6/3/2024

Research Report

Quantum TLC

LUCAS JACOBS

Table of Contents

Opportunity

The Quantum Talent and Learning Centers (TLC) Eindhoven is focused on advancing education in quantum tech, information, and sensing. The goal of Quantum TLC is to promote and ensure education at various levels, including WO, HBO, MBO, secondary, and primary schools, as well as those seeking a career change.

To help with their goal, we have been asked to make a software solution to make an online platform to host Quantum Chess Tournaments.

Quantum Talent and Learning Centers (TLC) Eindhoven has created a game called Quantum Chess, which is a variant of chess with dynamic effects of quantum mechanics. Right now, they can only play a game on one computer, making it so that users need to be physically together with one computer. The task that the software developers need to solve is to offer a software solution where users can use an online platform where they can join tournament-hosted games held by Institutes. This system is used for engaging and collaborative competition that spans all four Quantum Talent and Learning Centers (TLCs) in the Netherlands. A user can log in to the system, sign up for a tournament, and play online hosted games. The focus will be on a user-friendly design, participants can sign up and smoothly take part. This end goal is to have a working solution with a great approach to future working software developers for this project.

Research questions

Main question:

"How can an external quantum chess game be integrated into a scalable software platform with a large player base that can participate in tournaments while maintaining security, reliability, and privacy?"

Sub questions:

Methods:

1. What technologies and frameworks will best fit the quantum chess tournament system while looking at performance, scalability, and security?
    a. What back-end framework best fits our project?
    b. What building framework for the front end best fits our project?
        i What requirements does the project need to fulfil?
        ii How scalable does the project need to be?
        iii How big is the community and support of the framework?
        iv What browsers and devices need to be supported?
        v How flexible does the framework need to be?
        vi How frequently is the framework updated or is it at risk of becoming outdated?
    c. What database framework is a satisfactory solution for the tournament system?
    d. What architecture for the chess tournament system fits the needs of the non-functional requirements?

Methods:

- **Literature study:** Getting knowledge of the different available systems.
- **Community research:** look for communities to get their perspective on different kinds of software.
- **Comparison Chart:** Creating a comparison chart to evaluate various plugins based on factors such as cost and effort.

2. What are the specific needs and preferences of users interested in participating in a quantum chess tournament?
    a. Are there any specific prioritizations to the functionality of these people?
    b. How can we clarify and justify this in an elegant way to the stakeholders?

Methods:

- **User Requirements Exploration**: Gaining comprehensive insights into user needs and preferences for the proposed solution.
- **Focus Group**: Collecting diverse perspectives and insights on the issue at hand.
- Interviews: Conducting interviews with stakeholders to delve deeper into their opinions and requirements.
- **Stakeholder Analysis**: Identifying and addressing the needs of relevant stakeholders.
- **Prototyping**: Developing prototypes to materialize conceptual ideas.
- **Brainstorming**: Generating innovative ideas through collaborative brainstorming sessions. Also, in HackIT (week 1), having sessions with each other to produce ideas.

3. What are the technical requirements for the existing quantum chess game to communicate with the chess tournament system?
    a. What kind of software and tools are needed for communicating with the system while looking at performance?
    b. Are there any specific APIs that can be used to aid with communication to the quantum chess game, while having the consideration on vendor lock-in and performance?
    c. How do existing "e-sports" systems handle real-time updates on their platform?

Methods:

- **Literature study**: Getting to know what the embedding options are and how they work.
- **Available product analysis**: Look if any open-source projects implemented this kind of technique.
- **SWOT**: analysis to have a better overview of what can be done and what to look out for.
- **Code review**: Once versions are deployed on the embedding part, it is important to check the created code for certain security vulnerabilities and optimize performance.
- **Task analysis**: How is this made possible to integrate it?
    - **IT architecture sketching**: Based on the information and flow, make a diagram together to understand it properly.

4. What are the legal and regulatory requirements for hosting online gaming tournaments, with the consideration of GDPR?
    a. What are the main data protection rules/requirements?
    b. How do we apply the measurements of GDPR in our application?

Methods:

- **Literature study:** Looking for the regulations of the GDPR. What is it? How to apply for it?
- **Community research:** Get to know the ways other people have implemented these GDPR regulations.
- **Available product analysis:** Looking for existing products that apply regulations and looking for solutions that can be applied to this project as well.
- **Problem Analysis**: Thoroughly understanding the problem landscape. So why GDPR?

5. What are the best practices when it comes to the security of the website and the zero-tolerance cheat policy in online gaming tournaments?
    a. Which security measurements are considered when it comes to online gaming platforms?
    b. How can cheating be detected and prevented effectively?

Methods:

- **Literature study:** getting to know what the best options are to securing this application, and what the best software solutions can be against cheating in a game.
- **Best good and bad practices:** Looking at secure system solutions.
- **Comparison Cha**rt: Creating a comparison chart to evaluate various plugins based on factors such as cost and effort.

6. What are the main user interfaces and experiences expected when it comes to online hosted tournaments?
    a. What design elements in an online gaming platform contribute to a user-friendly application?
    b. How can we improve the experience when it comes to using the application on different browsers?
    c. What are the existing online tournament platforms, and how do these platforms operate?
        i How do these platforms approach UI/UX?
        ii What are the main strengths and weaknesses of these platforms?

Methods:

- **Product review**: Throughout the project, having enough meetings with each other, and discussing the looks with the stakeholders.
- **Usability testing:** Other users can investigate the product and give feedback. With this test, we can assess certain design choices and fix issues before the actual product goes live.
- **Community research:** look at what people find a nice UI/UX regarding a tournament system.
- **Literature study:** Look at the best design principles for creating a clear UI/UX.
- **Problem Analysis:** Thoroughly understanding the problem landscape.

7. How can we make sure that hosting and infrastructure are performing stable enough during a tournament?
    a. What hosting solutions can oversee a minimum of fifty-plus users at a concurrent time, with the option to automatically scale when needed?
    b. How can we maintain the performance and limit potential bottlenecks?

Methods:

- **Code review**: Look at if the structure of the hosting is set up correctly. The system is separated with the correct specifications.
- **Literature study**: Looking at the best solutions to performing and having a stable application.
- **Model validation** (ML): check if the system can oversee the expected traffic.
- **Tinkering**: Engaging in trial-and-error coding to identify optimal solutions.

8. What metrics are available to validate the product that will be used by Quantum TLC Eindhoven to ensure its quality?
    a. Are there any criteria that need to be met to say the system's satisfaction, keeping in mind any standard requirements?
    b. What testing and quality measurements can be considered to ensure the quality of our product?

Methods:

- **Product Review:** Giving a demo and reviews to the stakeholders every sprint, to get continuous feedback from them. This gives a good insight into what requirements need to be met.
- **Unit test:** verify that the logic inside the code is working as expected. Make sure the code is up to quality and prevents unwanted behavior.
- **Code review:** When a person is done with his functionality, another person in the group will review the code to find bugs, coding standards, and improvements.
- **System tests**: having a document with different test flows of functionalities to assess a case with the expected outcome.
- **Peer Review:** Collaboratively reviewing code using GIT to detect and rectify minor errors that may slip through unnoticed.

Deliverables

As mentioned for in the project, (Quantum, 2024, Project Plan, §3.1), there are specific deliverables for the project itself. Other than those documents we have the following:

- Process report, tracking a log on who did what on the research.
- Research Report: research question with the applied methods on how they have been approached and concluded/result. Also the results, a conclusion, and a reflection on what we have learned and can improve on.

Time estimation

To give an overview of when a sub-question is supposed to be ‘done’, here are the sprints in which they are expected to be finished. Note that we specify multiple sprints for a question since there is always improvement or more feedback to be implemented when showing the solution.

Sprint 0-1: Sub-Questions 1

Sprint 2-3: Sub-Questions 2, 3, and 6

Sprint 3-5:Sub-Questions 4, 5, 7, 8

Technology Readiness Level

Were using the following measurement table of the Technology Readiness Level.

Figure 1: Technology Readiness Level Indicator (WHAT ARE TECHNOLOGY READINESS LEVELS (TRL)?, n.d.)

In week 16 of our semester (June 9th, 2024), we had to deliver a fully functional and operational application that had to host around 40 concurrent people at the same time. These functionalities had to be set, experimented in concept, validated, and eventually deployed so they could be used by the end user. The system had to be built fast in this stage since we got the request around 3 weeks before the system needed to be done. Even though the scale of Technology Readiness Level had proven that our application is a 7, this application deployment was for demonstration purposes only, and it could handle tournament users. It is far from done, having several functionalities yet to be completed. Only when we had the time to also make proper testing for each scenario and did not have a 3-week notice on creating something, then it have a readiness level of 8 and even 9.

Activities to focus on

1. Proper testing and validation
    a. Objective to go from TRL 7 to TRL 8.
    b. Activities: have various testing scenarios, including performance, security, and UAT. Which is all validated in the operational environment.

2. Complete Functionalities
    a. Objective: Addresses the lack of functionalities operating for the tournament system and notification system.
    b. Activities: Go through the user stories and develop and integrate the remaining features. Focus on the completion of all planned functionalities to make sure the system meets all user requirements and specifications.

By applying this in receiving feedback and iterations. Furthermore, looking at optimizing scalability and performance, security, and privacy it will make sure that the product can become fully operational and go from a current TRL of 7 to eventually a TRL of 9, to achieve a fully functional and scalable system, which is ready for large-scale deployment.

Results

To give the main question a concrete answer, we made several sub-questions that need to be answered during this semester, with the usage of DOT-Framework. We will go over each sub-question with the conclusion to it.

Sub-Question 1: What technologies and frameworks will best fit the quantum chess tournament system while looking at performance, scalability, and security?

For the front end, we came up with the following, using a comparison chart to conclude:

Based on the factors that we looked at we created a table where you can see in an overview how each framework performed for that topic. Below the table, you can see the legend that displays for each framework what the conclusion was after our research. If you want to know more about one of the answers you can see more about it in the sub-questions above.

	React	Angular	Vue
Scalability	++	++	++
Performance	+	+	+
Support	++	++	++
Team skills	++	+	+
Flexibility	++	-	++
Compatibility	++	++	+
Security	+	++	+
Maintenance	++	-	++
Back-end compatibility	++	+	+
TOTAL	16	6	14

- = Dislike
+ = General
++ = Preference

As you can see the outcome of our research is to React, it has no downsides for the features and application that we want to create. Vue comes a close second but because our team is more familiar with React and we know how to connect it to the back end that we created we want to use React as the front-end framework.
Angular has a few benefits in comparison to the other two frameworks but since we want the flexibility of playing with libraries and setting the application up ourselves Angular has downsides for us which brings it to the lowest outcome for this project. To look further into how we concluded our research, see (Quantum TLC, 2024, Front-End Research Document).

For the back end we had the following:
In conclusion, all three platforms are suitable for creating enterprise software. Based on this research and our experience from the previous semesters we have chosen .NET Core for our back end. It offers the most flexibility, very extensive documentation and resources and is created for high performance. This comparison table gives an overview of all the platforms and .NET Core is the winner.

	Jakarta EE	Spring	.NET Core
Scalability	+	+	+
Performance	+	+	++
Support	+	++	++
Documentation	-	+	++
Team skills	-	+	+
Flexibility	+	+	+
Architecture Compatibility	-	++	++
Security	+	++	++
TOTAL	2	11	15

- = Dislike
+ = General
++ = Preference

To follow up on that, more information is found in (Quantum TLC, 2024, Research back end).

Furthermore, the architecture choice came down to the following.

In conclusion, the types of architecture we investigated were types of architectural types that define large service models and how they behave. Client-server architecture states that a client and a server communicate often using an API. This way it allows for data sharing and using it on every machine. SOA states that, instead of a monolithic approach, we split the project into smaller services which allows for reuse and easier integration with other existing services (through API calls e.g., HTTP calls) at a minor cost of performance and cost. Service-oriented is defined by splitting the product into smaller, contained services to allow for easier change and integration. While Event-driven architecture prioritizes performance and efficiency by having interested parties listen and respond. Serverless relies on the product owner hosting their services on a third-party platform, reducing cost and maintenance as well as handling architecture. Microservices is an implementation of Service Oriented Architecture that uses HTTP as its way to communicate with other services. Often used in conjunction with serverless, microservices are highly scalable and maintainable while its cost and performance issues can be solved by integrating serverless.
These architectures help with the development of software by deploying patterns to define structures that facilitate faster development through commonalities and processes that have been tested rigorously and are adopted by industries that consider them standards.

Moreover, the last framework we practically had to choose from, came down to the following conclusion based on a SWOT analysis that has been conducted.

So based on these SWOT Analyses, we can see they can have the same benefits and some threads such as the competition. Since the team doesn't have a budget, Oracle and Microsoft SQL are not a good option. Therefore in this following order, we wanted to prioritize the database usage, since we may want to use multiple databases.

1. MySQL: Scalable, community support, it has a high performance, and is adopted in multiple industries making it one of the options for this project with the limited budget and a need for reliability.

2. PostgreSQL: Great for this project, with great functionalities like those of MySQL.
3. MongoDB: Document-oriented, high performance, and efficient for semi-structured or unstructured data, which can be rapid in deployment, it uses a different approach to the traditional data structure.
4. Oracle Database: Longstanding, good features, but can become quite expensive.
5. Microsoft SQL Server: Great enterprise solution, but this also can become expensive.

More information is found in (Quantum TLC, 2024, Database analysis).

Therefore, with these conclusions, we got to choose several technologies. This concludes our research and keeping in mind the non-functional requirements, we will apply these technologies inside of our architecture.

Sub-Question 2: What are the specific needs and preferences of users interested in participating in a quantum chess tournament?

To have a particular answer to this question, we did self-research and applied methods such as User Requirements Exploration and Stakeholder analysis. Also, we did an ethical design which found the following conclusion.

From the findings that have been found from the TICT subjects and the Tarot Cards of Tech, we made some new requirements that are important for the development of the tournament system. For the security requirements, they are in the security design document (Quantum TLC, 2024, Security Design). Moreover, it specifies the Furthermore, in the (Quantum TLC, 2024, Software Requirement Specification, Ethical Requirements) the other specified findings are specified. More can be found in the document itself, (Quantum TLC, 2024, Ethical design)

Also inside (Quantum TLC, 2024, Design Document) the specific needs and the wireframes of how the front-end is going to be built are documented, which was all based on the specific needs of the user.

With this research that we have done, we came to the conclusion that it consists of concrete functional and non-functional requirements. These are a lot of requirements, and therefore we stated them inside of the software requirements specification, see (Quantum TLC, 2024, Software Requirement Specification).

Sub-Question 3: What are the technical requirements for the existing quantum chess game to communicate with the chess tournament system?

To answer this question, we investigated what people expect from our application, which was done by having a lot of meetings with the client. Also, having progress meetings to validate to the client if they agree with this. During this semester and research, we had a meeting with the CEO of Quantum TLC, which changed the course of our scope. Therefore, this also gave us a direct answer to what we need to use for our application to communicate with a game.

Sub-Question 4: What are the legal and regulatory requirements for hosting online gaming tournaments, with the consideration of GDPR?

With this research that is done and stated in (Quantum TLC, 2024, GDPR), it came down to several requirements that the Tournament system is going to implement.
We will handle GDPR compliance by handling the data with care. For the user, we will store their information privately and only for authorized persons. For the storage of login credentials, we use OAuth provided by a third-party application, which is trusted by big companies such as FedEx. Moreover, data is only collected when needed. Lastly, with Azure Cloud, we have a secure environment, using the protection Azure provides.

Sub-Question 5: What are the best practices when it comes to the security of the website and the zero-tolerance cheat policy in online gaming tournaments?

From the research we have done in (Quantum TLC, 2024, Security Design), we came up with several requirements that the application needs to adhere to. It is critical to have a secure application to prevent fines for Quantum TLC or even further punishment. By including automatic security tests and having proper SSDLC we will keep applying requirements while keeping in mind the security.

Sub-Question 6: What are the main user interfaces and experiences expected when it comes to online hosted tournaments?

The look and feel are important when it comes to such a system. With the research methods we applied, we came to several wireframes, that are keeping standard of various design principles, which include the importance of expected behavior, fast responses, and having good error handling. More information is found in (Quantum TLC, 2024, Design Document).

Sub-Question 7: How can we make sure that hosting and infrastructure are performing stable enough during a tournament?

First, we investigated cloud solutions and the benefits of the services that we can use. With the requests to the client with the balance of how much the services are going to cost, we got a budget and it fitted our infrastructure and kept our ability to host the application in a stable environment. This is further explained in detail in (Jacobs, 2024, Cloud Research).
Furthermore, with the actions described in (Jacobs, 2024, Technical Design), we had several actions taken into our architecture, that will make it possible to scale our application, as our traffic increases.

Sub-Question 8: What metrics are available to validate the product that will be used by Quantum TLC Eindhoven to ensure its quality?

By applying the research methods, we came down to using an automated pipeline in GitLab to validate the following stages: build, test, security, publishing, and deploying. Furthermore, having a structured way of working, by using Teams to keep track of our Agile way of working, branching strategies in git, and monitoring inside of the deployment environment to analyze incoming traffic and how the application will react to it. More detail is explained in (Quantum TLC, 2024, DevOps Report) and in (Jacobs, 2024, Coding Guidelines) explaining the branching strategy.
Moreover, with the research regarding testing, having a more in-depth look at (Quantum TLC, 2024, Acceptance Test plan).
The testing pyramid and various other testing methods are a fundamental concept for automated testing and guiding an agile project to create a balanced and effective test environment. While the original pyramid needs some adjustments to fit modern software development practices, it is still a core principle that continues to be essential for maintaining the tournament system of Quantum TLC to be a robust and reliable application.
This came down to knowing what tests we were going to perform to ensure the product was up to standard.

Main Question Conclusion

To integrate an external quantum chess game into a scalable software platform that will be used for hosting tournaments; while looking at security, reliability, and privacy, we want to address them one by one.

With the choices of our technology, which was a crucial decision. The conclusions made were based on optimizing and using technologies that support and ensure optimal performance, scalability, and support. With these technologies, having flexibility, security, and a lot of documentation available, this is essential for handling the demands of large player bases.

Next, the choice of architecture will ensure that with the choice of microservices, we allow for a flexible, scalable, and easier integration with the quantum chess game and other services in the future. Further, adhering to the GDPR, we will make sure privacy is optimized. The implementation of proper data handling, such as storing our sensitive information securely and minimizing data collection, helps build trust with the user and limits the risk.

Moreover, the security measurements that have been taken, such as encryption, authentication mechanisms, and security tests will ensure the security of the application.

Lastly, for the infrastructure and hosting, we are down to having a scalable, stable, and cost-effective solution. With the help of monitoring and scaling strategies, we can make sure that the platform will handle the expected load during tournaments, without losing performance.

So, in conclusion, by applying all these methods and technologies combined, we can make sure to have a scalable software platform for tournaments that maintains security, reliability, and privacy. With this approach and having a continuous approach, we will ensure that users will have a fair tournament experience, which can handle large amounts of players while securing user data and maintaining platform integrity.

Reflection

The research of this project went great. Even though we had a change of scope during the last quarter of the semester, we managed to complete this research which also added value to the changed scope. With our clear research plan, which included our strategy for approaching the questions, we could easily do research on our own. With having clear deadlines set with the client and each other we easily managed to complete our research. Before a question was specified one or more persons would have to look at the research before we could mark it as finished in the sprint. What we could have done better was applying from day one APA formatting, to prevent complications in the future. Also, the research questions were in our opinion a bit long and could have been shorter, or split up into shorter sub-questions. The questions did contribute to our implementation and we discussed them a lot to try and have the best fitting questions.

Contribution

Research:

Sub-Question 1: A team effort.

Sub-Question 2: A team effort.

Sub-Question 3: A team effort.

Sub-Question 4: Gabriela.

Sub-Question 5: Lucas

Sub-Question 6: A team effort

Sub-Question 7: Maike and Lucas

Sub-Question 8: Esther and Lucas

Research report: Lucas

References

WHAT ARE TECHNOLOGY READINESS LEVELS (TRL)? (n.d.). Retrieved from twi-global: https://www.twi-global.com/technical-knowledge/faqs/technology-readiness-levels

Quantum TLC (2024). Acceptance Test plan (Unpublished manuscript), FontysICT.
Quantum TLC (2024). Architecture Research (Unpublished manuscript), FontysICT.
Quantum TLC (2024). Cloud Research (Unpublished manuscript), FontysICT.
Quantum TLC (2024). Coding Guidelines (Unpublished manuscript), FontysICT.
Quantum TLC (2024). Database analysis (Unpublished manuscript), FontysICT.
Quantum TLC (2024). Design Document (Unpublished manuscript), FontysICT.
Quantum TLC (2024). DevOps Document (Unpublished manuscript), FontysICT.
Quantum TLC (2024). Ethical Design (Unpublished manuscript), FontysICT.
Quantum TLC (2024). Front-End Research Document (Unpublished manuscript), FontysICT.
Quantum TLC (2024). GDPR (Unpublished manuscript), FontysICT.
Quantum TLC (2024). Security Design (Unpublished manuscript), FontysICT.
Quantum TLC (2024). Technical Design (Unpublished manuscript), FontysICT.
Quantum TLC (2024). Research back end (Unpublished manuscript), FontysICT.
Quantum TLC (2024). Software Requirements Specification (Unpublished manuscript), FontysICT.
Quantum TLC (2024). User Stories (Unpublished manuscript), FontysICT.
Quantum TLC (2024). Project Plan (Unpublished manuscript), FontysICT.