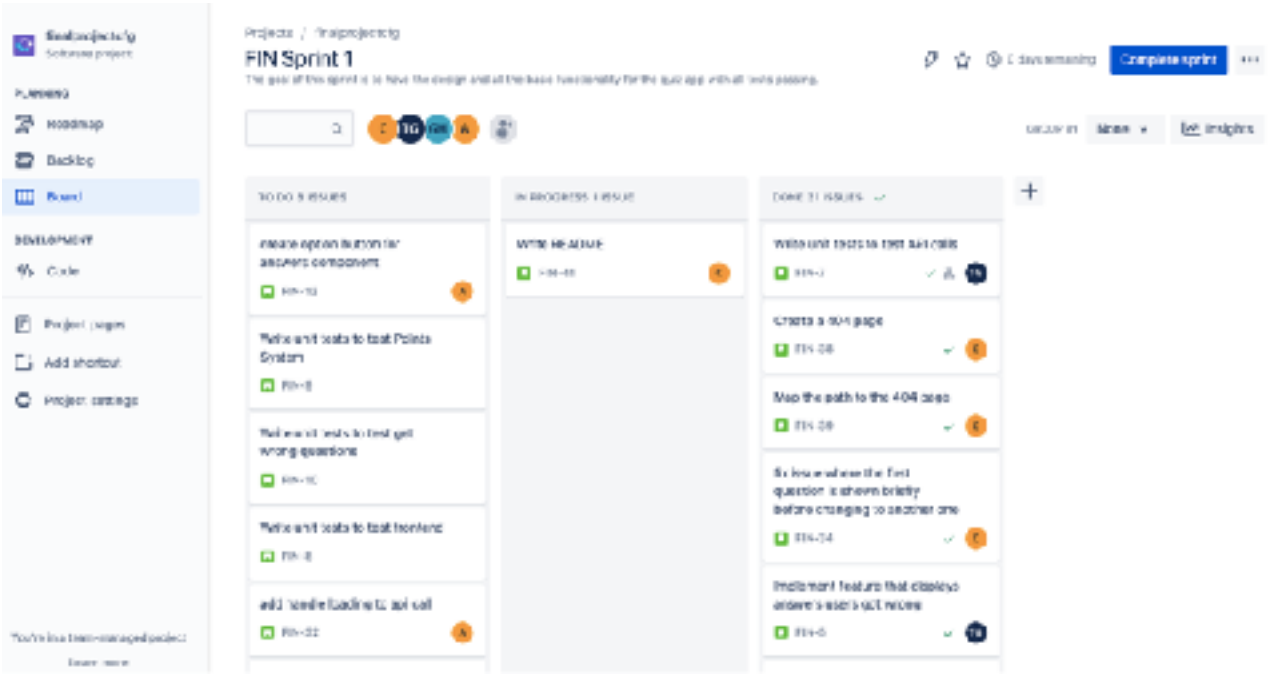
Project report

Aims and objectives:

Our aim with this project was create a quiz game web app. The user can choose the category and difficulty of the questions they want to play. At the end, the user can review the questions they got wrong. Out objective was to create a fun game that is also educational.
The typical user for this quiz app are everybody who loves quizzes and people who seek to improve their knowledge about various subjects.

Roadmap

We divided the project in small tasks, and added the tasks to a Jira board, where each person could assign to themselves the tasks they wished to work on. For the first sprint, the goal was to have the basic design and all functionality working. Then the last week (second sprint) was to work on the things we didn't complete on the first sprint, add finishing touches and work on additional features if we had enough time.



Background

Our quiz game starts with greeting the player and asking for their name. Then the player can choose the category they want to play and the difficulty of the questions. The game works by presenting the player with the questions and giving four options for answers if it's a multiple choice questions or two options if it's a true or false type of question. The player then chooses the answer by clicking on it. If the player answer the question correctly, a point is added to their score. At the end of 10 questions, the player is presented with the final score for that round, and two options to either start playing again or to review the questions they gave a wrong answer. If the player chooses to play again, they are directed to the start page. If they choose to review the questions and answers, they are presented with a screen where the answer they chose is highlighted in red, they correct answer is highlighted in green and the other answers are displayed in a neutral yellow.

Specifications and Design

Requirements

Accessibility / UX: The app should be accessible and user friendly. User is guided and can use the app effortlessly.

Functional:

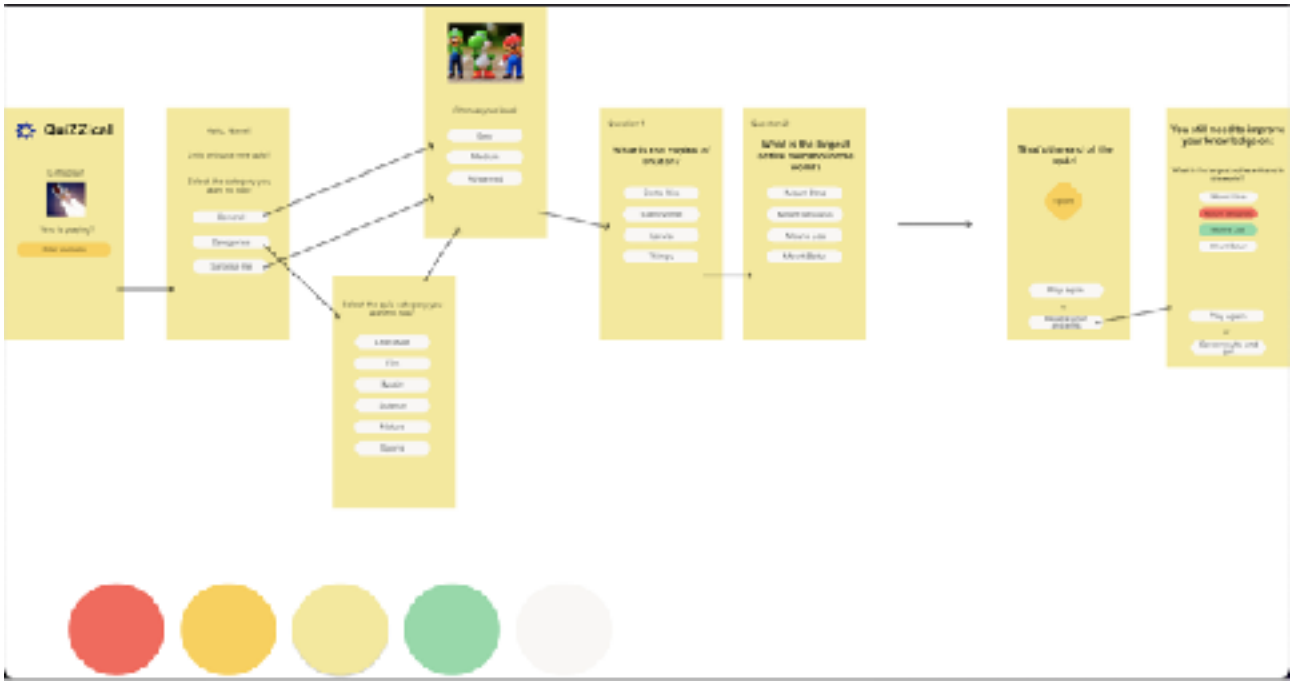
- The user should be able to know what the app is about. We created the name “Let’s get quizzical” to reference the type of name people usually give to their team in a pub quiz.
- The user should be able to personalise their experience by entering their name. The app will then greet them by their name and invite them to choose a category
- The user should be able to pick a category. The first step when starting the play the quiz game is to choose a category.
- The user should be able to pick the difficulty. In order to increase re-playability, the user can choose different levels of difficulty in all categories.
- The user should be able to see how many points they scored at the end of the round. The user is presented with their total score.
- The user should be able to review the questions they got wrong. In order to make the app more educational, it’s important to give the user the chance to learn from their mistakes.
- The user should be able to start a new game. The user is presented with the Play Again button at the end of the round and at the end of the page where they can review the questions. This way the user is always directed where they want to go, and the flow of the game is preserved.

Technical:

- Responsiveness: The web app layout should adapt and look good on desktop, tablets and mobile screens.
- The app uses React and JavaScript.
- The app should be available to users to access from a browser. The app was published using Netlify and can be accessed via internet connection by accessing the URL provided by Netlify: https://64726667e519213b1f996ae4--gentle-lolly-c774bf.netlify.app/category/difficulty/quiz
- The app uses the external API TriviaDB and has a loading feedback for when is fetching data, providing feedback to the user
- Pages:
  - Start Page where the user can see the app logo and enter their name
  - Choose Category Page: The user should click on a category to play
  - Choose Difficulty Page: The user should choose a difficulty level
  - Quiz Page: The questions are displayed in sequence, one at a time. There’s a total of 10 questions
  - End of Quiz Page: The user is presented with their total score and options to play again or review questions answered incorrectly
  - Review Questions Page: The user is presented with the questions which they answered incorrectly.

Design and Architecture

This is the layout with all the components and how they work together:



Implementation and Execution

Development approach and team member roles

Our approach to develop this app was to use a Jira board to have all the tasks that were necessary for the completion of the project laid out with scope and deadlines.

As we all have busy schedules and different times to work on the project we opted for not defining specific roles. Instead, each group member could pick the tasks they wished to work on on the time they had available.

Tools and libraries

- Figma / Diagrams
- Jira
- React
- TriviaDB API
- Lodash
- React-bootstrap
- React-router-dom
- React-testing-library

Implementation Process

Since the beginning we were really careful to keep the scope of the project small and achievable. There were a couple of features we would like to have, but as it was in our nice-to-haves list, we didn't have time to implement them. The first achievement was when the app was working as intended and the second one was when our first test passed. It was also great to see the app live. The biggest challenge was to figure the tests out.

Agile development

We used Jira to divide the tasks. We also used iterative approach by starting with the simplest and smallest tasks and making them more robust as we moved forward. We also used refactoring every time we noticed something could be better, or when necessary to work on future features.

Implementation challenges

The navigation was made using Links and then that proved not the best way to do it. By changing to use the useNavigate hook, it allowed for better testing and solve some issues that could break the app. Also, the way the answers were coded at first had to be refactored in order for them to be styled differently.

Testing and Evaluation

Testing strategy

We left the testing for last and if we could start over we would have started writing the tests as we wrote the code. We didn't take advantage of the testing that could show us the week points of our choices in how to develop the app, as the Link example mentioned before.
The testing was quite challenging and we if had started earlier, we would notice how much time it was required to test.

We did though test everything we did by interacting with the app and correcting what was not working as they happened.

System limitations

- In order to play the game online the user needs an internet connection. Instructions to run the game in the user's machine is given in the README, so they can play in their localhost.

- The user can only play using the questions available in the Trivia API, and cannot add their own questions

- The user can't save their progress, or their score

Conclusion

Our project aimed to create a quiz game web app that is both fun and educational. We successfully achieved our objectives by developing a user-friendly app that provides an interactive and engaging experience.

Throughout the project, we followed a well-structured roadmap, dividing tasks and utilising tools like Jira to track our progress. Our team members took on different tasks based on their availability and skills, ensuring a collaborative approach to development.

While implementing the app, we faced some challenges, but we tackled those challenges through refactoring while adhering to an agile development approach.

We are proud of our accomplishments and believe our app can bring entertainment and learning to a wide range of users.