2/26/2024

Front-end framework

Research Framework

TLC Quantum Chess
FONTYS

Table of Contents

Introduction

In this document, we are going to research what building framework best fits our project for creating the front-end. To do this, we will look at several sub-questions and finally conclude. In this document we are going to look at three different frameworks and compare them. The frameworks are: Vue, Angular, and React. We have chosen these frameworks because we have worked with these before, and they are commonly used and have documentation online.

To give a quick background in the project for which we will create the front-end, we will create an application for the Quantum TLC Eindhoven, they are 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.

We have been asked to make a software solution to make an online platform to host Quantum Chess Tournaments. Quantum 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.

What building framework for front-end best fits our project?

What requirements does the project need to fulfill?

The application needs a front-end where you can participate in tournaments and play quantum chess games. To split up the requirements we have split up the project in three parts: the tournament, the users, and the game.

Tournament

The application is going to have tournaments that can be of different types and durations, such as blitz which is short matches where you have to think quickly, to longer matches where you have more time to think.

The tournament is expected to be having 30-50 people per game. The functionalities that a tournament has are:

- View ongoing tournaments
- See upcoming tournaments
- View overview/score tournament
- See a scoreboard
- View games of the tournament

User functions:

The application is going to have three types of users: admins, normal users, and moderators. Each user has its own set of actions that it can do.

Normal users

- Login
- Register
- Notification settings
- Edit personal information
- See your history game/tournaments
- Settings
    - Change language
    - Change color design

Admins

- Add tournament
- View users
- Edit users
- Edit tournaments
- Add organizations

Moderators

- Add tournament
- View users
- Edit tournaments

In game functions:

The application is going to work with an embedded game where you can play it on the website as well as converse with your opponent and view other people play the game.

- Play the embedded game
- See all the previous moves
- Chat with the opponent
- See how the tournament in general is going
- Spectate the game

How scalable does the project need to be, and does this work with the building framework?

What are the functionalities of the application?

Based on the functionalities that we described in the first sub-question; we will now look into how scalable this needs to be and how each framework handles this. So far it is not clear yet how many users the application can expect, but we are expecting between 30 and 50 players per tournament entries. So, at a minimum it should be possible for 50 players to play at the same time. But to be sure we want to look at 70 players at the same time in case we want to upscale later.

Vue

Vue is designed to be efficient and scalable, allowing for efficient updates and rendering.

React

React uses component-based architecture making it well-suited for scalable applications. It has a one-way data flow which contributes to efficient updates and enabling a smooth performance even when the number of users increases. Besides that, it has a large library for optimizing performance.

Angular

Angular has built in features for scalability such as dependency injection and ahead-of-time compilation. Its architecture is created to easily scale and maintain large applications and because it’s built-in, it's easy to enhance scalability and reduce load times.

Conclusion

In conclusion all three frameworks are able to handle 70 embedded games at the same time with a load of at least 70+ players as well. Each framework offers its own way of features and tools to look into scaling the project.

(Literature study, library)

How big is the community and support for the framework?
Why is this important?

For any framework, strong community support is essential since it improves user experience and promotes ongoing development. A sizable community offers developers an extensive selection of tools, such as forums for users, tutorials, and thorough documentation, which simplifies the framework's learning and use. Furthermore, the continuous development process is powered by this vibrant community involvement, guaranteeing frequent updates, problem fixes, and the addition of innovative functionality motivated by insightful input. Additionally, the diverse user base increases the likelihood of encountering and resolving issues quickly, preventing development bottlenecks and enabling seamless progress.

React

React has a vast community of users. There are a few React-related communities such as: DEV’s React community and Reddit’s React community. They are conveniently accessible through platforms such as Twitter, Reddit, Stack Overflow and Discord. Such accessibility greatly facilitates new developers in seeking guidance and support for mastering React or troubleshooting specific issues within the framework.

Angular

Angular's community is also extensive and conveniently accessible through platforms such as Twitter, Reddit and others. As mentioned, this brings many advantages to the new developer and seeks guidance and support.

Vue

Vue has a bit smaller community and support than React and Angular, however it is growing vastly and in no time can reach them.

Conclusion

In conclusion, React and Angular have large followings from the community, and Vue is gradually building a strong community. As a result, each framework provides a similar amount of community and support, making the decision between them less crucial in this aspect.

(Literature Study, Library)

What is the expertise of the development team?

What have we worked and familiar with?

While there may be overlapping features and concepts among different frameworks, each one possesses its own distinct methodologies and approaches to development.

Our entire team has gained experience in React during our third semester at university. This familiarity with the framework is an advantage as it facilitates smoother progression in our development endeavors. Our collective expertise in setting up and navigating React minimizes the risk of wasted time typically associated with acquainting ourselves with alternative frameworks.

How flexible is the framework and how much flexibility do we need?

What are the functionalities of the application and the framework solution?

The application has several functions, we have split these up into the categories that you can see below:

Tournament functions:

- View tournaments ongoing
- See upcoming tournaments
- View overview/score tournament
- See a scoreboard

For the tournament functions displaying the tournaments should be easy for all three frameworks, scoreboard needs real-time updates which can be achieved with libraries, for example Socket.io which all three frameworks can use.

User functions:

- Login
- Register
- Notification settings
- Edit personal information
- See your history game/tournaments
- Settings
    - Change language
    - Change color design

Login, register and user profile are standard functions that all the frameworks can easily achieve, notification settings might need third-party libraries but also this is supported by all three of the frameworks.

Admin functions

- Add tournament
- View users
- Edit users
- Edit tournaments
- Add organizations

Managing tournaments and users requires CRUD operations which will be done with the back-end API. All three frameworks provide ways to handle forms and interact with the back-end.

In game functions:

- Play the embedded game
- See all the previous moves
- Chat with the opponent
- See how the tournament in general is going
- Spectate the game

For in game functions we need to integrate as well and this is also possible with all of the three frameworks.

Conclusion

For each framework there is a solution to add the function that we have thought of, some might be easier with a some framework but for each framework it is possible. React and Vue are highly flexible and have a lot of freedom in how to structure the application. Angular on the other hand is less flexible in comparison to React and Vue and provides a more structured setup and has a steep learning curve. So if you prefer to have more control and choose your own tools and libraries React and Vue are easier.

What browsers and devices need to be supported and does the framework ensure cross-browser compatibility?

Given our requirement to host Quantum Chess Tournaments on the website, it's imperative that the chosen frameworks offer robust support for the game. Ideally, they should also be compatible with mobile browsers to enhance accessibility for our target audience. Moreover, ensuring the game can be played seamlessly across all browsers is essential for maximizing user engagement and participation.

Based on our current understanding, the game was developed using Unity, which presents a favorable scenario for us. Unity offers the functionality to export the game as a WebGL build, enabling seamless embedding on the website for our target audience to enjoy and engage with.

React

React's seamless integration capabilities with various tools and libraries make it an ideal choice for our project. Particularly advantageous is the availability of modules designed to facilitate the embedding of WebGL builds within React applications, aligning perfectly with our requirements. Additionally, React's inherent cross-browser compatibility ensures a smooth and consistent experience for users across different browsers.

Angular

In contrast to React, Angular may encounter challenges with third-party integrations. Nonetheless, it does support Unity WebGL builds and allows for embedding within Angular applications, albeit potentially with some obstacles to overcome. While Angular, like React, boasts cross-browser compatibility, it's worth noting that there are common issues associated with Angular that may need to be addressed during development.

Vue

In Vue, integrating third-party components shares similarities with React in terms of ease, yet it may have some limitations. However, Vue's support for Unity WebGL builds is a notable advantage, especially considering potential constraints on third-party integrations. While Vue is compatible with major browsers like Firefox, Chrome, and Safari, it's prudent to conduct thorough testing across other browsers to ensure a seamless and consistent user experience.

In Conclusion

Fortunately, all frameworks support Unity WebGL builds and exhibit cross-browser compatibility. However, in this aspect, React emerges as the optimal choice. Its robust support for third-party integration, facilitated by a rich array of tools and libraries, combined with its seamless cross-browser compatibility, positions React as a top contender. While Vue also offers relatively straightforward third-party integration, its limitations in this regard may pose challenges during development. Additionally, rigorous testing across browsers is essential to mitigate potential compatibility issues. Angular, on the other hand, covers both criteria of supporting Unity WebGL builds and cross-browser compatibility, yet it may encounter hurdles with third-party integration. Therefore, considering these factors, React stands out as the most advantageous option for our project.

(Literature study, Library)

Does the framework have security features?

What security protections do we need?

There are several security features that we need to look into for the website to be safe:

- Input validation
  - By validating user input you make sure that only safe data is processed by the back-end.
  - Vue has available libraries that you van use to validate input validation.
  - Angular provides build-in features that you can use to validate input validation.
  - React needs to use third party libraries that can check input validation such as Formic.
- Secure communication
  - Ensure data transmitted between server and client is encrypted preventing interceptions, for example of user data.
  - All three the frameworks can use HTTPS for secure communication.
- Preventing Cross-Site Scripting
  - Prevent attacks from executing unauthorized actions on behalf of authenticated users.
  - Vue and Angular provide build-in sanitation methods to prevent XSS attacks.
  - Angular offers JSX which helps mitigate XSS by escaping any content rendering within curly brackets.
- Secure storage of details
  - Storing credentials secure on the client-side helps prevent unauthorized access to sensitive information.
  - All three frameworks handle secure storage of sensitive information, implementation between the three may vary but all three have the same core principles for secure storage.
- Security auditing
  - This helps identify threats before they have a chance to escalate.
  - This is the same for all three frameworks since this is not specific to a certain framework.

These are the security protections we will take regarding the front-end.

Conclusion

In conclusion, while the specific implementation may vary between the frameworks, the protections that we want to achieve in the front-end are possible in all the frameworks with it's own libraries and tools.

Angular is the easiest in this because of it's built-in security features that help mitigate web security vulnerabilities. Vue and React both don't have built-in features for security but do provide the developers with a flexibility to implement their own security measures according to the application needs.

How frequently is the framework updated or is it at risk of becoming outdated soon?

As it was stated in earlier question, these frameworks are still relevant because of the active community involvement that drives ongoing updates. Over the course of our project, this dynamic ecosystem will ensure that they adapt in line with evolving best practices and technology advancements, protecting against obsolescence. As a result, we can depend on these frameworks to deliver dependable and modern solutions, allowing us to effectively and swiftly satisfy both present and future objectives.

Conclusion

To summarize what we researched in this document we will give a small recap of the outcome per sub question:

Scalability

In conclusion all three frameworks are able to handle 70 embedded games at the same time with a load of at least 70+ players as well. Each framework offers its own way of features and tools to look into scaling the project.

Flexibility

For each framework there is a solution to add the functions that we have thought of, some might be easier with some framework but for each framework it is possible. React and Vue are highly flexible and have a lot of freedom in how to structure the application. Angular on the other hand is less flexible in comparison to React and Vue and provides a more structured setup and has a steep learning curve. So, if you prefer to have more control and choose your own tools and libraries React and Vue are easier.

Security

While the specific implementation may vary between the frameworks to get the same quality of security, the protections that we want to achieve in the front-end are possible in all the frameworks with their own libraries and tools.

Angular is the easiest in this because of its built-in security features that help mitigate web security vulnerabilities. Vue and React both don’t have built-in features for security but do provide the developers with a flexibility to implement their own security measures according to the application needs.

Community and support

React and Angular have large followings from the community, and Vue is gradually building a strong community. As a result, each framework provides a similar amount of community and support, making the decision between them less crucial in this aspect.

Compatibility

All frameworks, including React, Vue, and Angular, support Unity WebGL builds and cross-browser compatibility. React stands out due to its strong support for third-party integration, facilitated by a wide range of tools and libraries, along with seamless cross-browser compatibility. Vue also supports third-party integration but may face limitations, while Angular covers both criteria but might encounter challenges with third-party integration. Considering these factors, React is deemed the most advantageous option for the project.

Team Skills

The team's experience with React from university makes development smoother. This expertise reduces time wasted on learning other frameworks.

Comparison table

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 comes 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.