over the internet to run a chess tournament. The target user groups for this application are tournament organizers, chess coaches, and users with limited budgets.

My motivation for choosing this thesis project is that I believe it will help me achieve my aspirations of becoming a professional full-stack developer.

1.1 Actual Case

The idea for this project originated in 2020 while observing an actual Swiss-system chess tournament event where the recording of pairings and game results were done manually. During the event, I noticed that the organizers had difficulties conducting the tournament manually. I asked why they did not use software to facilitate this task and learned that the lack of availability or high cost of such tools were the reasons for conducting the tournament manually. This observation led to the potential need for a web app to make this type of event proceed more smoothly and efficiently.

In this thesis project, I will be using tools such as HTML5, JavaScript, React library, Redux Toolkit, Visual Studio Code IDE, PostgreSQL, Java's Spring Boot framework, IntelliJ, and Axios API. A brief description of these tools can be found in section 2.5. Depending on time constraints, I will try to add more features to this project such as additional tournament types that the software can handle in order to make the app more appealing to potential customers and increase the chances of them choosing to use my web application instead of a commercial one."

1.2 Requirements

Organizing a chess tournament requires a proper recording and storing or persisting of the resulting data. A tournament organizer needs to keep a record of the players information and associated tournament data. The app enables users to save data on a database.

Depending upon the type of the tournament at hand, a tournament organizer must perform the correct pairing of players at each round. This helps to improve the quality of the tournament organization and satisfaction of the tournament players.

Users have the option of selecting players, from the list of all players, who are going to participate in the upcoming tournament.

Users only see their own data.

At the end of the tournament, the tournament organizer may have the choice of saving the entered data or just print out the results in a sorted and tabulated manner which is appealing to end users.

The software can be used by multiple users where each user's data is kept separate. This makes the app reusable and accommodate as many users as possible to use the app without any conflict with other users.

The application can store current tournament data in cases of an accidental closing of the web browser that the app runs in or temporary interruption of internet access.

The web application is deployed on the internet and any user can have access to it.

The web application incudes documentations that enable new users to learn how to use it.

1.3 Scope and Limitations

This web application enables users to create user account. Users can add players to the database by filling out the players info on a form. The web app can save the players data. After conducting the tournament the user has an option to save the tournament data. Users can also view their own previously conducted tournaments.

This web app is not commissioned by any party. Any financial costs incurred is covered by the author. This project is done solely by the author. There are time constraints to add any features other than those listed in section 1.2.

2 Theoretical Background

2.1 Introduction

In this theoretical section, I will discuss the theoretical concepts of the traditional software development lifecycle and the various software development methodologies. These processes are fundamental to the success or failure of a project, so I give due emphasis on this area in this section.

Then, I briefly discuss about the tools that I use to build this web application. I am not going to go in depth but analyse those specific features of these tools that I apply in the actual project.

2.2 Brief History of Web Applications

The history of web applications cannot be seen as separate from that of the history of the Web or the World Wide Web. Web applications cannot exist without the Web. The development of the Web has propelled the rise in popularity of web applications.

Beginning from 1950 onwards, software programmes began the slow transition from the limited uses in the academic and military world into the business world (Jones 2013).

The frontend technology mainly consisted of static files - HTML and CSS. These were rigid and non-responsive webpages. But in 1995 with Netscape's introduction of JavaScript, a frontend scripting language, web developers were able to embed this new scripting language in the user's browser. (Haverbeke 2018).

With JavaScript developers were able to handle many common activities in the user's web browser. These were tasks like validation, rendering new forms, etc avoiding otherwise the many requests that would have been sent to the web browser to handle each task.

The first milestone in the history of web application is the 1999 first time introduction of the "web application" concept by Sun Microsystems. Sun Microsystems introduced this concept for the release of its version - Java Servlet Specification, v2.2. (Davidson & Coward 1999, 43-46).

Furthermore, in 2005 the introduction of asynchronous JavaScript and XML (AJAX) was also another milestone in the history of web applications (Hoffmann 2019).

The introduction of HTML5, in 2011 also added new important features that broaden the functionalities of web application. It enabled the use of graphics and multimedia without the need to use any scripting languages in the frontend. (Freeman 2011).

These are the five main new features added by HTML5: video tags, application memory, canvas element, geolocation and web workers. (Freeman 2011).

2.3 Software Development Lifecycle

Introduction

To produce anything, one needs to follow a procedure or guide to at least produce a semblance of the desired qualities. For example, to bake a cake, I have to know besides the ingredients, tools needed, the right recipe to produce high quality cake. Otherwise, the end product may not be what I had hoped for. It could also lead to wastes in resources and time.

Likewise developing a software needs to have a roadmap. This helps to save time, add efficiency, less errors and better quality.

Dooley (2011) mentioned about the “Code and Fix” model. This is technique programmers use when learning programming languages to practice and sharpen their skills. As Figure 1 shows, it practically means identify problem, code, and run. If there is error faced, fix it and run it again. And this is repeated until all errors are fixed without doing the important tasks like testing etc.

Dooley (2011) sates that this leads to dangerous outcomes and suggests it be used only for personal learnings. I, myself learned programming languages this way. I totally agree with J. Dooley that this is not a model at all. It cannot be used for commercial projects as it lacks the common tasks involved during a SDLC.

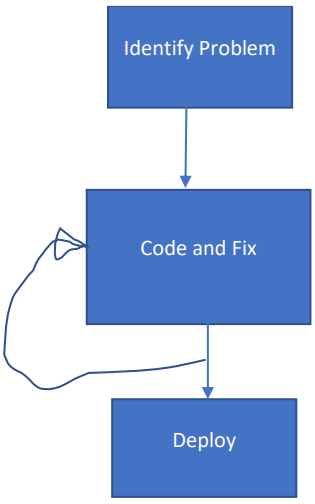

Figure 1. Code and Fix (adaptive from Dooley 2011).

Software Development Lifecycle (SDLC) is an important concept in software development. It was one of the earliest models used in the history of software development. It provides a general framework that a typical software development project follows to achieve a high-quality product. (Dooley 2011).

It provides the common phases and tasks involved in a typical software development process. SDLC helps to outline a sense of direction in the development process (Fishpool & Fishpool 2020).

In my understanding, many software development methodologies are in one way or the other has their roots based on SDLC. I think it is a great starting point to devise one's own methodology. But in my opinion SDLC is too slow and it also does not give room for feedbacks from the stake holders.

Later in section 2.4, I discuss the modern categories of software methodologies: incremental, iterative, and agile methodologies (with the focus on agile method-scrum). I used a mixture of techniques from these three methodologies for the development of my own web application.

That is why it is so important for me to go through the theoretical backgrounds of these methodologies.

The modified figure below here shows the common phases of SDLC.

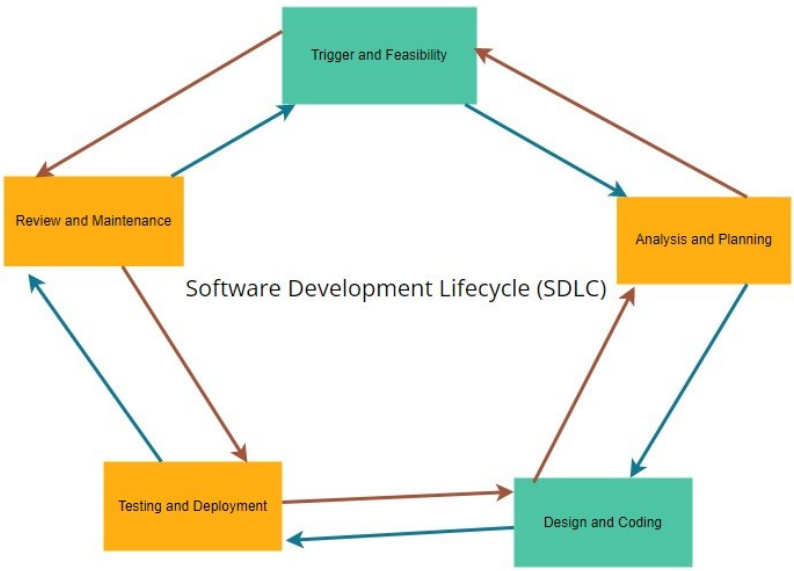

Figure 2. Software Development Lifecycle Phases (adaptive from Fishpool & Fishpool 2020, 24).

Figure 2 displays the phases of the SDLC model namely: trigger, feasibility, analysis, planning, design, development, testing, deployment, maintenance, and review phases (Fishpool & Fishpool 2020).

The arrangement of the phases inside Figure 2 is a sample illustration that I picked to show all the phases involved in SDLC. But variants of the grouping of phases than the one shown in Figure 2 is evident in many different books.

Some of the phases listed together in the figure could be separated or concisely be grouped together rather than as separate phases. I think it is up to the project manager, depending on the nature of the project, to determine the exact sequences.

Now I would like to go through each phase of SDLC and discuss briefly what kinds of activities are expected to be done in each phase.

Trigger and Feasibility

This is the starting point of any SDLC process. It triggers the whole SDLC cycle. Basically, it is the recognition of a problem or a whole new idea which is selected to undergo a further feasibility analysis. The trigger usually comes from the management, stakeholders, or customers. Once the trigger is identified, it is checked whether the trigger is worthy enough for the analysis phase. One factor that can be used to do this is ROI (return on investment). A low ROI would result in the trigger to a halt. (Dooley 2011).

Fishpool and Fishpool (2020, 23-24) state that if the trigger passes ROI test, then the management analyses all the viable options to solve the problem like:

- Buy a readymade solution
- Adapting an existing software
- Build a new software

Analysis

If a decision is taken to proceed with building a new system to tackle the problem, the next step is then the analysis phase. In this phase, the development team makes a thorough analysis of the problem. (Fishpool & Fishpool 2020, 25).

Planning

According to Fishpool and Fishpool (2020), this phase of SDLC is sometimes skipped but that this is not right. It is in fact s an important phase of SDLC.

Design

Fishpool and Fishpool (2020, 25) state that in this phase planning is done for the development phase which consists of:

- a graphical depiction of the system or UI (user interface).
- a detailed description of the functionalities of each unit in the UI which can also be supported by using pseudocodes
- a detailed outline of the backend including the database design- ER diagram and etc

Development

This is the phase where the coding takes place. Its duration is usually considerably longer than the other phases.

Testing

It is a general fact that before a software could be deployed, it must undergo rigorous testing. Untested software is a guarantee for failure.

Companies follow many different timings to do the testing. Fishpool and Fishpool (2020, 25) point out that some choose to do it even during the development phase, others after the development phase is over and some right before shipping to customers.

Deployment

Now that the testing is done, the new system is ready to be deployed to customers.

However, customers or companies still should consider the risks associated with switching to new systems. There can occur unforeseen problems which can lead to the failure of the new system, data loss etc.

To mitigate such kinds of risks, companies must navigate the various deployment options.

Fishpool and Fishpool (2020, 25-27) states that there are four major ways of deployment:

- Big Bang: the straightforward immediate switching off the current system and adoption of the new one. However, this is a highly risky practice. For example, if some problems are encountered with the new system, then the entire company’s operations could be disrupted.

In my opinion, this may be suitable for smaller companies, but bigger companies should consider the following alternatives.

- Pilot: introduction of this product first in one branch- the pilot program. Risk is limited to that specific branch.
- Parallel: running the new system side by side with the older one. This entails low risk but high cost. The higher cost is due to the execution of same tasks twice which leads to more work, overtime costs, or hiring of extra employees.
- Phased: similar to parallel but limited to a single branch of the company which progresses to other branches depending on the success of the new system in that branch.

Maintenance

After the system is implemented, a maintenance of the system is needed to keep the system functioning optimally.

Fishpool and Fishpool (2020, 27) reasons that this is necessary because there can occur problems or errors with the software that customers come across with. I also completely agree here with the Fishpools. Once a software is built, there still can occur some bugs that the testing phase may have overlooked.

As a result, software companies must have a maintenance plan to address issues or feedbacks from their customers.

Fishpool and Fishpool (2020, 27) state that maintenance plans fall under three categories: corrective, perfective and adaptive. The first two categories are classified based on the priority level of the feedbacks from the customers. They are reactive which means that the actions are only taken after getting feedbacks from users. The third however is proactive.

Adaptive is a planned activity in response to new technology or techniques that could enhance the performance of the system greatly (Fishpool & Fishpool 2020, 28).

Review

Fishpool and Fishpool (2020, 28-30) define review as the final phase of the SDLC. All the parties involved in the project meet to discuss if the new software has met its goals or not. They also review about pros and cons of the development process and answer questions like:

- Problems faced?
- Has each phase been allocated enough time?
- How to improve the development process?

- What lessons learned that can be applied for the next project?

SDLC model suits for kinds of projects where the tasks are predictive, rather straightforward, and wherein the tools used to build the software is well known in advance. But in a more complex and unpredictive projects, SDLC proves to be rigid and ineffective. It is also in practice susceptible to common project flaws such as scope creeps, missing deadlines, and conflicts in stakeholders demands. (Fishpool & Fishpool 2020)

2.4 Modern Software Development Methodologies

There are four major methodologies by which a software product could be developed. These are sequential, incremental, iterative and agile (Fishpool & Fishpool 2020).

Precisely speaking, agile is not a methodology. It is an umbrella term and is the basis for many methodologies which share on the concepts and ideas defined by agile. (Schwaber & Beedle 2002).

In my opinion, scrum is one of the most popular of these groups. I said so because I have heard about it so frequently. But I have not heard much about the other agile models namely: DevOps, extreme programming etc. Therefore, I will focus more on Scrum.

Each methodology has its own methods and rules for the software development process.

2.4.1 Sequential

It is practically SDLC with the requirement that each phase be completed before proceeding to the next. The best model which represents this methodology is the waterfall model. (Dooley 2011).

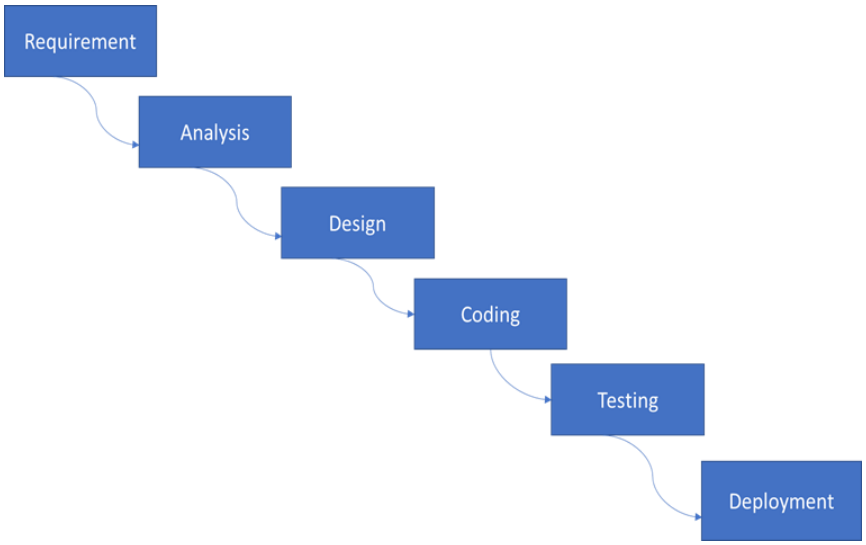

Figure 3. Waterfall Model (adaptive from Dooley 2011).

The waterfall model prohibits going back to the previous phase. It requires a thorough completion of each phase before deciding to go to the next phase. This kind of methodology is also known as linear modelling. (Dooley 2011).

On major drawbacks of this model is its rigidity and inability to accommodate feedbacks. Most of the time, it is difficult to know all the requirements of a new system. Dooley (2011) states that to solve this problem, one can use a variation of the waterfall model which is known as the waterfall with feedback model. The only difference is that, in waterfall with feedback model, it is possible to go back to previous phases as needed. This accommodates the rigidity of the waterfall model. (Dooley 2011).

2.4.2 Incremental

This kind of methodology is convenient for those kinds of projects where the different parts of the product could be grouped into separate modules. (Fishpool & Fishpool 2020, 36).

Then each development team can work independently on the separate modules. For example, let's think of the final product to be a car. Then each member of the team work on separate parts of the car. One could be building the tires; other member works on the engine parts etc.

The car is being built in an incremental manner. Finally, the end products of the team members are assembled to provide the full product, the car.

2.4.3 Iterative

This kind of methodology is useful when all the requirements of the software are not known well in advance. The development team start with their currently known requirements and goes through standard SDLC phases up to and including testing. (Fishpool & Fishpool 2020, 37-38).

Then after customer feedbacks and with the addition of new requirements, the above process repeats. This continues iteratively until a final product is agreed with the customers (Fishpool & Fishpool 2020).

This kind of methodology fosters innovation and involves the customer actively in the development process so that the outcome will be in accordance with customers' expectations (Fishpool & Fishpool 2020).

In my opinion, this method has a lot of similarity with the following models i.e., agile framework models. The only major difference it has with agile models is that iterative model is exposed to scope creeps problems while scrum addresses this issue elegantly.

2.4.4 Agile

Agile is an umbrella term for all methodologies which encompasses both the iterative and incremental ways of developing a software. It is widely used in the software industry. (Heath 2020).

As the name suggests, agile is dynamic, flexible, and adaptive to different situations faced during development. There are many methodologies which follow the agile framework: Scrum, extreme programming, crystal, XP etc. (Heath 2020).

Scrum

During the late 1990's, Ken Schwaber and Jeff Sutherland devised an agile way of developing a software which is known as Scrum. Scrum follows the agile principles and guides. It is the most popular Agile framework. (Heath 2020).

Scrum does not prescribe how to develop out software. It provides my development process clear boundaries. I can adopt any development processes, or various methodologies seen in the previous sections as long as I remain inside the scrum boundaries. (Heath 2020).

The fundamental principle of Scrum is empiricism. It emphasizes on experiences, observations, and personal judgement than rigid methodologies that were discussed in previous sections. The core foundations of Scrum are transparency, inspection, and adaptation. (Schwaber & Sutherland 2016, 3).

Before I discuss the Scrum in action, I want to briefly describe the key scrum terms based on my understandings so that later the use of these terms is clearer.

Product backlogs are dynamic sets of a new product's requirements, features etc. They are usually collected from the feedbacks of customers, development teams, management and any other stakeholders who are going to use the product. (Schwaber & Beedle 2002, 72).

Product Owner is the person who oversees the product backlogs.

Scrum Master is the change agent who makes sure the scrum principles are followed. He or She arranges the daily scrums, help the development team, explains the scrum ideas to both the scrum team and outside parties, removes obstacles, protect the development team from scope creeps etc. (Schwaber & Beedle 2002, 142-146).

Development Team are the team who actually build the product. They consist of 5-9 software professionals across many fields of expertise. (Schwaber & Beedle 2002, 35-38).

Sprint Heath (2020) defines Sprint as a "time-boxed" period of maximum one month long where incremental works are carried out. The official Scrum Guide, its link can be found in the reference section, calls it the heartbeat of Scrum.

Sprint Backlog are the backlogs that are selected by the development team from the product backlogs to be done in the upcoming Sprint (Schwaber & Beedle 2002, 71).

Heath (2020) states that Scrums consists of three major parts: scrum team, scrum events and scrum artifacts.

Scrum Team

Schwaber and Sutherland (2016, 5) define the scrum team to be composed of

- One Product Owner
- One Scrum Master
- And the development team, optimally 7 members.

Schwaber and Beedle (2001,36) suggested that the optimal size of the development team to be limited within the range of 5-9 professionals.

The book reasons that too few team members prohibit the magnitude of communications within the team and leads to poor results. I totally agree. I think teams that consist of less than 5 members can potentially be loaded with tasks which may not be in their field of expertise. For example, the lack of a professional software tester in the team could lead to poor software quality.

Schwaber and Beedle (2001,37) also stated that too many members in the team cause difficulties for the Scrum master to manage the daily scrums and adds complications. I also think here that in a larger team it would be more difficult to follow the core Scrum principles, effective communication, and transparencies.

Scrum events

Heath (2020) points out that Scrum events consist of: sprint planning, daily scrums, sprint review, and sprint retrospective.

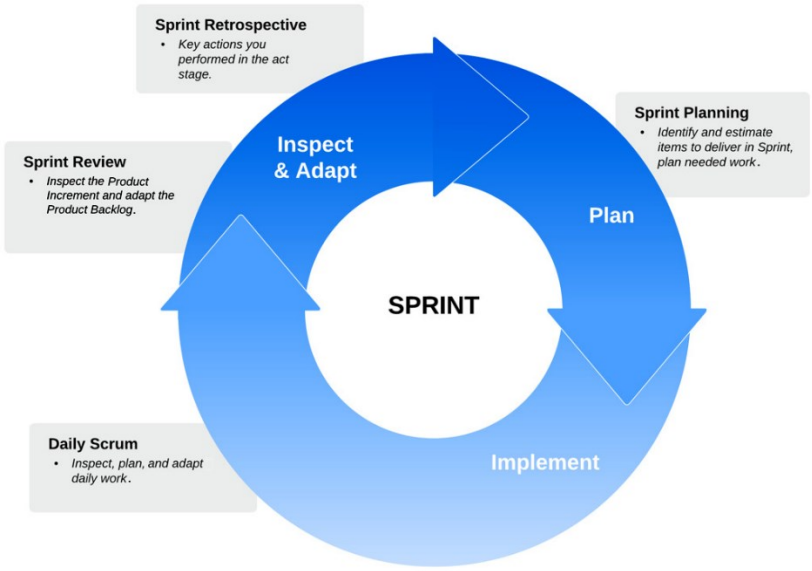

Figure 4. The events in a Sprint (Heath 2020).

Those are basically the activities that are done within a single Sprint. They are done iteratively in each series of Sprints to attain a series of product increments at the end of each Sprint. Figure 4. shows the cyclic and repetitive nature of these events. (Schwaber & Sutherland 2016, 7-13).

The sprint began with a Sprint Planning. In the planning meeting, all the Scrum team members and other stake holders meet to discuss what should be done in this Sprint. A Sprint Goal is then agreed, and the development team decides and chooses the Sprint backlogs. (Schwaber & Sutherland 2016, 9).

Then the Scrum master and development team meet daily in what is known as Daily Scrums to briefly discuss what is done, what to do next and any problems faced. If needed, the developers make necessary adjustments to meet the Sprint time span. Here the Scrum master tracks the work done by using burndown chart (Figure 5). (Schwaber & Sutherland 2016, 11-12).

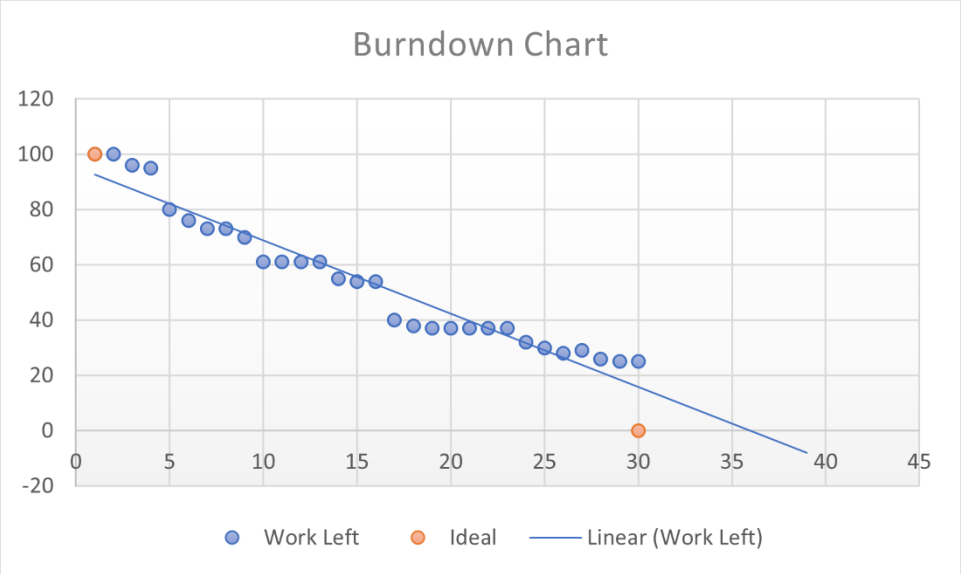

Figure 5. The Burndown Chart (self-made)

At the end of the Scrum period is held a series of two meetings: Sprint Review and Sprint Retrospective (Schwaber & Sutherland 2016, 11-12).

In the Sprint Review, the Scrum Team and all stake holders meet to discuss about the Sprint's product. They determine whether the Sprint output has met its goal and any future changes needed to adopt. This can result in additions of items to the product backlogs. (Schwaber & Sutherland 2016, 7-13)

The Sprint Retrospective meeting is conducted by the Scrum Team to reflect on what went well in the Sprint. What can be taken for the for the next Sprint and what should be avoided. This gives an opportunity for the Scrum team to refine their methodologies as the progress from one Sprint to the next. (Schwaber & Sutherland 2016, 7-13).

Scrum artifacts

Schwaber and Sutherland (2016, 13) define Scrum artifacts as “representing the work or value.” They provide transparency to the Scrum.

They consist of product backlogs, sprint backlogs and product increments (Schwaber & Sutherland 2016, 13-15).

2.5 Tools used to develop the app

2.5.1 Visual Studio Code

Introduction

Dechalert (2021) points out that Visual Studio Code (VS Code) is one of the most popular source code editors for frontend developers. Basically, it is a lightweight code editor which can be run on a desktop and be used across different operating systems like windows, macOS and Linux. It already has built-in support for programming languages: TypeScript, JavaScript and Node.js etc. It also has a rich source of extensions for other languages, libraries, and frameworks (such as Django, React, C#, Java, Python, PHP, Go, .NET). (Johnson 2019).

I spend most of the time developing this app on VS Code as the frontend part covers large parts of the whole project. So, I find it necessary to discuss more in depth about this tool. I believe that familiarizing myself with this tool can increase my productivity, reduce work time and thus leads to higher efficiency.

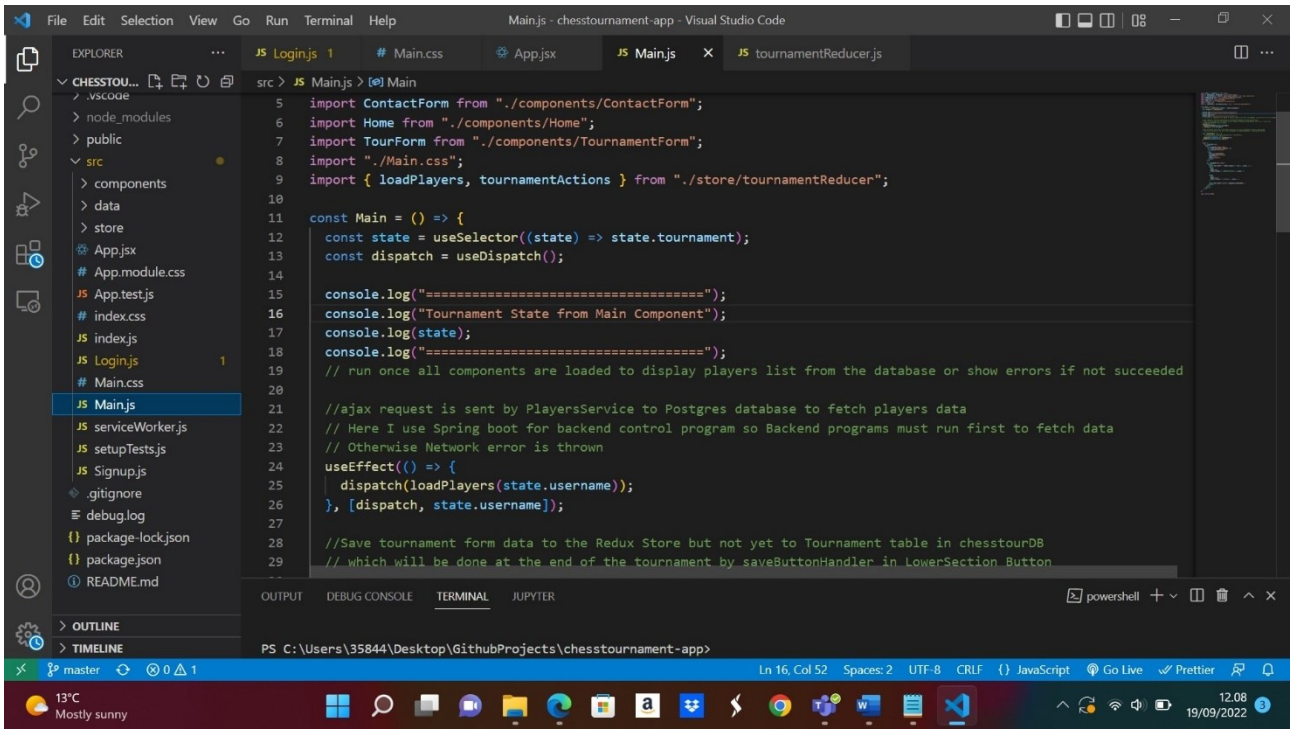

Figure 6. VS Code Editor (self-made).

According to the official Visual Studio Docs, VS Code is not an integrated development environment or IDE. It is a text editor. It does not provide some of the features that a fully IDE such as Visual Studio does. (Visual Studio Code Official Docs).

For my project, I found VS Code to be familiar, easier, and convenient to work with. My justification for choosing VS Code editor also becomes evident as I describe more about its features and uses in boosting developers' productivity.

Basic Setup

Installing VS Code is very easy. Because VS Code is a text editor, its download size is considerably much smaller than that of an IDE.

Depending on the needs of my application, if I need a certain tool for VS Code to complete a task, I can easily get it from the market extensions place within VS Code.

Johnson (2019) describes that VS Code is installed by executing the following three simple steps.

1. first install the installer, which is available through the URL https://go.microsoft.com/fwlink/?LinkID=534107

Then run the downloaded installation

2. VS Code is now installed and ready to go

VS Code Features

There so many support features that makes developing application with VS code so much easier faster and convenient. Here, I picked up some of these features that I found to be so much relevant when I use VS Code. These are basic editing support, keyboard shortcuts, extensions marketplace, multiple selection, and snippet.

Editing support

VS Code primary support is code editing. It enables programmers to produce and code efficiently. Coding with VS Code saves time and improves efficiency.

VS Code utilizes the following editing mechanisms to accomplish this smooth code editing experiences for developers: keyboard shortcuts, multi-cursor, shrink or expand selection, save or autosave, hot exit, find and replace, search and replace, IntelliSense, folding and etc.

Here I want to select from the above list, some of the features which I found to be more important and that I used more often during the coding process.

Keyboard shortcuts

Johnson (2019) defines keyboard chords as a combination of keyboard keys pressed at the same time to perform a pre-defined tasks that are associated with these combinations of keys.

VS Code supports wide variety of keyboard chords for many commonly used commands. Here I list some of the most used keyboard chords from the official Visual Studio Docs (see sources).

- Alt+Shift+A for toggling block comments,
- Ctrl + Space to call IntelliSense,
- Ctrl + comma for settings,
- Ctrl + Ö to open the terminal window,
- Ctrl + Shift + P to open command palette,
- Ctrl+ Shift + X for extensions etc.

And on a case that a new user of VS Code was already using the keyboard setups of other IDEs or software editor, it can be hard to switch back to VS Code system of keyboard chords setup. This is where Keymapping comes handy. Using Keymap from the extension, this new user can migrate his familiar keyboard system to VS Code. Thus, avoiding the need to learn a whole new keyboard shortcut system. (Visual Studio Code Official Docs).

There can also arise a situation where a user does not want to change the entire keyboard shortcut system but rather makes a few changes to the existing keyboard shortcuts systems. In this case, VS Code also allows the customization of its current default keyboard setups. (Visual Studio Code Official Docs).

Multiple selections (Multi Cursor)

During the coding process, there sometimes arise a situation where I need to modify same modifications on many parts of the same file. Rather than the time-consuming operation of editing same thing multiple times across the editor, VS Code provides users to select those parts by pressing Alt key and clicking on the desired areas to create multi cursor and then editing one time affects all. I found this to be quite time saving and handy. (Johnson 2019).

Snippets

In VS Code, snippets are templates that makes it possible to load commonly used programming codes like the for loop, functions, class etc (Johnson 2019).

For example, to write any function in JavaScript I have to use the function keyword, the function name and the opening and closing curly braces.

Rather than typing this boiler plate code every time that I want to write a function, I just type a few letters like func to activate the function snippet and press enter and I have the basic function structure. What is left now is just to provide the function name and its body parts.

Snippets are one of the most useful features of VS Code. There are three ways to get a snippet: VS Code's built-in snippets, custom snippets or user's own snippets, and third-party snippets which can be installed via the extension section. (Visual Studio Code Official Docs).

Extension Marketplace

Besides the features that VS Code provides, it makes it possible to install third party tools which helps the coding process (Johnson 2019).

This makes VS Code an extensible editor, a key feature of a robust editor. For example, one headache area for programmers is formatting. Formatting is an important part of any programmer tasks. But it can consume quite a time to painstakingly edit each new code line. Lack of consistency in formatting patterns can also cause visually unattractive code display. Fortunately, through VS Code extensions marketplace, one can install a popular configurable formatter like Prettier to automate this task and spare the programmer from formatting worries. (Johnson 2019).

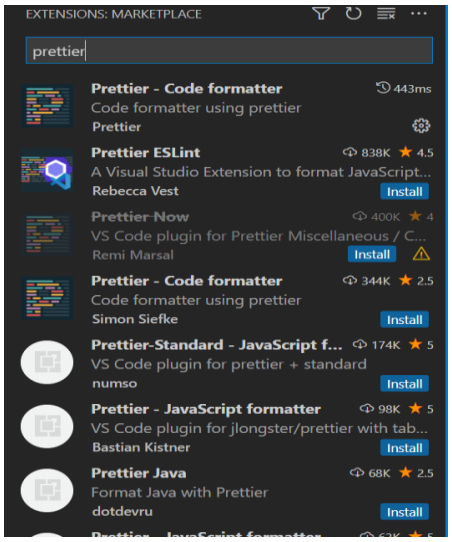

Figure 7. VS Code Extension Marketplace (self-made).

2.5.2 IntelliJ IDEA

It is an Integrated Development Emulation and Analysis (IDEA) software for Java programming language (Java). It is one of the three most popular Java IDEs. The other two are Eclipse and Net-Beans (Hagos 2021).

Figure 8. displays the different parts of IntelliJ UI.

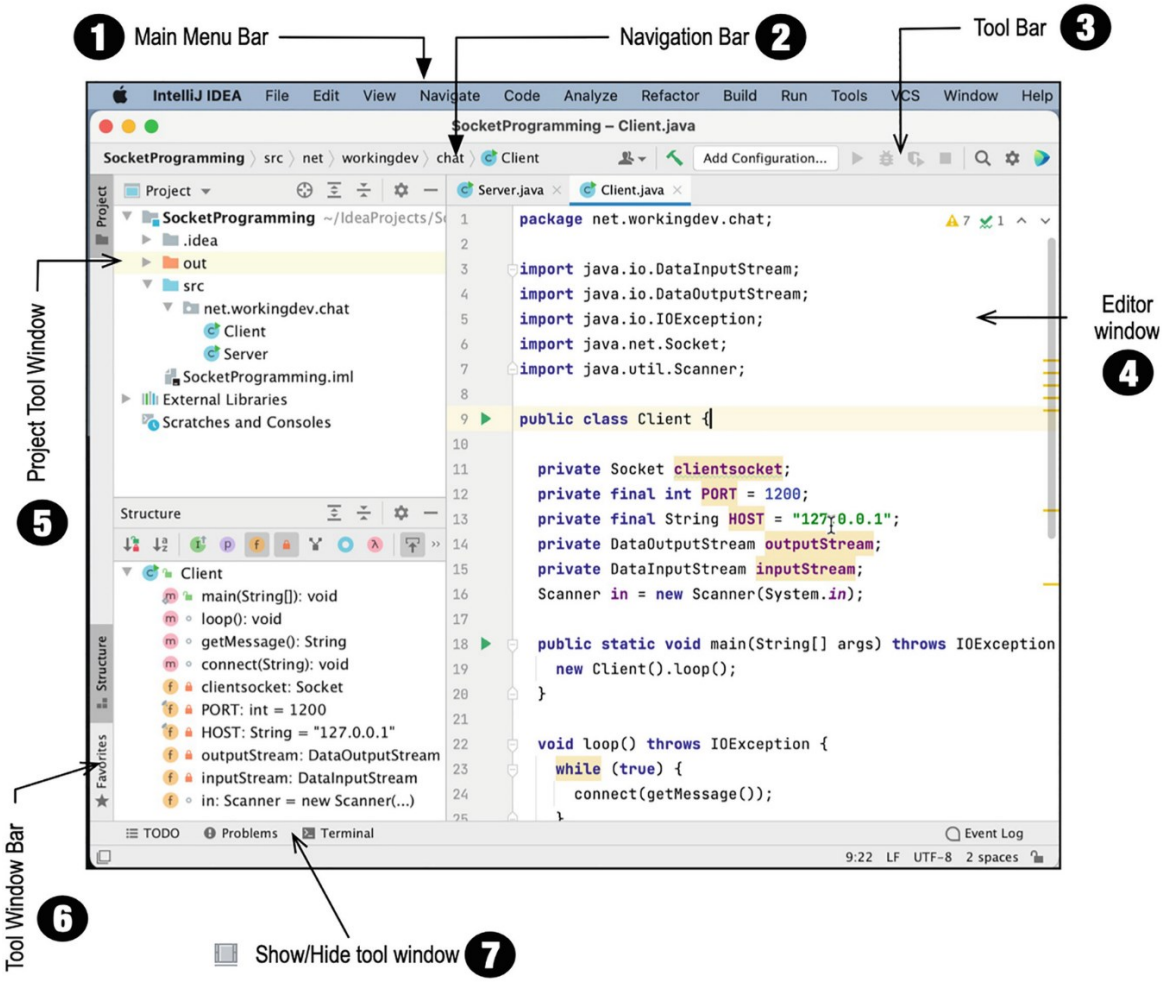

Figure 8. IntelliJ IDEA (Hagos 2021).

Basic Features of IntelliJ IDE

Here I would like to discuss some of the features of IntelliJ that made me to switch from Eclipse IDE it for my backend development.

Community edition

IntelliJ has two options of installing the software. The first one is the ultimate commercial edition and the second is the free community edition (CE). The community edition has limited features when compared to the ultimate commercial edition.

But Hagos (2021) states that the CE does support Maven, Git (source code version management tool) which I plan to use. So, I found CE to be sufficient for my project.

Build Automation.

In my Spring boot app, I use Maven, a dependency management tool for Java. I found IntelliJ IDE's rich support for Maven to be very appealing. I can tweak my preferences in IntelliJ Maven settings. (Korchmalski 2014).

- Work offline. For example, if I want to limit Maven from downloading new updates, I can mark the work offline checkbox.
- Override. This checkbox is used to set different path for Maven downloads than the one used by default.

Code Completion

IntelliJ's Code generation really does exactly what its name infers. It is one of the most important advantages of IntelliJ. It saves time by generating Java's boilerplate codes. (Korchmalski 2014).

In windows the key command 'Alt + insert' generates the following window

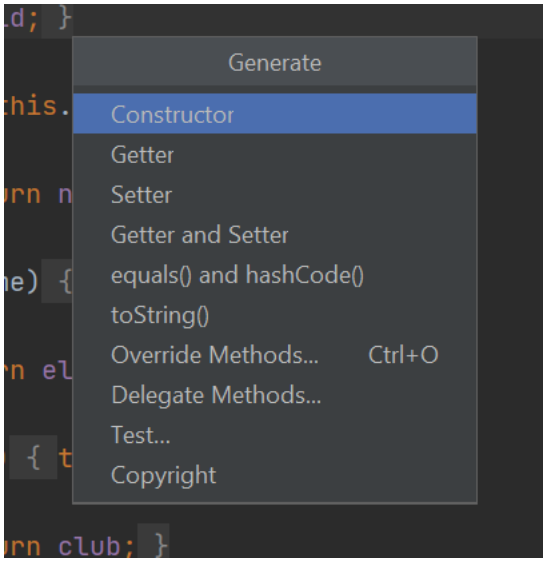

Figure 9. IntelliJ Code Generation Tool (self-made).

As the figure shows I have the options of generating the codes for Java's class constructors, getters' and setters' methods and etc.

2.5.3 React Library

Introduction

I would like to use the definition for React Library given by a member of the founding React team, Pete Hunt. Hunt (2013) defined React as:

> "React is a library for building composable user interfaces. It encourages the creation of reusable UI components which present data that changes over time." (pp. 1)

I found this definition to be precise and best describe what React is all about. React is a lightweight library. It adds small set of features to the plain JavaScript or vanilla JavaScript. Unlike web frameworks, it does not provide all the tools needed for the common activities in web application developments. It is primarily focused on the user interfaces of a JavaScript Application. (Banks & Porcello 2020).

React was invented by a Facebook employee known as Jordan Walke. In 2013, Facebook Inc. made React available as open-source tool. This led to the rise of very useful tools such as React Router, Redux etc. I have adopted the first two in my application. I will also briefly discuss what they are and their uses. (Banks & Porcello 2020)

The release of React as an open-source project coupled with the subsequent rise of great third-party tools increase the widespread use of React by top companies (Banks & Porcello 2020).

Why React?

React is not the only web tool to be used for the frontend. There are many tools that can be used for the frontend like Angular, Vue.js etc. So why should one be choosing React? What makes React stand out?

Patel (2020) stated the following as React's advantages over other alternatives:

- Components: React uses components rather than templates.
  - Learning Curve: Compared to other frontend frameworks, React is easier to learn. There are not many new ideas introduced other than JSX.
  - Reusability: These components can readily be reused.
  - Performance: React's virtual DOM makes rendering faster
  - Strong Community Support

Basic Features of React

React Component

Component is part of the User Interface (UI). It is usually designed to support a specific task. For example, user's login page is one part of the whole UI.

In React the UI is broken down to smaller components. These components are represented by their respective JSX files which is discussed next. (Banks & Porcello 2020).

This way of organizing the UI makes it easier to test, debug and reuse components in an efficient way.

React JSX

JSX means JavaScript and XML. It is a way of using HTML tags inside a JavaScript file. The direct use of HTML in JS files is way easier than using React's createElement function. The latter way of creating React elements is cumbersome and is susceptible to errors. (Banks & Porcello 2020).

Using HTML in JS file is made possible because of Babel. Babel is a JavaScript compiler. When this JSX file is compiled, it is converted by Babel to plain JS with the HTML tags converted as arguments to React's createElement function. (Banks & Porcello 2020).

Data Flow

Data flows one way in React. It is passed from the parent component to its children through props. Props a shorthand for properties are attributes of a React Component. (Banks & Porcello 2020).

For example, to a React Component Login, data can be passed through its username props.

<Login username= {} />

But to pass data from child to parent component, I have to implement action event listener.

ReactDOM

React uses virtual DOM. This virtual DOM is a copy of the actual DOM. Whenever there is a state update in React Components, React first run its virtual DOM to reflect the changes. It then compares it current virtual DOM with its previous one. If it found that there is a change between the two, then it will render only this change to the actual DOM. It does not render the whole real DOM. This would have caused a real performance issue. (Banks & Porcello 2020).

This is one of the benefits of using React. Because of this characteristic, React is able to render the DOM faster than would a vanilla JS apps. (Banks & Porcello 2020).

How to create a React project using VS Code?

Before proceeding to create a React App in VS Code, I must make sure that Node.js is installed in my system.

Node.js is a runtime environment for executing JS codes outside a browser. It was first devised by Ryan Dahl in 2009.

The following year the node package manager (npm) was incorporated in Node.js installation files. This makes the uploading and receiving of Node.js packages simpler. It also provides a command line interface (CLI) tool to interact with npm registry.

According to React official Docs (see sources), I can run the following commands in CLI to create my first React project called react-project in VS Code.

1. npx create.react.app react-app
2. cd react-app
3. code .

Then React will create the following structure of folders shown in Figure 10 in my VS Code IDE.

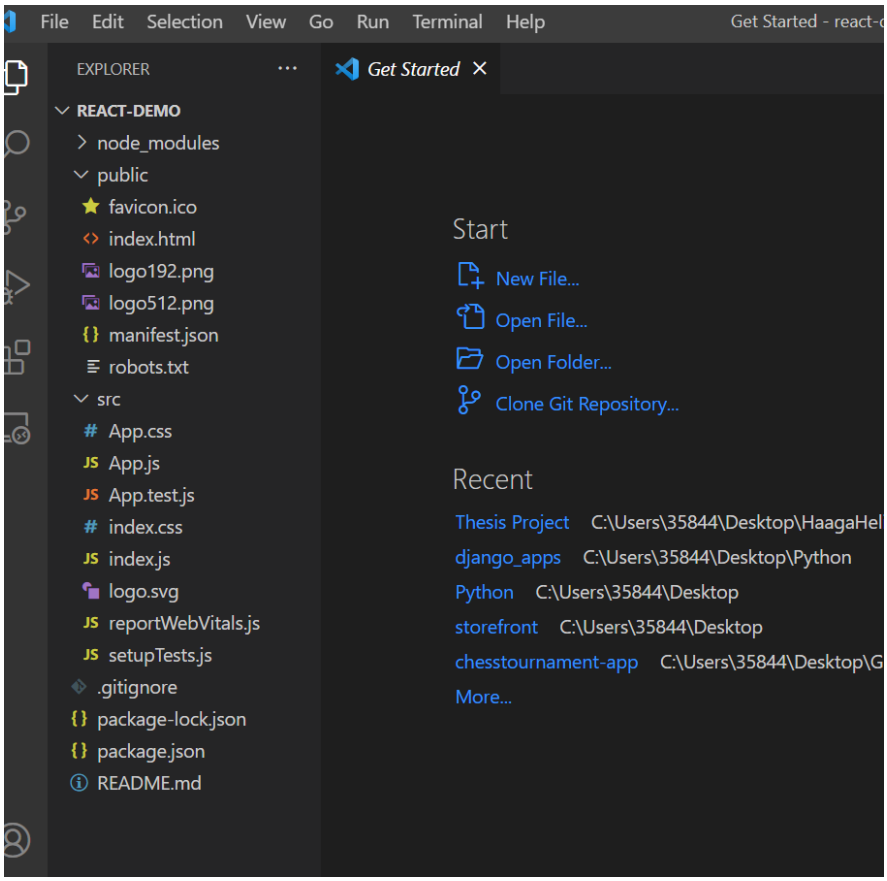

Figure 10. A snapshot of VS Code React Project Structure (self-made).

2.5.4 Redux Toolkit

Introduction

Redux was first developed in 2015 by Dan Abramov and Andrew Clark (Abramov 2015).

Redux is another open-source sate management library for JavaScript applications. It makes a shared state readily available to any components that subscribe to the Redux store. (Garreau & Faurot 2018).

Figure 11 best illustrates the working model behind the Redux library.

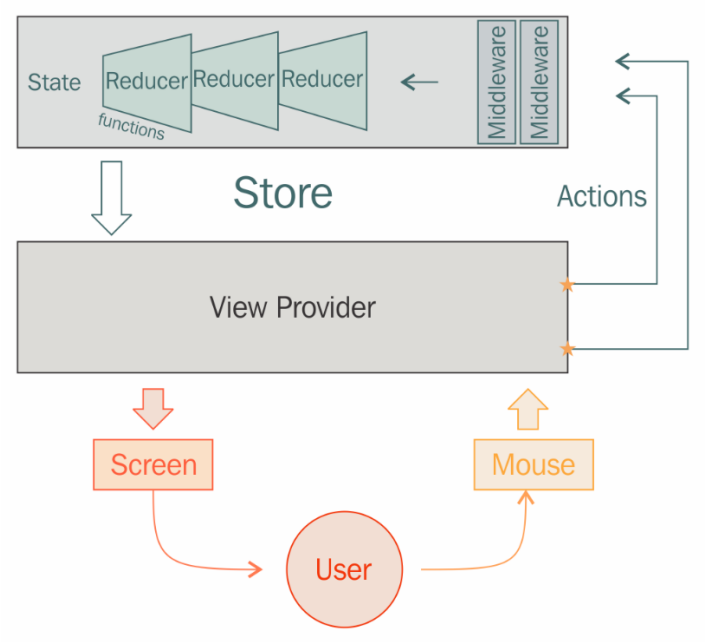

Figure 11. Redux Cycle (Bugl 2017).

Redux Store

It is the centralized store where the contents of the state of the application is stored. It is an object created by using the Redux function ‘createstore’, which takes in as arguments the reducer and optionally the initial state of the app.

Actions

Actions are data objects that are generated as a result of the interaction of a component and the user. In Redux the state cannot be changed directly by the component.

First the component dispatches or passes the generated actions to the reducer.

Reducer

The reducer is a pure function which takes in two arguments as input: the current state and the actions dispatched by a component. It then identifies the operations to be performed via the payload “type” of the action object. It then performs the required modifications on the current state immutability, without directly modifying the current state.

Then it returns the updated state.

Why Redux?

Garreau and Faurot (2018) points out asynchronicity and mutations as the two primary reasons why one should switch to Redux to manage the app's state. These two phenomena make state management very difficult.

Asynchronicity means a user has clicked a button which makes a request to the server. But the precise time of response is unknown. This process continues in the background while the user proceeds to do other actions. Mutation is a change in the value of a state. (Garreau & Faurot 2018).

If these two phenomena are not handled with care, our application could end up in an inconsistent state. The data in the frontend may not be the same as the one in the backend leading to errors. This is where Redux is needed to make a consistent state management. (Garreau & Faurot 2018).

Redux also helps to avoid props drilling. If my app component tree is longer, then passing data between two components on the extreme end of my React component tree requires a series of props calling. This phenomenon in React is called props drilling. (Narayn 2022).

Figure 12 shows a heavily nested React component tree. The root component is called App. The rest are child components nested inside each other. Now without Redux, a data flow between the root component App and the leaf component L is quite cumbersome.

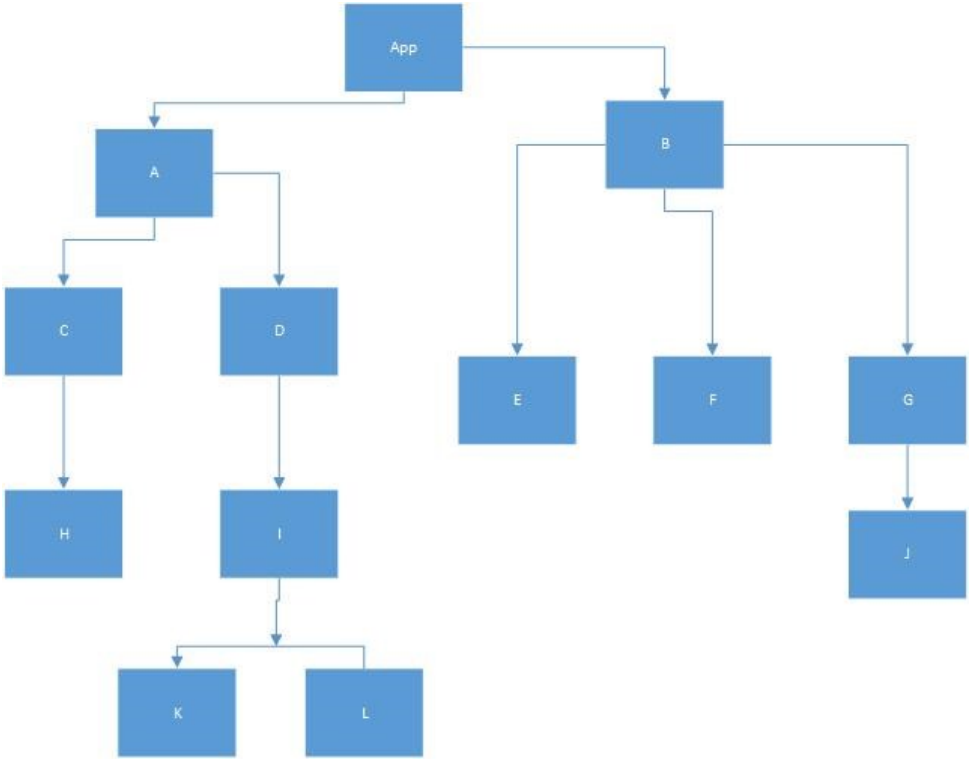

Figure 12. Deeply Nested React Component tree (self-made).

2.5.5 Spring Boot Framework

The Spring framework has made development of web applications easier than plain Java web development. But there are many drawbacks of Spring Framework. Before one gets to start coding the business logic part, users of Spring Framework would have to do tons of work such as setting up the application xml configuration files, install web server such as Tomcat, deploy and run the application using the installed web server, make sure that versions of third-party dependencies are compatible etc. (Walls 2015, 540).

This made using Spring Framework quite cumbersome. But with the introduction of the Spring boot, users no longer have to endure the above stated unpleasantries. Instead, Spring boot takes care of these tasks on the user's behalf by using Spring team's highly opinionated practises. This enables users to rapidly get to the business logic part without getting bogged down in the configuration process. (Walls 2015, 540).

Walls (2015, 541) states that there are four main features that Spring Boot uses to accomplish this task:

- Spring boot starters - groups many similar dependencies into one single dependency.
- Autoconfiguration – automatically sets up the xml configuration files using Spring teams highly opinionated practices and the convention over configuration design paradigm. This is flexible and can be overridden by the user.
- Command line interface (CLI) – uses Groovy programming language in combination with autoconfiguration to simplify the development process.
- Actuators – provides a monitoring service to the Spring Boot application.

Since I am not familiar with Groovy programming language, I am not going to use Spring boot's CLI feature.

Spring boot starters

The spring boot starters provide the required dependencies needed for the build according to the type of project chosen. For example, a Web project has certain dependencies that may be included simply by adding the spring-boot-starter-web starter. (Walls 2015, 541-545).

Starters organize the dependencies under a named banner which makes the dependency specification small and neat. For example, rather than specifying all the dependent jars for, say, Hibernate Validator and Tomcat's embedded expression language, I can simply add the spring-boot-

starter-validation starter. That's all. All the required dependencies are automatically imported under that banner. (Walls 2015, 541-545).

Walls (2015, 543-545) list some of the Spring Boot starters:

- Spring-boot-starter-data-rest: Exposing Spring Data repositories over REST via Spring Data REST.
- Spring-boot-starter-hateoas: Eases the creation of RESTful APIs that follow the HATEOAS principle when working with Spring / Spring MVC.
- Spring-boot-starter-jersey: Framework for developing RESTful Web Services in Java that provides support for JAX-RS APIs.
- Spring-boot-starter-web: Build web, including RESTful, applications using Spring MVC. Uses Apache Tomcat as the default embedded container.
- Vaadin-spring-boot-starter: A web framework that allows users to write UI in pure Java without getting bogged down in JS, HTML, and CSS.

2.5.6 PostgreSQL

Introduction

PostgreSQL, also known as Postgres, is an enterprise level object relational database management system (RDBMS). It is a successor to the INGRES database system which was developed in the University of California. Its name reflects its relationship with INGRES (Post Ingres). (Ferrari & Pirozzi 2022).

Postgres is an open-source database. Its development started in 1987. It underwent many improvements and versions over the years leading up to 1994. In 1994 Postgres was released under the name Postgres95. In 1996, according to the official PostgreSQL document, Andrew Yu and Jolly Chen added the SQL capabilities to Postgres95. (Le & Diaz 2021).

The following year, Postgres95 was renamed to PostgreSQL to emphasize the part that the query language adopted for searching is SQL (structured query language). (Le & Diaz 2021).

Postgres has been in existence for over 30 years now. It remains to be one of the most popular and advanced RDBMS. (Le & Diaz 2021).

Installing PostgreSQL

To install PostgreSQL, I browse to its official website: https://www.postgresql.org/. I navigate to the Packages and Installer section and choose the right operating system. Then I run the installer.

Here I should be careful to make sure that pgadmin4 checkbox is checked. Pgadmin4 is a GUI based visual application. (Ferrari & Pirozzi 2022).

Figure 12 shows the pgadmin4 that I use for this project.

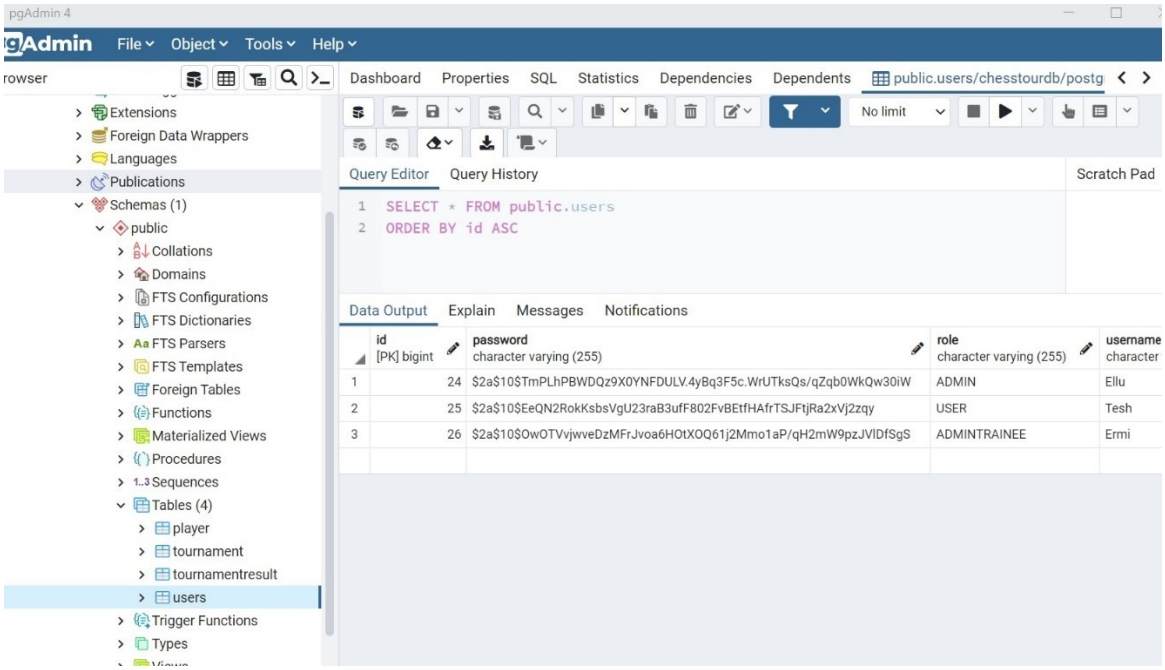

Figure 13. Pgadmin4 graphical tool for PostgreSQL database (self-made).

As can be seen in Figure 13 the Pgadmin4 graphical tool helps to design my database tables, to insert, edit or delete data into my tables and to write SQL commands to PostgreSQL database. Ferrari and Pirozzi (2022) list the three basic features of PostgreSQL:

- fulfils the properties of database transactions namely ACID (atomicity consistency isolation and durability).
- extensibility with other programming languages such as Java, Python etc. I make use of this property to connect PostgreSQL database with my Spring boot framework.
- runs on all operating systems.

3 Empirical Part

3.1 Initial Stage

The starting point of this application are the requirements set in section 1.2 and the previous web application I was working on the same topic.

Basically, the previous web application is devoid of all the requirements listed in section 1.2. Other than that, it has almost complete user interfaces built in React, a backend and database.

To make matters very clear, I travelled back in time, using git checkout, to the starting period of this thesis work and run the application at that point to show what my previous website's user interface looked like then. The following figure showed the result.

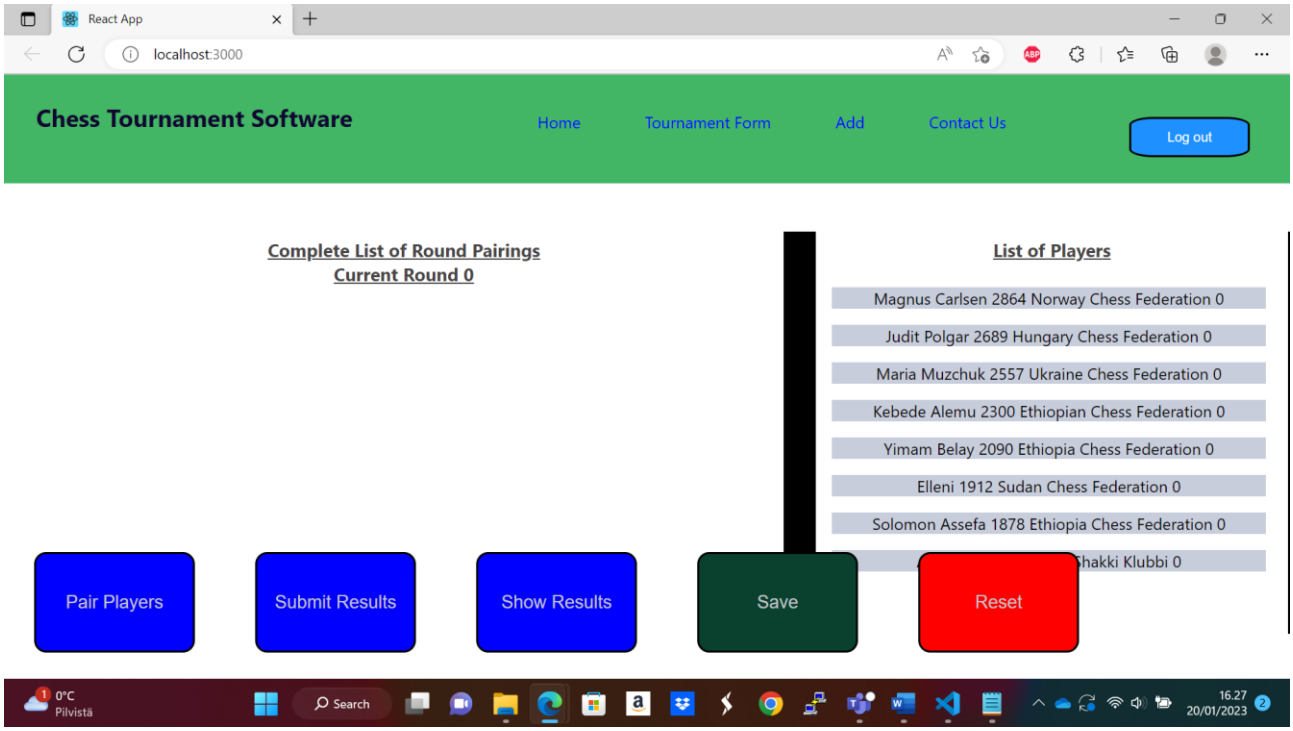

Figure 14. Previous Work (self-made)

Although the difference between the user interfaces of the previous and final web apps, Figure 14 and Figure 25, may seem visually small, implementing the requirements of section 1.2 has induced a lot of changes in source codes of the frontend and backend applications.

This project is not commissioned by a company. It will mostly be done by using either free or open-source applications. Any pending costs that must be incurred to execute the project will be covered solely by the author of this thesis project.

The application plans to automate the manual registration of a chess tournament events. There will not be fees to use this application. I believe that this helps to attract potential users to my application. As the popularity of this application increases, I plan to cover the costs associated with the hosting and operation of the app on the internet by running advertisements.

There are some limitations in developing this application. First factor is time. As already stated in the project scope section, I am the only one involved in developing this app. Thus, I may not have enough time to provide all the features promised in section 1.2.

The other factor is financial matter. As this is a non-commissioned project, any necessary costs are covered by me. This means that running this app in the long run depends on the success of my app in generating revenue to at least cover running costs.

The successfulness of this project is determined if it meets most of the bare minimum criteria set in section 1.2.

3.2 Development

First, I want to show how the web application is going to perform its tasks by describing the general interaction among the three units: frontend, backend and the database. Then I discuss separately and in detail the development stages of each unit.

Figure 15. shows how a user interacts with the web applications. Obviously, the database and the backend will be hidden from the user. User interacts solely using the React user interface (UI) which enables the user to add a new player, enter tournament information, retrieve players data and the option of saving tournament data at the end of the tournament.

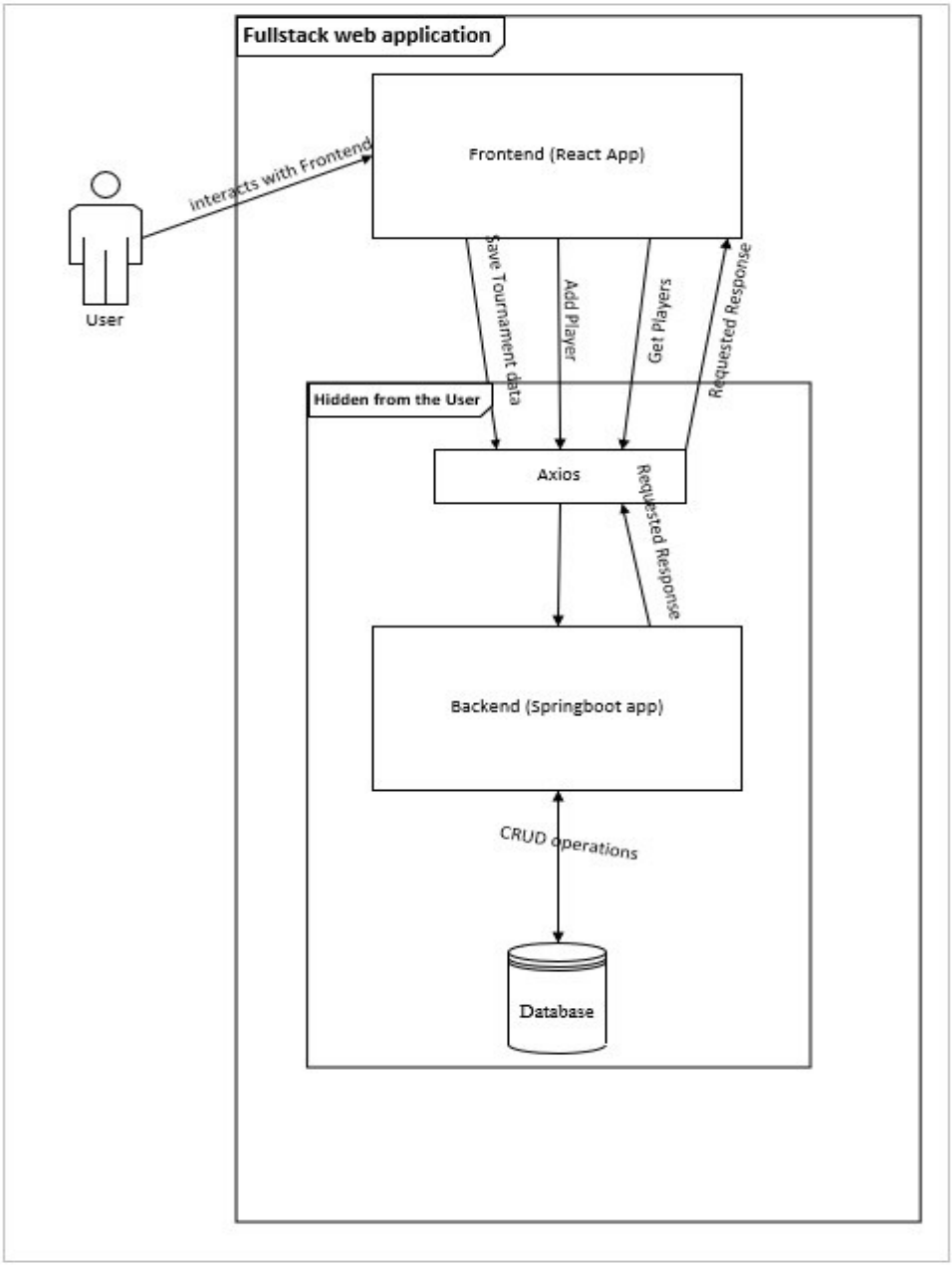

Figure 15. UML Communication Diagram (self-made)

These user interactions are transmitted to the backend using a third person tool known as Axios. It is a JavaScript library that the web app uses to make http requests. (Kollegger 2018).

The backend handles the request by taking the necessary actions to execute the requested items. This is done by interacting with the database. For example, to add a new player, the Springboot app sends a request to the database to insert the new player's data in the player table etc.

Now I discuss about the development processes of each unit.

3.2.1 Frontend

In order to build the frontend section, I used a combination of software models stated in section 2.4.2,2.4.3 and 2.4.4 namely: incremental, iterative and agile models. I did not have the whole complete requirements when I start the development process. It grew as I see it needed during each sprint.

I picked part of requirements from section 1.2 and proceed to complete all the development phases (right before the deployment phase) shown in Figure 2 of section 2.3. This includes testing which is done in all of the sprints. This model of development also applies to the backend too. At the end of each sprint, I have an app ready to be deployed.

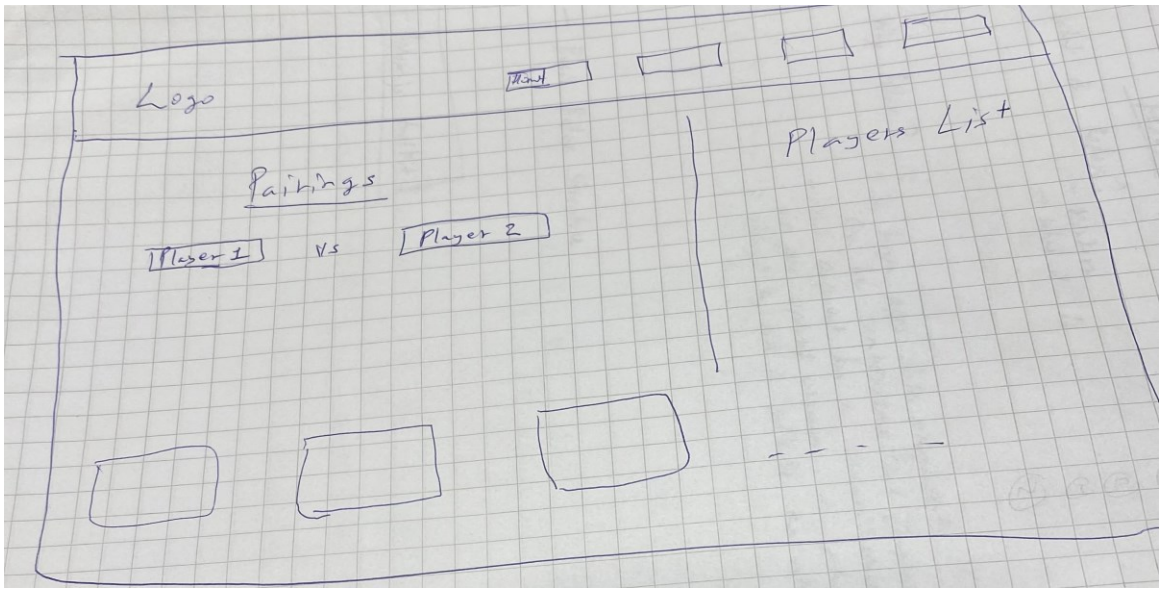

Figure 16. Graphical UI Design (self-made)

In the design phase, I came up with the sketch shown above in Figure 16. The design has three parts namely: the top row, middle row which contains two displaying columns (the right is players listing section and the left is players pairing section) and the bottom row which contains the five control buttons. These are the main control units.

In the top row there are two items inside. The logo is on the left side and next to it lies the five links which lead to separate page where a user can add players, tournament info etc.

In the coding phase, I implemented the functionalities listed in the design phase using the React library. I use the React library as I find it is easier to manage the user interface and the application's large state than plain JavaScript.

Figure 17. shows the large state object that is managed by the Redux Toolkit.

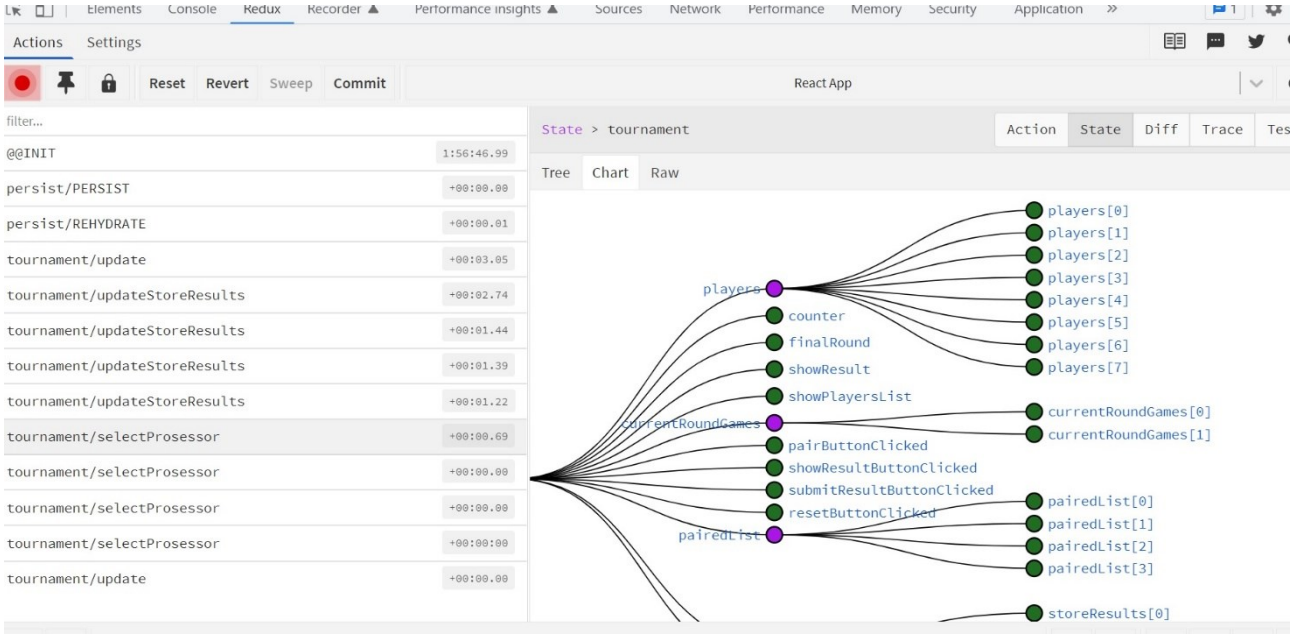

Figure 17. The state object of the web application taken using Redux DevTools. (self-made)

Users interacts with the UI (user interface) primarily using the five control buttons: pair players, submit results, show results, save, and reset buttons. These are the soul of the application. Thus, I find it necessary to include the tasks associated with each one. I begin with the reset button.

Reset button

This button click clears both the displaying columns in the middle row. It also resets the scores of each player to zero, resets the opponents lists of each player and the entered tournament data. In short it resets the state of the app back when it was originally before any use of the app.

User must be careful in accidental clicking of this button because it is always in active mode. This is done to enable the user to revert to the initial state from any point in the tournament stage.

Pair Players Button

This button unlike the reset button have enabled and disabled status depending on the state of the app. The button's click generates the display of the players' pairings list in the pairing list section. It

activates the submit button but deactivates the show button and itself. Figure 17. demonstrates this moment.

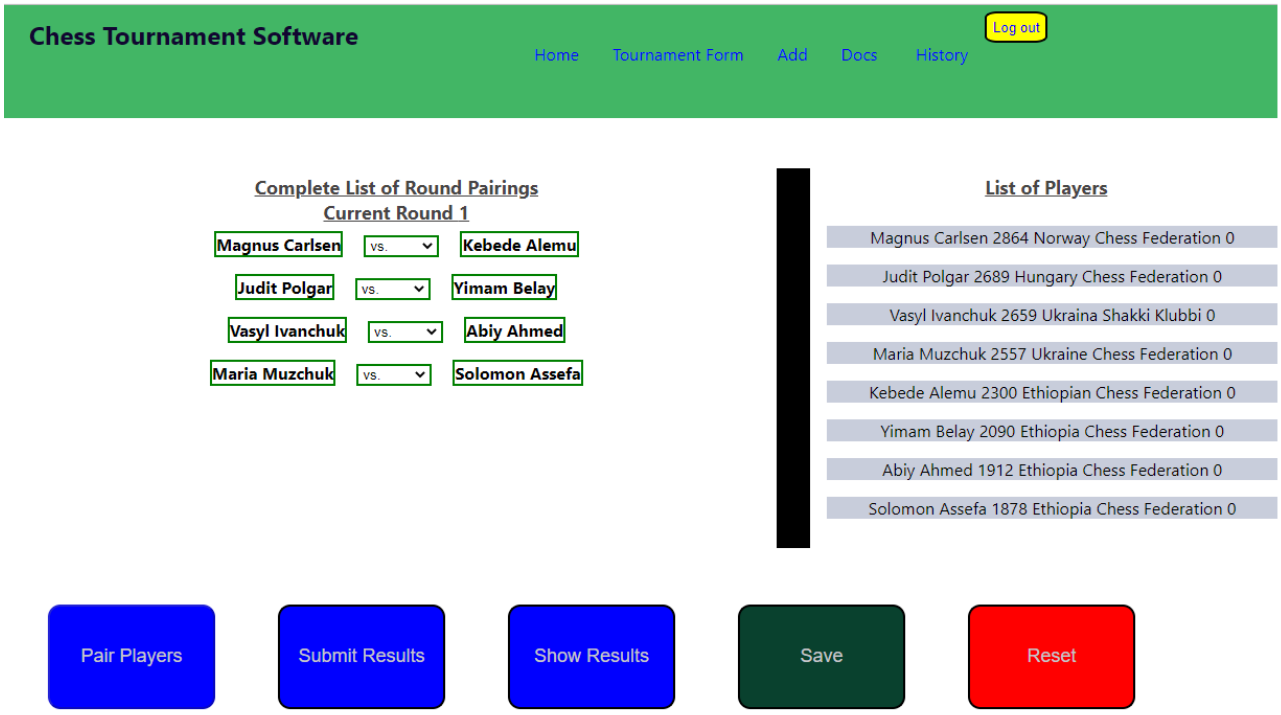

Figure 18. UI after pair button is clicked (self-made)

Submit Results button

This button click triggers the storing of each player's score, generates the paired list and increments the round number. But this is done only if the user has selected the result from the dropdown list for each pairing. Otherwise, it informs the user to enter all of the pairings' result in the round pairings section.

Show Results button

It basically displays the current scores of each player in descending order in the round pairings list section. It does not modify the app's state.

Save button

If the user has filled the form in the tournament page and the tournament has been run the minimum number of times which is entered in the tournament form, then the save button click triggers the app to send request to the backend to save the tournament data. After that the frontend UI returns to the starting position just like as if the reset button has just been clicked. The app is ready once again to run another tournament.

Figure 19 displays all my application's dependencies. Dependencies are the third person tools that I use to build my app.

Figure 19. List of the App's dependencies (self-made)

3.2.2 Backend

I use for the development of this backend server-side Spring boot application, the IntelliJ IDE (integrated development environment). I have discussed in more details in section 2.5.5 the reasons behind my choosing this IDE.

In order to handle my dependencies elegantly, I use the Apache Maven dependency management tool. Varanasi (2019) states that Apache Maven is an opensource project management framework that facilitates the rapid development of a Spring boot application, provides standard directory structure, declarative dependency management and many other advantages. If I had not used maven, then I would have to get all my dependencies one by one manually which is time consuming. (Varanasi 2019).

The official spring initializer website, https://start.spring.io/, makes the Spring boot development process even easier. It helps to quickly set up a new maven project. All I need to do is select the dependencies I need. Then Spring initializer provides me with a new maven project where I can directly proceed with the business logic that is handling the endpoints, setting up security filters etc. (Long & Bastani 2017).

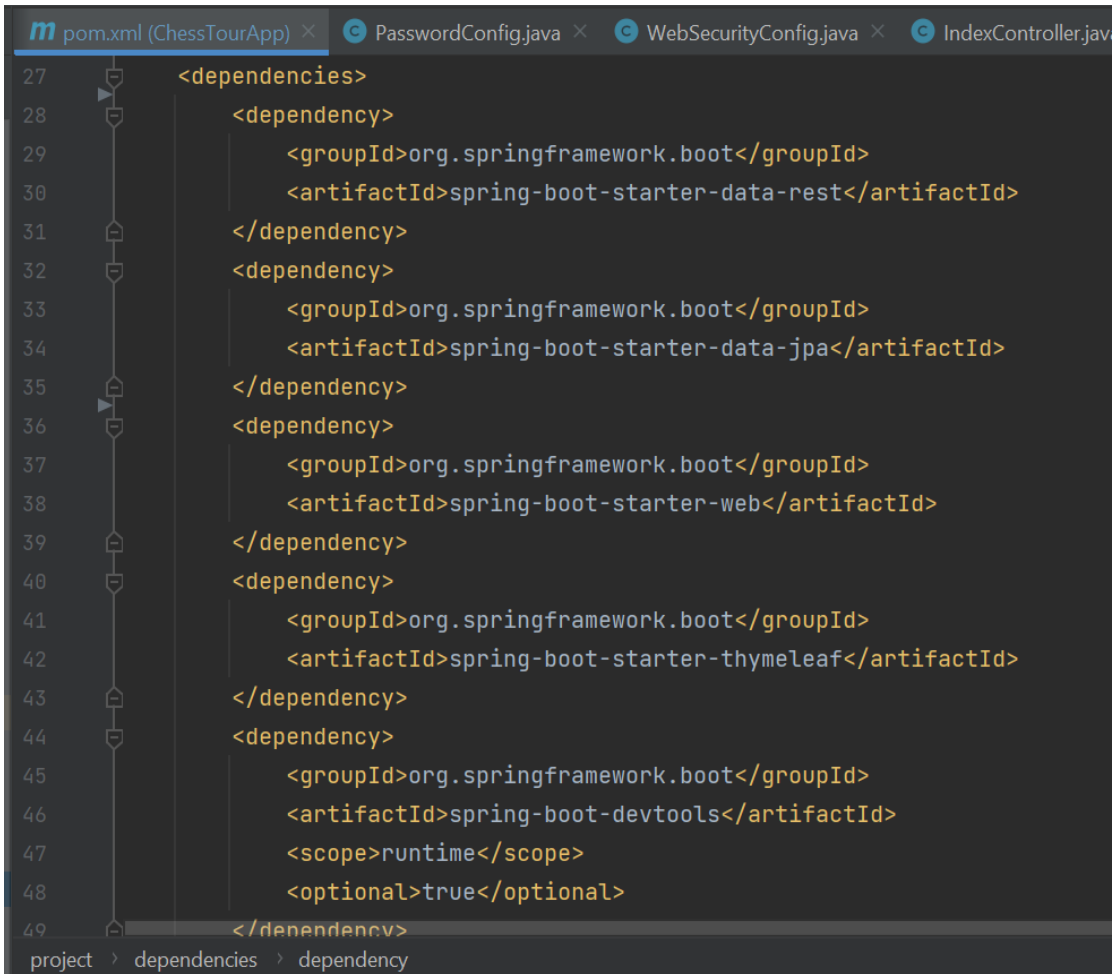

Figure 20. Backend app's POM (project object model) file (self-made)

Figure 20. lists some of the starter dependencies of the spring boot app. Section 2.5.5 describes more what starter dependencies are. This list is automatically generated by the spring initializer during the setting up of this application.

To handle endpoints, I added controller java files to handle requests from the frontend. These are: the index, player, tournament and user controllers. Endpoints are web addresses that a user can browse to gain access to a certain file. (Venitta 2017).

The index controller handles the login and logout mechanisms while the player controller handles the addition of new players, retrieval, updating or deletion of existing players. The tournament controller in turn monitors the addition of new tournaments, retrieval, updating or deletion of existing tournaments and checking whether a tournament with a given id exists or not whereas the user controller is used to authenticate a user during login and saving a new user in the database.

I also configured the web security configuration file so that it only authorizes requests from a specific endpoint which begins with "player/", "tour/", or "user/." I use the BCrypt encryptor for users'

passwords. Jyeett (2021) states that BCrypt is a cryptographic algorithm which converts user's password into a hash. Jyeett (2021) describes hash as a digital fingerprint. (Jyeett 2021).

That means in the database users' passwords are stored by using this encrypted password. This adds security to user's login credentials. Figure 20 shows how users' login data is stored in the database's user table.

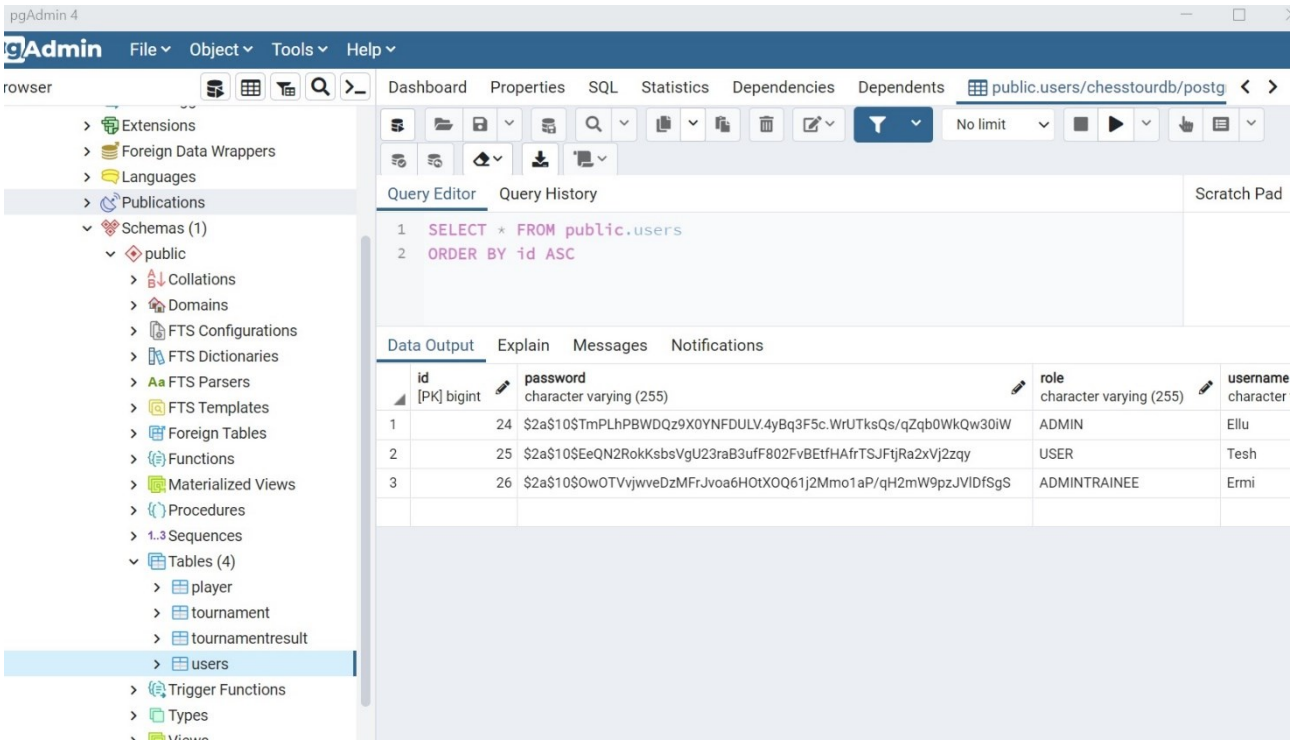

Figure 21. Users table with encrypted passwords.

3.2.3 Database

I chose the PostgreSQL database management system (RDMS) for the following reasons.

Firstly, it is an opensource tool. There are no fees to use it. Secondly, I am familiar with the integration of this tool with a Spring boot application. Therefore, I find it easier to work with than other systems like MySQL, MongoDB etc.

In order to meet the storage demands of my application, I devised an entity relationship diagram (ERD) diagram which is shown in Figure 22.

The Player table stores the players data whereas the Tournament table stores the tournament data which the user enters on a form in the tournament page.

Then I created the TournamentResult table by joining the two tables. This table facilitates the fetching of result data from a specific tournament. All I need to do is set a query on TournamentResult with a given TourId. This gives me a list of all the players who participated in that tournament.

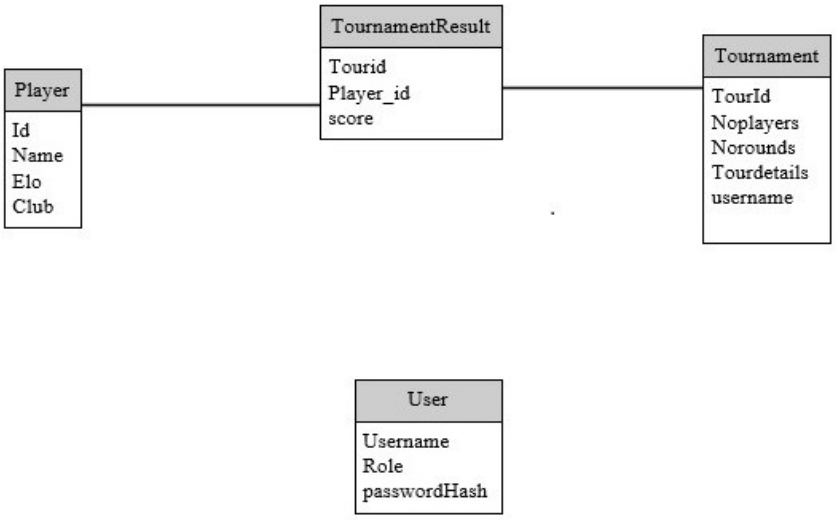

Figure 22. Database entity relationship diagram (ERD)

The User table stores the username, user's role, and password. This is used to validate the user's credentials whenever the user logs in to the system. The role determines what kind of activities the user can do with his or her data. It determines whether the user is administrator, guest etc. By default, all new users are set to have a guest role.

3.2.4 Deployment

Introduction

For the deployment of the application, I choose Amazon Web Service (AWS) because AWS has a Free Tier option where most of the services are free of charge depending on size and period. The Free tier program is valid for the first 12 months since signing up. I also like AWS's pay-per-use pricing model. Wittig and Wittig (2018) state that AWS bill is like an electric bill. You get charged for the memory used, server running time etc. I find these options to be convenient.

There are so many AWS key services which serve the needs of different parts of the deployment process. For example, EC2, S3, IAM, RDS, ELB etc. Sometimes two or more of these services can be used to do same task. For example, EC2 instances and Elastic Beanstalk can be used for running server applications. Each has their merits and demerits depending on specific cases (Wittig & Wittig 2018).

Operation

I choose AWS S3 for static web hosting (React), AWS route 53 for hosting new custom domain name, AWS EC2 instance for the backend and AWS RDS for the database hosting.

Figure 23 shows the whole deployment process. I follow this exact chronological order to deploy the web application.

First, I migrate the PostgreSQL database to AWS RDS (Relational Database Service) service. Then I link up the EC2 instance with the RDS. Here I had the option of choosing Elastic Beanstalk rather than EC2. Soni and Soni (2021) point out that EC2 is just the virtual machine (VM) while Elastic Beanstalk is a pre-configured virtual machine which simplifies the deployment of web application. But I want to configure the VM myself. That is why I opted for EC2. (Soni & Soni 2021).

Then, I install the correct Java JDK (Java Development Kit) version that matches that of the Spring boot on the EC2 instance. Next, I upload the Spring boot jar file on a new AWS S3 bucket other than the one I used for the React App. Then I download this jar file on the EC2 instance by pulling it from this S3 bucket.

Now I have both the correct Java JDK version and the Spring boot jar file on the EC2 instance. I run the jar file using the command "java -jar (jar's file name)." This results in an active server and database in the AWS Cloud environment.

I use the EC2 instance URL for the frontend's http requests. This replaces the localhost URLs I was using during the development stage. The AWS S3 static website bucket host my React app. This generates AWS S3 static web URL.

Then I route the internet traffic from my new custom domain: "chesstourapp.com to this S3 URL. And this concludes the deployment process.

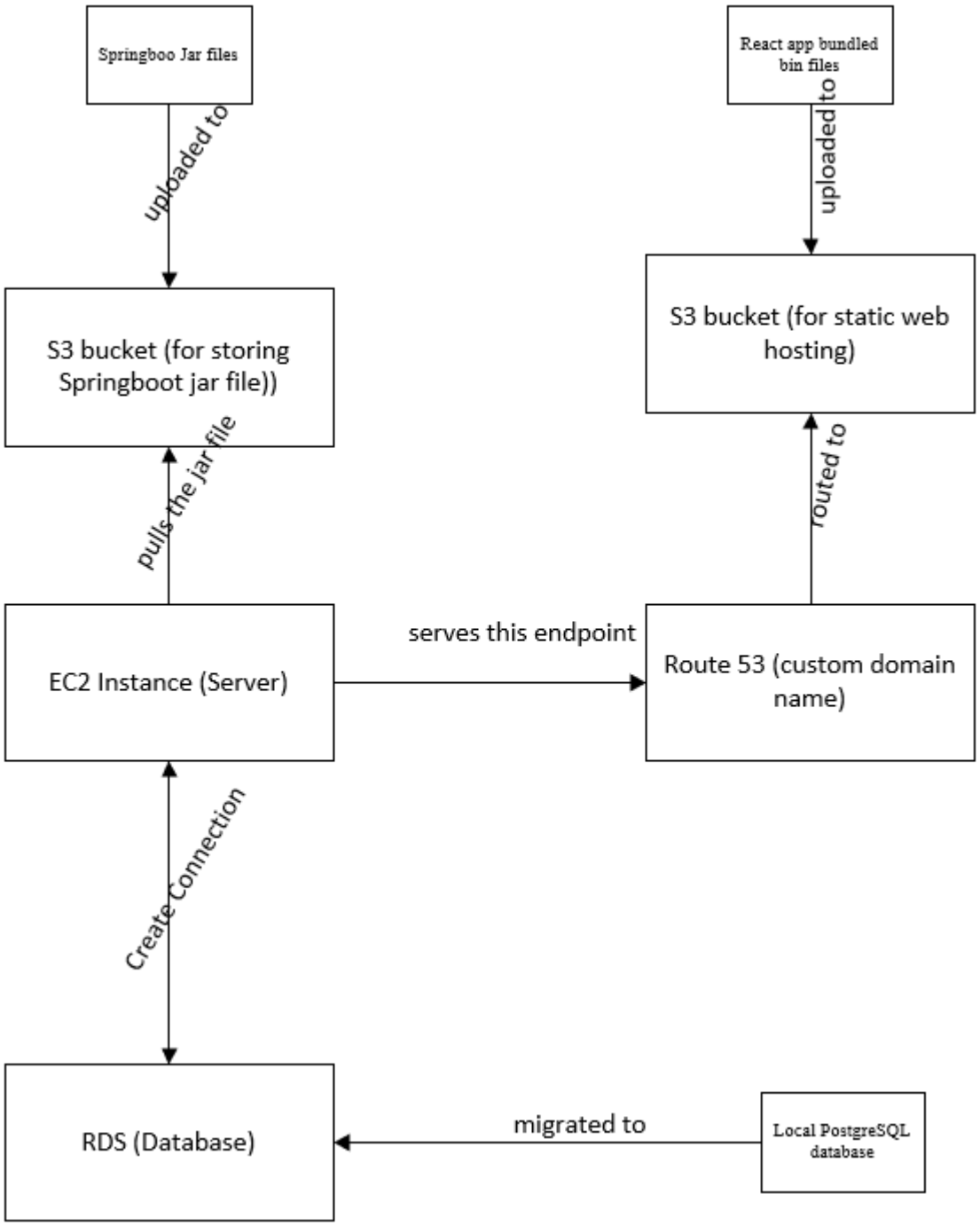

Figure 23. Deployment Process Diagram

The web application can now be found on the following URL: www.chesstourapp.com

3.3 Final Result

This final web project is deployed at the following URL: http://chesstourapp.com/ . Users are first presented with a page requesting for login credential or a sign-up option if the user is new to the website. Figure 24. shows this login page. For demonstration purposes, a new user can login and checkout using the credential given in Appendix 2.

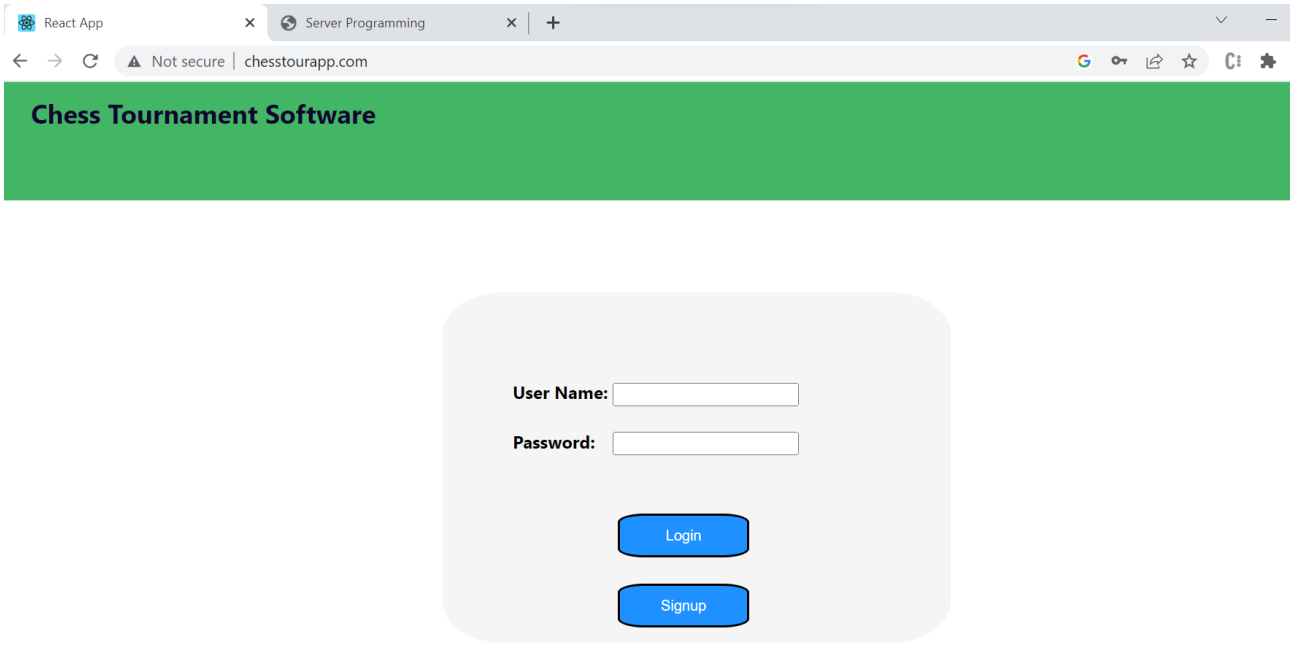

Figure 24. Login Page (self-made)

When the user has successfully logged in, he or she will be directed to the main page where most of the functionalities of the app can be found. The page, Figure 25, also contains useful links which enable the user to add new players, select players for upcoming tournament, fill in tournament info, get complete information on how to use this web app etc.

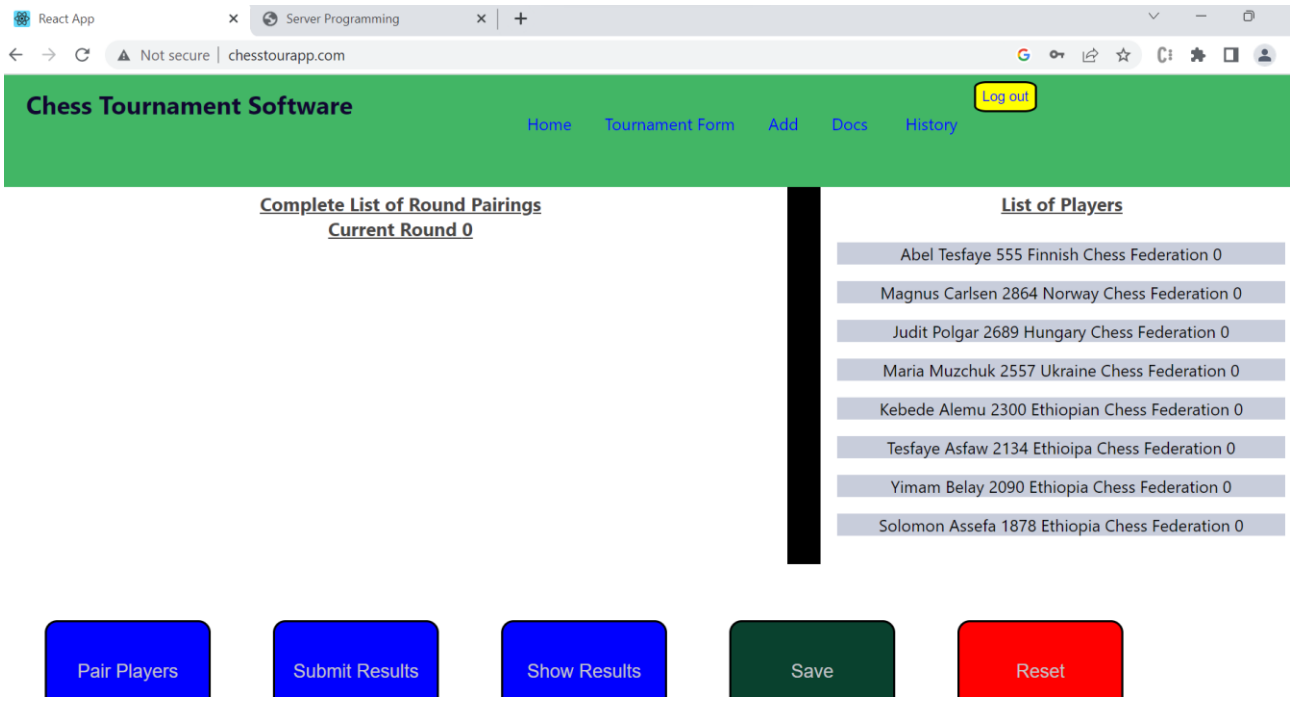

Figure 25. The main page (self-made)

One important link which is found in the green main bar is the “History” link. This can be thought of as a back window for the user to see what has been saved in the database. And as such it has nothing to do with the operation of the current tournament.

That is why when the user clicks this link, unlike the rest, it opens a new page, Figure 26, where the user is prompted to enter once again his credentials. I made this requirement so that only data saved by the current user, not other users’, is fetched from the database.

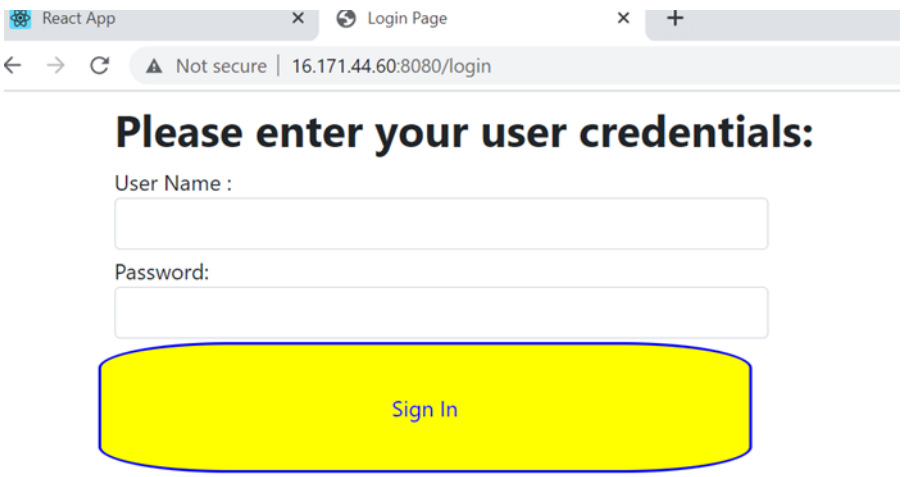

Figure 26. Backend data presenting login webpage (self-made)

After a successful login, Figure 27, the user is able to see past tournaments' information, existing players data etc. The user can also edit or print players, Figure 28, and tournaments' information. But deleting existing players and tournaments is not possible as users are not assigned administrator access rights.

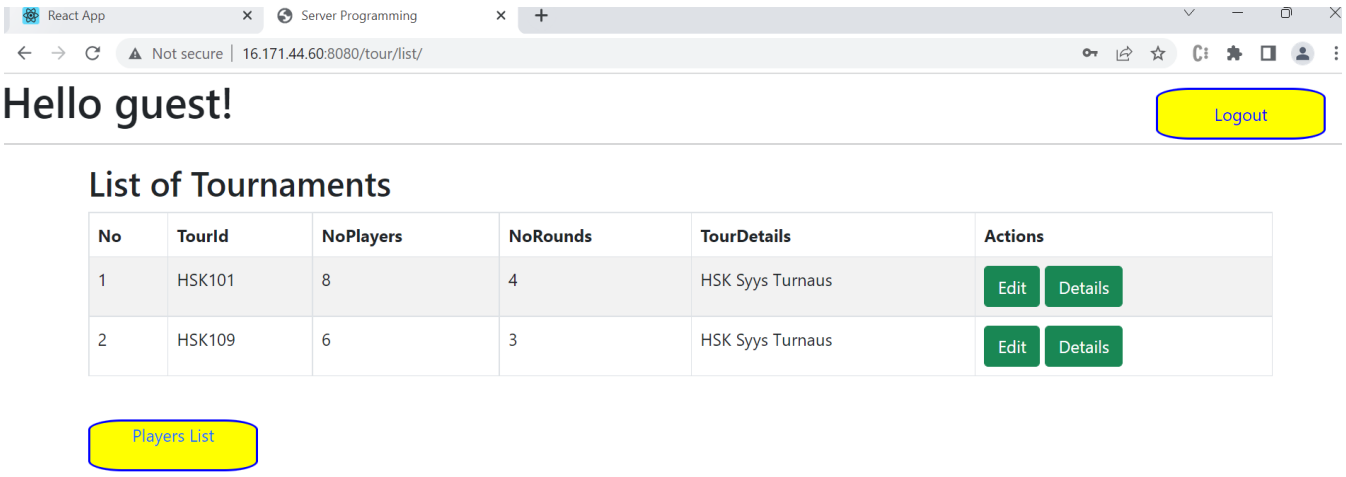

Figure 27. Main webpage (self-made)

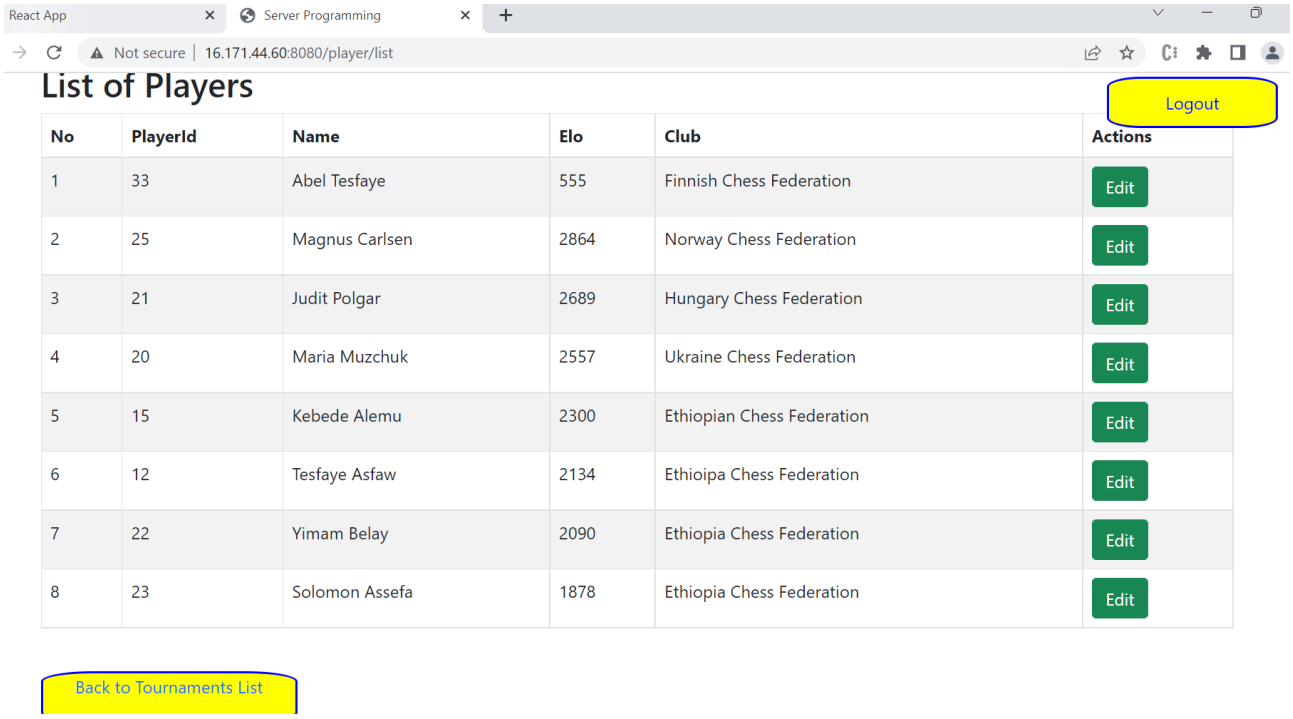

Figure 28. List of players' page.

4 Discussions

As stated above, the web application is now fully deployed and ready to be used by users. I have implemented all of the requirements I set in section 1.2. I had tested the web application thoroughly before I deployed it. It is working as expected. And I can consider this project to be quite a successful project. When I started this thesis project, I was not sure if I would be able to finish the project in this Autumn (2022). Because there were tasks in the project that I plan to execute but had no prior knowledge about it. For example, the deployment phase was completely new to me. I had no prior knowledge about AWS cloud services. I studied AWS cloud services and then proceeded to apply it to deploy the app. That is why I originally schedule to finish next year in January.

In my opinion the two main strong points of my web application are first that it is completely free and second its appealing graphical user interface (GUI). There are no fees to use any part of the application. I also plan to keep it this way.

When I designed the frontend's graphical user interface (GUII), I put a high emphasis on the intuitive usability of the website. One consistent idea I had when designing this app is that a new user should be able to start using my app intuitively. That is why I opted to use the five control buttons instead of adding combo boxes like: files, insert etc., which is commonly seen in Microsoft windows applications. And just in case the user still wants to check out the help pages, I have added a thorough documentations on how to use every section of the website.

I think the ease of using my web app would make it more appealing to use. I have seen a commercial chess tournament software before. True it provided more features than my app does, but it looked overtly complicated to use. A new user would need to spend quite a time on the software documentation pages before starting to use it. Even after that I noticed the organizer sometimes running into problems and having to visit those help pages.

The major weakness of my web application is that currently it supports only one type of chess tournament that is Swiss-system tournament. Other chess tournament types like round-robin and double round-robin are easier to implement than the Swiss-system tournament. But unfortunately, due to time constraints, I did not incorporate them.

Building this web application has enabled me to develop my skills as a software developer. I learned new skills; know more about the tools I used to build this app; appreciate the uses of software development methodologies. I have faced at times lingering problems. But having the right composure and persistence helped me to overcome them. I never sought anybody's help.

Sources

Abramov, D. 2015. "The History of React and Flux with Dan Abramov". URL: https://threedevsandamaybe.com/the-history-of-react-and-flux-with-dan-abramov/. Accessed: 04 November 2022.

Banks, A. & Porcello, A. 2020 Learning React. 2nd ed. O'Reilly Media. E-book. Accessed: 24 November 2022.

Bugl, D. 2017. Learning Redux. Packt Publishing. E-book. Accessed: 12 October 2022.

Davidson, J. D. & Coward, D. 1999. Java Servlet Specification ("Specification") Version: 2.2 Final Release. Sun Microsystems. pp. 43–46. Accessed:27 July 2008.

Dechalert, A. 2021. Is Visual Studio Code Really the Best Code Editor? URL: https://www.tabnine.com/blog/visual-studio-code-really-the-best-code-editor/. Accessed: 15 November 2022.

Dooley, J. 2011. Software Development and Professional Practice. Apress. E-book. Accessed: 6 October 2022.

Ferrari, L. & Pirozzi, E. 2020. Learn PostgreSQL. Packt Publishing. E-book. Accessed: 04 November 2022.

Fishpool, B. & Fishpool, M. 2020. Software Development in Practice. BCS Learning & Development Ltd. Swindon.

Freeman, A. 2011. The definitive guide to HTML5. Apress. New York.

Garreau, M. & Faurot, W. 2018. Redux in Action. Manning Publications. E-book. Accessed: 04 November 2022.

Hagos, T. 2021. Beginning IntelliJ IDEA: Integrated Development Environment for Java Programming. Apress. E-book. Accessed: 8 October 2022.

Haverbeke, M. 2018. Eloquent JavaScript. 3rd ed. No Starch Press. E-book. Accessed: 06 November 2022.

Heath, F. 2021. The Professional Scrum Master (PSM I) Guide. Packt Publishing. E-book. Accessed: 13 October 2022.

Hoffmann, J. 2019. What Does AJAX Even Stand For? URL: https://thehistoryoftheweb.com/what-does-ajax-even-stand-for/ . Accessed: 07 November 2022.

Hunt, P. 2013. Why did we build React? URL: https://reactjs.org/blog/2013/06/05/why-react.html. Accessed: 16 October 2022.

Johnson, B. 2019. Visual Studio. Wiley. E-book. Accessed: 8 November 2022.

Jones, C. 2013. The Technical and Social History of Software Engineering. Addison-Wesley Professional. E-book. Accessed: 10 October 2022.

Jyeett. 2021. What is Bcrypt and Why? URL: https://dev.to/jyeett/what-is-bcrypt-and-why-2dd1. Accessed: 13 October 2022.

Kollegger, E. 2018. What is Axios.js and why should I care? URL:https://medium.com/@Minimal-Ghost/what-is-axios-js-and-why-should-i-care-7eb72b111dc0. Accessed: 25 November 2022.

Krochmalski, J. 2014. IntelliJ IDEA Essentials. Packt Publishing. E-book. Accessed: 8 October 2022.

Le, Q. H. & Diaz, M. 2021. Developing Modern Database Applications with PostgreSQL. Packt Publishing. E-book. Accessed: 1 October 2022.

Long, J. & Bastani, K. 2017. Cloud Native Java. O'Reilly Media. E-book. Accessed: 21 November 2022.

Narayn, H. 2022. Just React! Learn React the React Way. Apress. E-book. Accessed: 08 November 2022.

Patel, H. React JS: What and Why? URL: https://harsh-patel.medium.com/react-js-what-and-why-e6cad2dfb4c3#:~:text=React%20allows%20developers%20to%20create%20large%20web%20applications,works%20only%20on%20user%20interfaces%20in%20the%20application. Accessed: 11 October 2022.

React Official Docs 2022. Create a New React App. URL: https://reactjs.org/docs/create-a-new-react-app.html. Accessed: 04 November 2022.

Schwaber, K. & Beedle, M. 2002. Agile software development with Scrum. Prentice Hall 2002. Upper Saddle River.

Schwaber, K. & Sutherland, J. 2016. The Scrum Guide. URL: https://scrumguides.org/docs/scrumguide/v2016/2016-Scrum-Guide-US.pdf. Accessed: 08 November 2022.

Soni, R. K. & Soni, N. 2021. Spring Boot with React and AWS: Learn to Deploy a Full Stack Spring Boot React Application to AWS. Apress. E-book. Accessed: 28 November 2022.

Varanasi, B. 2019. Introducing Maven: A Build Tool for Today's Java Developers. Apress. E-book. Accessed: 27 November 2022.

Venitta. 2017. What is a Web Service Endpoint? URL: https://www.biztalk360.com/blog/web-endpoint-monitoring-biztalk360/#:~:text=In%20simple%20terms%2C%20a%20web%20service%20endpoint%20is,HTML%20files%20or%20active%20server%20pages%20are%20exposed. Accessed: 27 November 2022.

Visual Studio Code Official Docs. Documentation for Visual Studio code. URL: https://code.visualstudio.com/docs. Accessed: 14 October 2022.

Walls, C. 2015. Spring In Action. Manning Publications. New York.

Wittig, M. & Wittig, A. 2018. Amazon Web Services in Action. 2nd ed. Manning Publications. E-book. Accessed: 27 November 2022.

Appendices

Appendix 1. GitHub Repository

Frontend URL: https://github.com/tesJS/chesstournament-app

Backend URL: https://github.com/tesJS/ChessTourApp

Appendix 2. Deployed Web Application URL

URL: http://chesstourapp.com/.

For demonstration purposes use the following credential to login:

User: guest

Password: guest