GROUP PROJECT - FULLSTACK

Fun 4 All

A centralised hub of theme park accessibility information for individuals with additional needs

INTRODUCTION:

Group 2: Sarah Pascoe, Bethany Hopla, Warda Egal, Bronwen Grainger, Zuzanna Wnekowska, Majella O'Mahony

• Roadmap of the report

• Aims and objectives of the project

Our project aims to create a centralised hub of accessibility information for individuals with additional needs, enhancing their theme park experience. The website currently provides detailed information for three parks: Legoland, Thorpe Park, and Chessington World of Adventures. This information is divided into two main categories: a list of available accessibility features at each park and the wait times for the rides. Additionally, users have the option to favourite rides based on the provided information.

BACKGROUND:

Our website is designed to provide essential theme park ride information to the user. To use the website, the user must select one of the three available parks and click the "Explore" button to view the wait times. By pressing the "Accessibility" button on the home page, the user can access a list of available accessibility features. We have also included a navigation bar, allowing the user to return to the home page at any time to explore other theme parks if they wish. The final feature is a "Favourites" button. This allows the user to save rides for later access and view these rides by clicking on the Favourites label on the nav bar.

SPECIFICATIONS AND DESIGN:

Our website's development was guided by comprehensive requirements both technical and non-technical. This ensured that both functional and user experience aspects were effectively addressed.

• Requirements technical and non-technical

1. Technical-
    - Frontend Framework- React.js for an interactive user interface
    - Backend framework- Express.js for handling server-side operations and API requests
    - API Integration- Queue Times API to fetch real-time wait times for theme park rides
    - Testing Libraries- Jest and React Testing Library for unit testing
    - Version Control- Git for managing code versions and collaboration

2. Non-technical-
    - User Experience and Usability- implement a user-friendly interface with appealing visual elements like an engaging colour scheme and relevant imagery
    - Documentation and Support- README file with clear instructions on how to run our website

• Design and architecture

Design-

1. Wireframing (Our figma design-(https://www.figma.com/design/hFIFpMzMrt3tGaZAT0grhy/MUI-(Material-Design-Component-Figma-Library-For-React)-(Community)?node-id=629-234&t=TJhbCVu6uVBw6UIm-0) - we used Figma, a wireframing and user interface design tool to visualise our design before starting the coding process. This visualisation of the web pages enabled us to address how the user would move between pages and find the data they need to plan their day at the theme park as well as suggest a visual layout that was clear and easy to navigate. We used some pre-made components from the MUI library because they follow good design principles and would be easy to implement on our website. This helped us plan a UI that would be close to the final product. We went through an iterative review

process looking at the user journey as a group, consequently creating a more user-centric design.

2. Visual design- for the visual design of our website we considered factors such as colour schemes, typography and imagery. We picked a harmonious colour scheme that resonated with our website identity and user experience goals. We were inspired by a colour palette from Color Hunt (https://colorhunt.co/palette/3c486bf0f0f0f9d949f45050) . Furthermore, we utilised imagery of the parks to create a website that is engaging for the user.

Architecture-

1. Component-based- Our website adheres to a component-based architecture, where elements are designed to be reusable, promoting better maintenance and scalability. By breaking down the interface into modular components, we ensure consistency across the application and streamline the development process.
2. Frontend framework- we used react for our frontend framework. This framework helped build a dynamic and interactive user interface. Moreover, it facilitated a component-based structure.
3. Backend framework- we used Express.js to create a server that listens for incoming requests and handles them appropriately.
4. API- Fun 4 All interacts with the external API *Queue Times* (https://queue-times.com/pages/api) . This API retrieves real-time data, a crucial source of information, enabling us to provide users with up-to-date data.

IMPLEMENTATION AND EXECUTION:

• Development approach and team member roles

Throughout the project we maintained a highly collaborative approach, holding meetings often and discussing/reviewing work done. The roles for each member were:

1. Sarah- API component (later added to ride.js) and resolving CORS issue/backend server , collating available accessibility information for Thorpe Park, Legoland, Chessington, Alton Towers, Paultons Park, two SQL databases for accessibility information for all parks (not used in final project), project document, presentation slides
2. Majella- creation of reusable components and their functionality (hooks, props)for Theme Park Accessibility Button, Accessibility DropDown Menu, Access Page display options, Reusable Link: Home Page to Accessibility Page. Reusable Unit Testing for Accessibility Button.
3. Bronwen- Linking the rides page via the Explore rides button and DropDown on the home page. Formatted this with Rides.js, RidesList.js and RideTable.js to be able to pull through the API information in a table format. Also contributed to the project document.
4. Warda - Utilised Bootstrap and other in-built CSS functionality to create a visually appealing app and style all of the different components across the pages. Ensured that styling remained consistent across browsers and devices and that it was also visually accessible to users. Contributed to the project document and various brainstorming sessions with the team, flagging and fixing bugs in our app where possible.
5. Zuzanna- Frontend focus,- created the Figma file, Rides.js, RidesList.js and RideTable.js and About.js, creating dummy components before connected to API, adding CSS grid layouts, editing the routing and homepage. Used inspect to ensure that styling remained consistent across viewports.

6. Bethany- Homepage & image carousel; routing paths on homepage; re-reusable nav component; favourites button (including save to local storage functionality); favourite rides table component; populating rides table with API data as well as favourites button. Also assisted with API CORS issues by creating an express server and backend folder and linked API component files. Additionally, created unit tests for components, some using jest to mock required functionalities.

• Tools and libraries

1. React
2. Bootstrap
3. Queue Times API
4. React router dom
5. CSS
6. React testing library
7. Jest
8. Material-UI (MUI)
9. Git
10. Figma
11. Express.js

• Implementation process (achievements, challenges, decision to change something)

One change we decided upon was not to include a database for accessibility information. Instead, we opted to list this information on the respective pages. We made this decision after consulting with our instructor, as implementing a database would be overly time-consuming and could compromise other aspects of our project.

We initially struggled finding an appropriate API to match our project idea. As we originally pitched a similar idea of wait times but instead for airport security. The APIs we found for this often were only accessible for businesses or did not contain all the information we needed. It was after this that we found the theme park API and so adapted our project idea to this.

Although this API worked better for us it did present problems when implementing as it has a CORS policy in place which didn't allow us to make the request from local host:3000 on the browser. To solve this we had to reroute the request through our own server which we built through express.

• Agile development (did your team use any agile elements like iterative approach, refactoring, code reviews, etc)

We made sure to have regular Zoom meetings where we set deadlines for the next steps of the project. Then at the next meeting we looked over the work and looked for improvements/ adaptations we could work on until the next meeting.

For some of the more challenging sections of code we paired up to review each other's work, looking for any improvements and also to see if there is anything we could support the other on.

• Implementation challenges

1. Workload management

- Our first challenges came in the form of conceptualising how best to divide tasks and combine the work of others in order to make sure our work fit together.

2. Version Control Conflicts

- As we were all working on the same codebase it can lead to conflicts in version control, especially when we were working on the same files when our roles overlapped eg linking the home page to a page someone else is working on.
- To combat this we each worked on a separate feature branch and regularly pulled from the main branch. We also tried to resolve conflicts early and used pull requests for code review and integration to the main branch.

3. Code Consistency and Standards

- As we all worked on separate parts maintaining consistent coding standards and styles across the team needed to be discussed
- We established a colour scheme at the start so everything looked standardised. We then discussed our strength and those that preferred CSS conducted code reviews to completely standardise everything at the end of the project.

4. Communication and Coordination

- With 6 members in the team and everyone having different schedules we needed to ensure that all team members were on the same page regarding project requirements and progress.
- We regularly held Zoom meetings (once or twice a week) to discuss progress, blockers, and next steps. We also recorded the sessions to look back on in case someone couldn't attend and then wrote up a summary on slack of what was discussed along with everyone's responsibilities for the week.
- As many of us also had full time jobs we found it helpful communicating asynchronously through Slack when we could to update everyone on our progress.

5. Managing Dependencies

- We needed to keep track of third-party libraries to ensure that all team members were using the same versions.
- We did this via a commit of package-lock.json to the repository to regularly update dependencies.

TESTING AND EVALUATION:

• Testing strategy, functional and user testing

Our testing strategy is centred around ensuring each component of our web application was thoroughly tested. We adopted the following approach:

- **Separation of Testing:** Each team member was responsible for creating tests for the components they developed. This approach ensures a deep understanding of each component's expected behaviour, leading to more effective and targeted tests. By having the developers who wrote the components also write the tests, we ensured that the tests were comprehensive and addressed the specific functionalities of each component.
- **Unit Testing:** We primarily utilised unit testing to verify the functionality of individual components in isolation. Unit testing helps in identifying and fixing issues at the component level, ensuring that each part of the application works correctly on its own. This method is essential for maintaining the robustness and reliability of the application, as it allows us to catch bugs early in the development process.
- **Testing Library:** React Testing Library and Jest
    - React Testing Library- enabled us to write tests that closely resemble how users interact with our website. This library is particularly valuable because it focuses on testing the application from the user's perspective, ensuring that the user interface behaves as expected. By simulating user interactions, we can identify and resolve issues that may affect the user experience.
    - Jest- the mocking capabilities of Jest allowed us to isolate components during unit tests as well as supporting asynchronous testing. This meant we could test components that depend on data fetching. In the 'ThemeParks' test we use 'waitFor' to ensure the mocked 'RideTable' is rendered after the data is fetched.

• System limitations

Our website currently features only three parks. In the future, we aim to expand its scope to include a wider selection of theme parks across the UK. To support this expansion, we could implement a comprehensive database. Additionally, we plan to enhance the user interface to accommodate more parks without compromising the user experience. For instance, we could introduce a filter system that enables users to efficiently find and explore parks based on their preferences. This filter system could likewise be applied to the favourites page, allowing users to better navigate the page through filtering by park.

Another limitation of our current website is the use of local storage for the favourites page. As a result, the parks favorited by users are only stored until the cache is cleared, which can lead to the loss of saved data. To address this issue, we could transition to server-side storage or cloud-based solutions, ensuring that users' favourite parks are securely saved and accessible across different devices and sessions. This would provide a more reliable and seamless experience for our users, allowing them to maintain their favourites regardless of browser or device changes.

CONCLUSION

Our project, Fun 4 All, aims to enhance the theme park experience for individuals with additional needs by providing centralised accessibility information. By offering a detailed list of available accessibility features and real-time wait times for rides, we strive to empower users to enjoy their time at the parks.

Throughout the development process, we tailored our work to both technical and non-technical requirements to ensure a user-centric and functional web application. Our design and architecture decisions, such as using component-based architecture, allowed us to build a dynamic and maintainable website. Despite facing several challenges, from conceptualising task division to managing version control conflicts, we always worked through these as a team, maintaining a high level of communication. Our challenges have helped us grow as developers, enhancing not only our React skills but also our understanding of project building.

Our testing strategy ensured that each component was well-tested, with a focus on unit testing and user interaction through the use of the React Testing Library framework. This approach helped us identify and resolve issues at the component level.

In conclusion, Fun 4 All is a testament to our desire to make theme parks more inclusive and accessible. We are proud of what we have achieved and look forward to expanding and refining the platform to better serve users in the future