INTRODUCTION:

The aim of our group project was to design a clean card game application that was engaging and easy to play, utilising what we have learnt in the full stack course.

We have built our gaming app based on the card game 'higher or lower'. Our main objective was to build a game that had a sleek appearance, as we had noticed that oftentimes there are lots of adverts and clutter on gaming sites that distract from the user experience. We also wanted to ensure that navigating our site was simple and logical, whilst looking good and being functional. We aimed to build a card game that is enjoyable, engaging and addictive, a game the user will want to play over and over again.

Report Roadmap

- Background: context of the project and rules of the game
- Specifications and design: requirements of the project and wireframes for our project
- Implementation and execution: our development approach and roles, tools and libraries used, implementation processed and agile working and implementation challenges
- Testing and evaluation: our testing approach, successes and limitations
- Conclusion: our project summary

BACKGROUND:

As our application is a game, the context for the application is mainly based around the rules of the game.

The game rules are simple and easy to follow. First, you hit the start button, and the furthest left card will be revealed to show your starting card. From here, you will then bet whether the next card will indeed be higher or lower than the value of your face up card - using the higher or lower buttons on screen. If lady luck is on your side, and you make the right decision, you get to move to the next round and choose all over again. Your adrenaline is pumping, the cards are coming thick and fast, can you make it to the end of the six cards without a game over, and get that winning title?

Should you make any wrong decisions, or if luck is out to get you; maybe you get a pair, or maybe you even run out of time… all will result in the ending of the game. But not to worry, all you have to do is shuffle the cards and hit that start button again. Ready for round 2? Or 3, or 4?

Don't be fooled though, always remember, King is high, Ace is low, and you will be sure to make it through at least one round unscathed… how many turns will you take?

Figure 1: Our most recent MoSCoW prioritisation chart

Our requirements for this app to work and have a very simple, usable user interface were ensuring there were no distracting features whilst playing, such as removing the shuffle animation whilst the game is in play; and designing the game in such a way that allows for a sleek user experience whilst playing the game and navigating the pages.

There are very simple buttons used to play the game and anything that is not used to play the game, is removed from the page whilst the game is in progress. The buttons that should not be used whilst the game is not in play, also become inactive when you reach game over. These all contribute towards achieving the objectives set out at the start of the project.

SPECIFICATIONS AND DESIGN:

Throughout our project, we utilised the MoSCoW method (M - Must have, S - Should have, C - Could have, W - Won't have) to decide which technical features we were going to have and prioritise. This changed throughout the timeline of the project, based on coding/timing/merging limitations, and the MoSCoW was updated as and when required, see Figure 1 for the most up-to-date MoSCoW chart.

All of our most important features can be seen in figure 1, but the most important technical features were to have our minimum viable product working, which was the game page, and make sure that all components of this game and logic worked as expected. Therefore, our key functional requirements were as follows:

- Higher button
- Lower button
- Start button (appearing and disappearing)
- Cards to flip at the correct times (on clicking start and on clicking each button)
- Game logic to correctly identify when a choice made was either correct or incorrect and react accordingly
- Game logic to start and end the games at the correct times (winning, losing, getting a pair or running out of time)

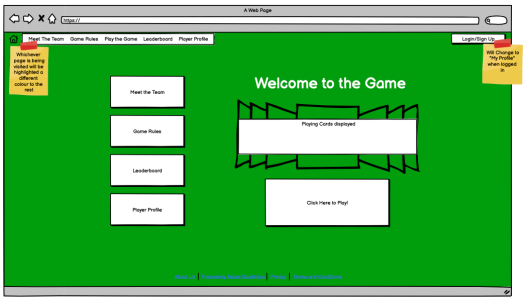

Figure 2: Wireframe for the landing page with links to other pages via a navigation bar and buttons.

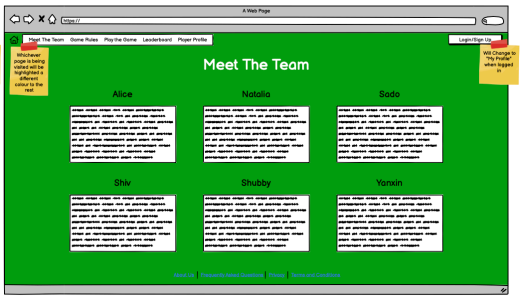

Figure 3: Wireframe for the meet the team page containing hobbies for each team member with links to other pages via a navigation bar.

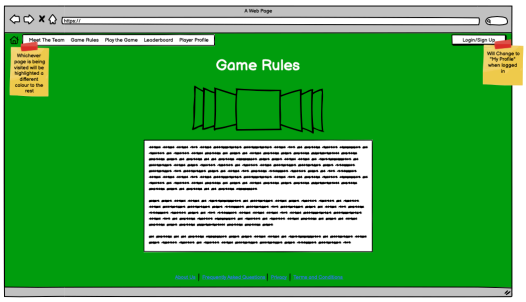

Figure 4: Wireframe for the game rules page stating how to play the game, with links to other pages via a navigation bar.

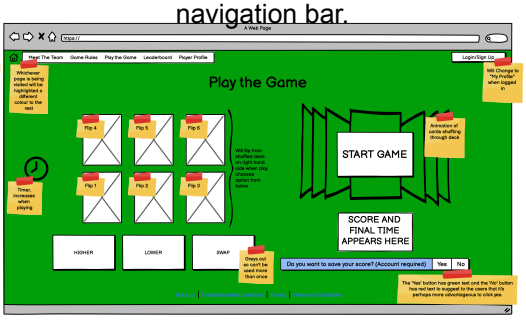

Figure 5: Wireframe for the game page, this includes the cards and the buttons for the game with links to other pages via a navigation bar.

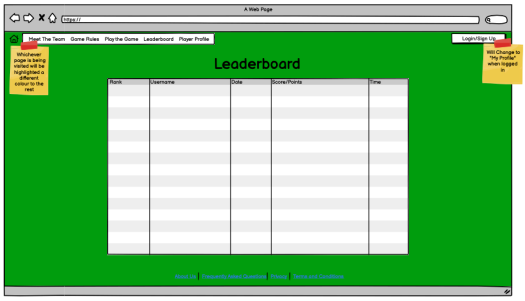

Figure 6: Wireframe for the leaderboard page and titles for the leaderboard as well as links to other pages via a navigation bar.

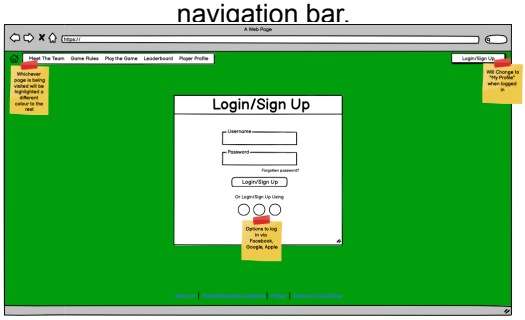

Figure 7: Wireframe for the login page where you can login and sign up to register your game scores as well as links to other pages via a navigation bar.

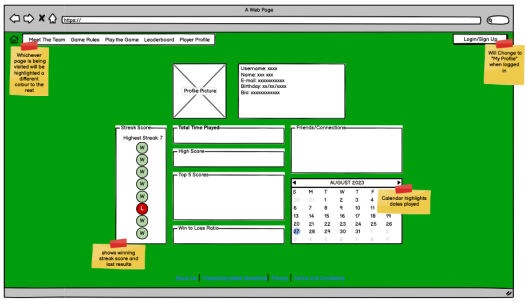

Figure 8: Wireframe for the player profile with links to other pages via a navigation bar.

Our non-technical requirements for this app were to ensure the app was usable and accessible for all. The colour scheme is simplistic, and there are no conflicting colours or readability issues. The navigation is simplistic, and uses icons so it can be understood by many people, and uses the correct routing to each page.

We used wireframes to design our initial project sitemap, these can be seen in Figures 2-8 (as well as in our GitHub repository for larger versions). As the project developed, we did not include all pages seen in the wireframes below, just those from Figures 2 and 4 (combined to make the home/landing page), 3, 5 and 6.

IMPLEMENTATION AND EXECUTION:

Development Approach and Roles

When starting up the project, we set up Miro to brainstorm ideas and manage the project throughout its lifecycle. One of the first things we did was state our areas of strength, growth, interest and working styles. We used these to establish our roles and responsibilities (Figure 9).

We originally split our roles into back-end and front-end, making sure everyone had an area they were interested in. This was of great benefit because it meant we were interested in the tasks we were assigned. At several stages throughout the project, we came together to decide a list of tasks that needed completing, and each took on a task to work on until the next meeting.

We quickly learned that there were going to be a lot of changes to the original code in different branches. We then made the decision to have another team member review and approve any pull requests to ensure we could maximise the chance of catching any errors.

When issues arose throughout the project, these were discussed as a group and resolved through unanimous decision.

To resolve issues, we discussed them as a group and it was resolved through unanimous decision. This was useful especially when having to resolve code conflicts. We handled all code conflicts as a group, using different branches to check each other's merges and explain through our code to ensure nothing got mixed up.

Overview of Project Roles (Figure 9)

Natalia	**Siobhan**	**Alice**
Wireframes, game page design and logic, navigation bar, styling, game rules, project doc setup	Miro setup, wireframes, agile, team page design and functionality, project doc, homepage	ReadMe, card flipping, footer, styling, timer, leaderboard
Sado	**Yanxin**	**Olasubomi**
API cards, card deck, flipping cards, game page game and button logic, project doc	GitHub setup, Bulma, navigation bar, React testing, project doc	Leaderboard and linking with game page, team page, timer, project doc

Figure 9: overview of project roles

Tools and Libraries

- React
- JS
- CSS
- HTML elements/JSX
- API for card game to fetch and shuffle the cards and show images of the front and back
- Bulma and FortAwesome for styling
- Media and Key frames
- NPM packages
- Routing
- React testing Library
- Jest

Implementation Process (decisions, challenges and achievements) including Agile Development and Scrum Values

We implemented some agile methodologies from the outset and throughout our project using the 5 Scrum values (commitment, courage, focus, openness, and respect) from our first meeting without even realising it.

We were **open and honest** with each other about where our skills lie and our availability for the time we had to complete the project, this meant that we were all aware of what each team member could commit to. Being this way from the start made it a lot easier to work together and have **courage** to speak up if we thought something could be done differently and take accountability for tasks.

Throughout the project, even when the scope and the plan for the minimum viable product changed, the whole group was **committed** to achieve the MVP, and when required all **focused** on the necessary tasks to produce the final MVP.

We had **respect** for one and other during sessions by ensuring everyone was involved and everyone's opinions/inputs were taken into account. We always made a decision as a collective so that we weren't disregarding anyones ideas and or work.

Our initial idea was great, yet optimistic. Once we started to build the application, we realised how great a task it would be to complete it all to a good standard. We quickly decided to change the scope of our minimum viable product and what we would need to do to complete this in time to meet our deadline. Having discussed this as a group, we decided that our MVP would include the Meet the Team page (as we had created this already) and a working Game page. This meant that we ended up working in an agile way, adapting to change by agreeing on a new minimum viable product. With this in mind, without realising, we were working with an iterative approach by taking little tasks away to work on and adding them to the project bit by bit and building on the base of the app.

Throughout the project we followed a similar format to 'sprints,' as we were each taking away a task to work on and then regrouping two to three days later. When regrouped, we would do code reviews to make sure we were all comfortable with each other's work, whilst also raising any impediments we were coming across and getting support from the group.

We decided together as a group which features we needed to work on, and assigned 1-2 people per task depending on the scope of the task. We first prioritised the game logic so that we had a working game, and then decided to work on the routing between the pages we had made so far. Whilst also accounting for and handling any styling issues and bugs that were identified.

As expected, we had some challenges throughout the project, the main one being time. Once we'd decided on the revised project scope, we had 3 weeks to work on the project. This being a short period of time as well as differing availability provided a real challenge to our team - this being the main reason for the scope of our MVP changing.

Another key challenge we faced was merging some of the code that had been created. We had a navigation bar created by one team member, but every time it was merged to the most up-to-date branch, the styling for each page would be overwritten and everything would change. We couldn't figure out exactly what it was from that navigation bar that was causing the issues. We therefore decided that we would deprioritise it until we completed everything else, and then try to merge it at the end. If it was then still not merging correctly we would decide, as a collective, on how we wanted to approach it. Whether that be to have the NavBar that was created and sacrifice the styling of the pages or sacrifice the NavBar and use a simplified version so that the styling stayed as it was. In the end we decided on the latter, and our app and navigation now works nicely, with all styling intact.

During one of our user tests, it was identified that there was an issue with our main code base, meaning that some card values were incorrect, mainly the Jack, Queen, King and Ace cards. This would go on to affect our MVP if not fixed. Lots of user testing was used to play the game, output values to the console log, and decipher and fix the error using code. This however, was proving rather difficult, and another approach had to be taken to look through the API itself. Having looked through the API it was realised that the Jack, Queen and King were not assigned values in the API and these were then rectified in the code to assign values to all cards with the correct values, i.e. using 11 instead of 'JACK'.

<u>Implementation challenges</u>

As discussed in previous sections, we faced challenges such as time restraints, availability and merging of code.

On top of this, at the start we anticipated a back and front end split would work well, however we ended up needing to change this to a more task-focused approach. We focused on smaller tasks instead of splitting the back and front end and it meant that work was completed more quickly and didn't have as many conflicts.

Another struggle we had was having limited working knowledge of javascript and the React library. This meant we were spending longer than necessary on components, having to do lots of research to complete the tasks.

Overall, we handled the challenges we had as a group to come to a decision on how we would manage them going forwards.

TESTING AND EVALUATION:

Our main testing approach included the following strategies:

User testing: we user-tested each of our functions when creating them using React. User testing provided invaluable insight into the game and its mechanics, and without it we would not have our MVP.

For testing the components, a combination of unit testing and integration testing was employed. The goal of this strategy was to thoroughly test the functionality of the component and ensure its interaction with other components in the system. This was completed using the React Testing Library and Jest.

- Unit Testing: This involved creating test cases for various scenarios, such as flipping cards, checking button visibility, and counting the number of unflipped cards.
- Integration Testing: The interaction between different components was tested to ensure they work together seamlessly. The component's behaviour was tested when interacting with the different elements to ensure a smooth flow of events.

Test Cases:

1. Footer Test: Validates footer text rendering.
2. Game Test: Validates that the start game button flips a card, verifying its presence, visibility, and expected behaviour.
3. Leaderboard Test: Checks submission and score buttons and input fields. Submissions appearing on screen post-score submission are validated.
4. Team Test: Validates name card flip behaviour on the team page.

<u>System Limitations and Evaluation</u>

Though we have completed initial unit, integration and user testing which sufficiently tested our app, due to time constraints more thorough tests were not completed. Furthermore, there are some tests we have started but could not complete such as with the app file. Further and complete testing of the game logic would have significantly cut down the time spent user testing the game and would ensure edge cases were accounted for.

CONCLUSION:

The aim of our group project was to design a clean card game application that was engaging and easy to play, utilising what we have learnt in the full stack course. We met our objectives, and made our app with a working Hi-Lo card game. All components of the game app work, as well as the meet the team and the landing page. We did manage to make a demo leaderboard, but this is one of the features we would look to improve in the future.

In the future, if we had more time we would focus more on the logistics of the game. We would look to implement a lobby, where you can choose which difficulty level you want to play at. We would also look to improve the leaderboard to record final scores, which would mean improving the game logic to include the initial scoring we set out to do. We would also like to add functionality to play with other users, whether that be people you know, or random users.

Due to time constraints adaptations were made to the initial design, but we still managed to implement our minimum viable product. Having said this, throughout the project, we all learned a lot about the React framework, as well as JavaScript, CSS and HTML. Fundamental research was required by each team member in order to successfully facilitate creation of the game. This project allowed us to put our theory into practice by bridging the gap between the two.

In terms of working in a team, we managed conflict well, and meeting regularly allowed us all to maintain accountability for the project and solve any conflicts and issues together. By the end of the project, we became a cohesive team, working fluently together to finalise the project.