[Date]

Software Requirements Specification

Group Project: TLC Quantum

Table of Contents

Glossary

TERM	DEFINITION
RBAC	Role-based access control (RBAC), also known as role-based security, is a mechanism that restricts system access. It involves setting permissions and privileges to enable access to authorized users.
GDPR	The general data protection regulation. The EU General Data Protection Regulation (GDPR) governs how the personal data of individuals in the EU may be processed and transferred.

Introduction

This document will describe the important functions that are needed to be implemented inside of the quantum chess tournament. Also, it will show the behaviors that the users expect from this system, including additional attention to the most important ones: performance, scalability, security, and privacy.

Functional Requirements

The functional requirements will specify what the user expects the system to do. The Chess Tournament system can do many things, so it is good to first prioritize this and set an overview of how the user flow is. After that, we can further specify what is going to be important for the system functionalities by listing them one by one.

User Journey

Julia is a student and plays chess at the top level. At her campus, where she is studying Physics, she sees a poster that says, “Ready to be the top-level Quantum Chess Player”. She is thinking about joining. She looks up the website and starts to register.
To further investigate the journey of Julia, here is the backbone of how the system works, when Julia is trying to become the best player. This is viewed with the main steps and the things that involve these steps

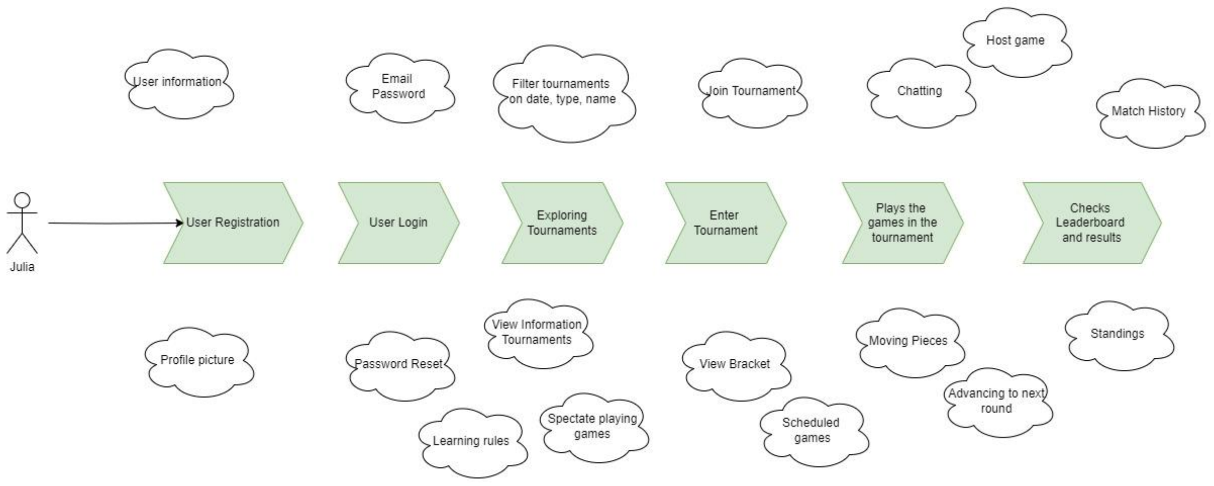

Figure 1: Julia's Journey

Now we have a clear overview of how a user can use the system. To go further in-depth, we have the specific functional requirements listed below.

Functional Requirements TLC Quantum

ID	Description	Priority (MoSCoW)	Remarks
FR-01	The user shall be able to log in to Quantum Chess Tournament System with email and password.	Must	
FR-02	The user shall be able to register as a user with basic information.	Must	
FR-03	The user shall be able to view the upcoming tournaments.	Must	
FR-04	The user can filter tournament searches by date, type, and name.	Should	
FR-05	The user shall be able to reset his password.	Must	
FR-06	The user shall be able to update, view, and delete all his personal information.	Must	First name, last name, and picture.
FR-07	The user shall be able to select a tournament to join.	Must	

FR-08	The user shall be able to see all the matches of a tournament that is in progress.	Must	
FR-09	The user shall be able to play quantum chess against another player.	Must	
FR-10	The user shall be able to chat with the player that is competing in the game with him.	Could	
FR-11	The user shall be able to view information about a specific tournament.	Must	
FR-12	The user shall be able to view moves made during a game.	Must	
FR-13	The user shall be able to view his own game history.	Should	
FR-14	The user needs to see the upcoming matches of the participating tournaments.	Must	
FR-15	The user shall be able to analyze his previously played games.	Could	
FR-16	The user shall be able to change his email.	Must	
FR-17	The user shall be able to view previous tournament results.	Could	
FR-18	The user shall be able to see a leaderboard of played tournaments.	Must	
FR-19	The user shall be able to see the brackets of tournaments.	Must	
FR-20	The user shall be able to view other people their game.	Must	
FR-21	The user shall be able to log out of the application.	Must	
FR-22	The user shall be able to view the rules on how Quantum Chess works.	Should	
FR-23	The user shall be able to see the results of the played match.	Must	
FR-24	The admin shall be able to make announcements about tournaments, updates, and deadlines.	Must	
FR-25	The admin shall be able to oversee complaints and reports, by reviewing them.	Must	
FR-26	The admin shall be able to create, edit, view, and delete a tournament.	Must	
FR-27	The admin shall be able to manage user information to inspect users.	Must	Aligns with the GDPR
FR-28	The admin shall be able to add users to upcoming tournaments	Must	
FR-29	The admin shall be able to create accounts for users.	Must	
FR-30	The admin shall be able to edit the activity of the user.	Should	

Non-Functional Requirements

The non-functional requirements are the behaviors that the system should have, and what the user expects from it. There are a lot of non-functional requirements that we need to keep in mind when developing the Quantum Chess Tournament System, such as scalability, maintainability, reliability, etc. The team mainly wants to focus on performance, scalability, security, and privacy while also keeping in mind the long-term project, so maintainability, understandability, portability, etc. To give further clarification, see the following points of the most important non-functional requirements for this application.

Performance

Making an application that attracts players from different schools, to play a tournament, needs to be a fast-paced application. People who play a live game need to have a seamless experience, otherwise, they will be dissatisfied fairly quickly, more on that after we discuss some issues first. This means performance is important to try and optimize this. Performance issues can come from different sides, therefore we need to further look into this, divided into four main categories.

1. **Memory Issues**: Inefficient memory assignment or leaks can lead to out-of-memory errors, causing an application to crash and be unresponsiveness. Optimal memory management is crucial for stable application operations. Also, it is good to consider how to handle configured memory limits, since when there is a big tournament that can lead to a high spectator count, this leads to spikes in traffic which can trigger out-of-memory errors.
2. **Database Issues**: Slow queries, large result sets, and insufficient database connections can impact our application responsiveness. When looking at chatting with the opponent during a game, this needs to happen in real-time, so slow performance database issues will lead to dissatisfaction of the user.
3. **Integration Issues:** Slow service calls and transactions with external systems can delay application response times, affecting user experience. This is important for the tournament system to keep in mind since we need to find a solution to implement the existing quantum chess game in some way inside the application.
4. **Infrastructure Issues:** Server capacity limitations and network connectivity issues can cause slowdowns or failures in application operations. Therefore, we want to limit as many of these issues by implementing good hosting/server solutions. Especially in tournament games, this is important since tournament games can become intense with the time limits, making connection extremely important.

(jogetworkflow, 2019)

Therefore, it is good to look at these issues throughout the project. Furthermore, while you might think what is the matter of having an extra waiting delay of one second during a switch between two pages? This matters, and there is a certain rule called, the 10-times rule. When we look at responses of 0.1 seconds, people will feel that their actions are directly happening. But when looking at one second, it can already feel that the computer is causing the result, although it doesn't bother the attention span of the user, meaning the user still feels in control. Then there are ten seconds, at this point, people there average attention span will max out, and the attention flow will break, leading to people leaving the website. Also, ten seconds is the time used to see if the page is okay or that bad that they are going to leave. Therefore, performance will also give a first impression of what the application will offer, since people are impatient on the internet, you need to satisfy them immediately or they are gone. (Nielsen, 2009)

Performance requests per minute

To give further information around performance, how many requests the system should handle in a minute which will give a rough estimate of the load that the application needs to handle.
To calculate this, we need the following formula:
$r = n/(T_{\text{response}} + T_{\text{think}})$

- n: the concurrent user, in our case we will pick the peak of the system, 100000 concurrent users.
- r: The amount of requests per second.
- T_{think}: The average time to think for each request. So the time to proceed to the next step, when you for example login into the system and then want to explore a tournament, the time between this is the 'think time'. This will be around 3 seconds on average.
- T_{response}: the response time at peak load, one second.

So the requests per second will be:
r = 100000 / (1 + 3)
r = 100000 / 4
r = 25000
So the load will be 25000 requests per second, meaning that each minute there will be 1.5 million requests. This all needs to be performed without any degradation in performance.

Scalability

The quantum chess tournament will be used by schools to host a tournament, for students or external people who are eligible to participate to play inside the tournament. These tournaments can reach up to 100 players. Furthermore, since Quantum Delta, is evolving internationally, it will be good to keep in mind that this application needs to be scaled so that users from other countries can also use the application. This will lead to even more demand. Also, right now we only need to focus on the game quantum chess, but it needs to be extendable so that it can support multiple games in the future, leading to even more demand. Let's therefore say, looking at the Netherlands we can assume that around 50 schools can use this, which can lead to 5000 concurrent users. While also thinking of further global extend, this can lead to even 200 schools, having a total of 10.000 concurrent users. Moreover, we can apply the 10-times rule, we will aim at 100.000 concurrent users that can use the quantum chess tournament system. This is set as are goal higher than what is needed, but is a good strive for peak usage, when also keeping the extendable ideas of more games, leading to more demand.
Additionally, when looking at these high amounts of demand, we can scale this in two ways:

- Horizontal scaling, providing more servers.
- Vertical scaling, adding more CPU and RAM to existing machines.

When keeping this in mind, the system will remain stable and maintain performance while the users, data, and business operations increase. (Nonfunctional Requirements in Software Engineering: Examples, Types, Best Practices, n.d.)

Security and Privacy

When thinking of security, we need a system that is from the beginning secure, without having to add more security layers later on. When security is not properly set up, this can lead to extreme damage to the system when an attack happens. There are four types of security to think about when looking at IT security.

- Network security: Looking at security between devices that are on the same network.
- End-point security: This is focused on the used devices, such as a laptop, mobile, computer, etc. This needs to be saved to avoid unwanted users intrude the system.
- Internet security: also known as cybersecurity, which deals with the transit and use of information.
- Cloud security: This is related to securing the risks in the cloud.

(What is Software Security and Why is it so Important Now?, n.d.)

Therefore, looking at security for quantum chess tournament system is important, and needs to be prioritized. To prevent attacks such as DDoS, unauthorized access, and data breaches, we need to implement and think of best practices such as RBAC, and keep in mind the importance of GDPR, and ISO27001. Furthermore, standards to gain knowledge and spread awareness around the development are looking at OWASP's top 10, which will show the most common vulnerabilities to a system. (Nonfunctional Requirements in Software Engineering: Examples, Types, Best Practices, 2023) For handling the data of the users, we need to strictly apply and follow the GDPR, so that the application can not receive a fine which can be up to 20 million euros or 4% revenue of the company. (Establishing Performance Goals)

Ethical Requirements

From the conducted research and findings from the ethical design document that the team made (Quantum TLC, 2024, Ethical Design), there are some important things to have a thought-out application that is ethically responsible.
Regarding security and privacy, we have taken into account several requirements, which are in (Quantum TLC, 2024, Security Design) that specify requirements that need to be validated and what tools to achieve these requirements.
Regarding the originality of the application, it can be more clarified using the guide and read more options. These requirements can be found back in the NFR list.
Furthermore, all the other findings were already considered by the group and noted down.

Non-Functional Requirements Of TLC Quantum Chess Tournament System

The non-functional requirements will be based on what is described above, but also other non-functional requirements that have added value to this project and are of importance.

ID	Description	Priority (MoSCoW)	Remarks
NFR-001	The system should be able to respond within a second when the user does standard operations.	Must	These behaviors are for all the standard operations in the application such as logging in, registering, joining tournaments, etc.
NFR-002	The system should be able to respond without noticing a delay when players compete in a game together.	Must	Smooth experience is a key priority when playing the game. This means around 0.1 seconds.
NFR-003	The system should be able to respond within 5 seconds on highly intensive tasks.	Must	Intensive tasks can cost more time such as generating brackets, advanced filtering, etc.
NFR-004	The system should handle peak usage of 10.000 concurrent users, without any performance degradations.	Must	Keeping in mind the 10-times rule, to handle up to 100.000 concurrent users.
NFR-005	Implementation of the quantum chess tournaments meets the standard requirements that are agreed on beforehand.	Must	
NFR-006	The system needs to be regulated and comply with the GDPR rules.	Must	
NFR-007	The system needs to meet the standard of having authentication and authorization functions.	Must	Protect the user data and secure features that are authorized for only certain types of roles.
NFR-008	The uptime of the system needs to be 99.99%.	Must	Downtime is needed to have frequent updates, especially at the beginning of development.
NFR-009	The system should support internationalization.	Must	The option to change language is mandatory for the system since multiple users from different countries will use this application.
NFR-010	The system should be capable of running on different browsers, without losing any performance, and the supposed behaviors.	Must	Run on browsers such as Mozilla Firefox, Chrome
NFR-011	The most used features need to be able to be easily accessed and with minimal navigation.	Must	
NFR-012	The system should be able to run on different OS while maintaining the same functions and performance.	Must	Windows, Mac
NFR-013	The system needs to be protected with 2FA for improving security.	Could	
NFR-014	The system should have automated tests in the CI/CD pipeline of	Must	Finding early on issues can spare a lot of time, compared to finding them later on.

	development, to quickly discover potential issues.		
NFR-015	The project should use proper version control, while properly applying best practices to this.	Must	
NFR-016	The system should be maintained with the standard coding principles that comply with the microservices architecture	Must	To comply with SOLID, design patterns in both code and architecture.
NFR-017	The system should have continuous improvement in processes.	Must	Having regular retrospectives, and feedback from the stakeholders to improve maintainability.
NFR-018	The system should have included an extensive user guide that is accessible to all users	Should	To justify the originality
NFR-019	The platform shall be able to provide a “Read More” option or tooltips at the beginning to offer additional information and context where needed.	Should	

References

jogetworkflow. (2019, 10 24). *How to Solve Your Enterprise App Performance Problems*. Retrieved from jogetworkflow: https://jogetworkflow.medium.com/how-to-solve-your-enterprise-app-performance-problems-2f412a9de635

Nielsen, J. (2009, 10 04). *Powers of 10: Time Scales in User Experience*. Retrieved from nngroup: https://www.nngroup.com/articles/powers-of-10-time-scales-in-ux/

Nonfunctional Requirements in Software Engineering: Examples, Types, Best Practices. (n.d.). Retrieved from altexsoft: https://www.altexsoft.com/blog/non-functional-requirements/

What is Software Security and Why is it so Important Now? (n.d.). Retrieved from cpl.thalesgroup: https://cpl.thalesgroup.com/software-monetization/what-is-software-security

Quantum TLC. (2024). Security Design (Unpublished manuscript), FontysICT.
Quantum TLC. (2024). Ethical Design (Unpublished manuscript), FontysICT.