4CodeGirls present

Winter Wonderland Map

(Project Documentation)

1. INTRODUCTION

Our project, initially conceived as a fictional winter wonderland due to shared love for the festive season, transitioned to a more feasible scope, focusing on the vibrant city of London. The primary aim was to create an interactive Winter Wonder Map that provides users with a delightful experience to find and book events during the festive season. The roadmap for the report includes an outline of the project's objectives and a clear overview of the sections to be covered.

2. BACKGROUND

The Winter Wonder Map is a website that offers users a seamless one-stop experience with exclusive access to finding and booking curated events during the festive season, in London (appendix 1).

The website functions as follows:

a) Upon launching, users are directed to enter their name and a secret password (appendix 2);
b) Once logged in, the website greets the user with a button to explore its features (appendix 2);
c) When a user clicks on the explore button, it navigates to an interactive map where they can select event categories to display events happening around the city on the Winter Wonder Map (appendix 3);
d) Users are then able to explore the map, read about the events (including illustrative images) and choose to book specific winter activities (appendix 3);
e) The flow continues to a booking page where users enter their details, culminating in a confirmation page (appendix 4).

The decision to shift from a fictional setting to London reflects our pragmatic approach to project management. The time saved on designing our own map allowed us to reallocate resources to more vital tasks so we could complete the project within the given time period.

We made further adjustments to the original design (fig. 1) during development by introducing additional screens, specifically the launch screen featuring a secret password.

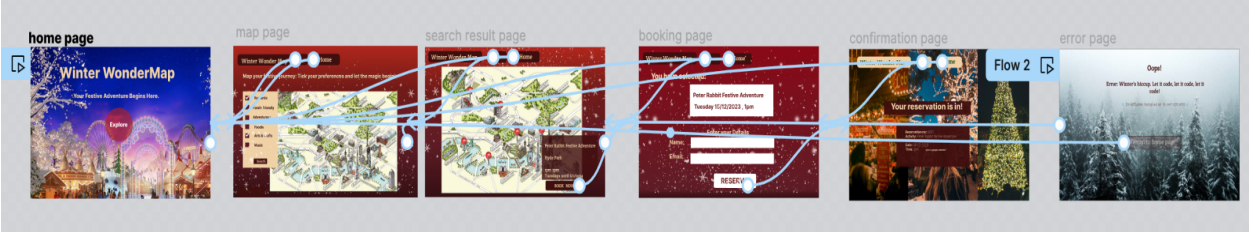


Fig. 1 - Initial conceptualisation of Wonderland Map design prior to project development.

3. SPECIFICATIONS AND DESIGN

The project's technical and non-technical requirements were carefully considered, emphasising a seamless user experience. Our design and architecture choices aim to create an intuitive and visually appealing platform for users to interact with when booking events for the festive period. We wanted the design of our application to match the chosen theme - festive fun. We added a custom festive font, royalty free wintery images, and colours from our holiday colour palette (appendix 5). The use of a colour palette ensured our design was consistent throughout the application. The color codes can be consulted at the *dependencies.txt* file.

As for the technical specifications, we designed our application to function as a booking site with a login password to access the exclusive Winter Wonder Map events to create a smooth and authentic user experience (appendix 2).

To produce trouble-free interactions for users, our Design and Architecture can be seen in the following diagrams:

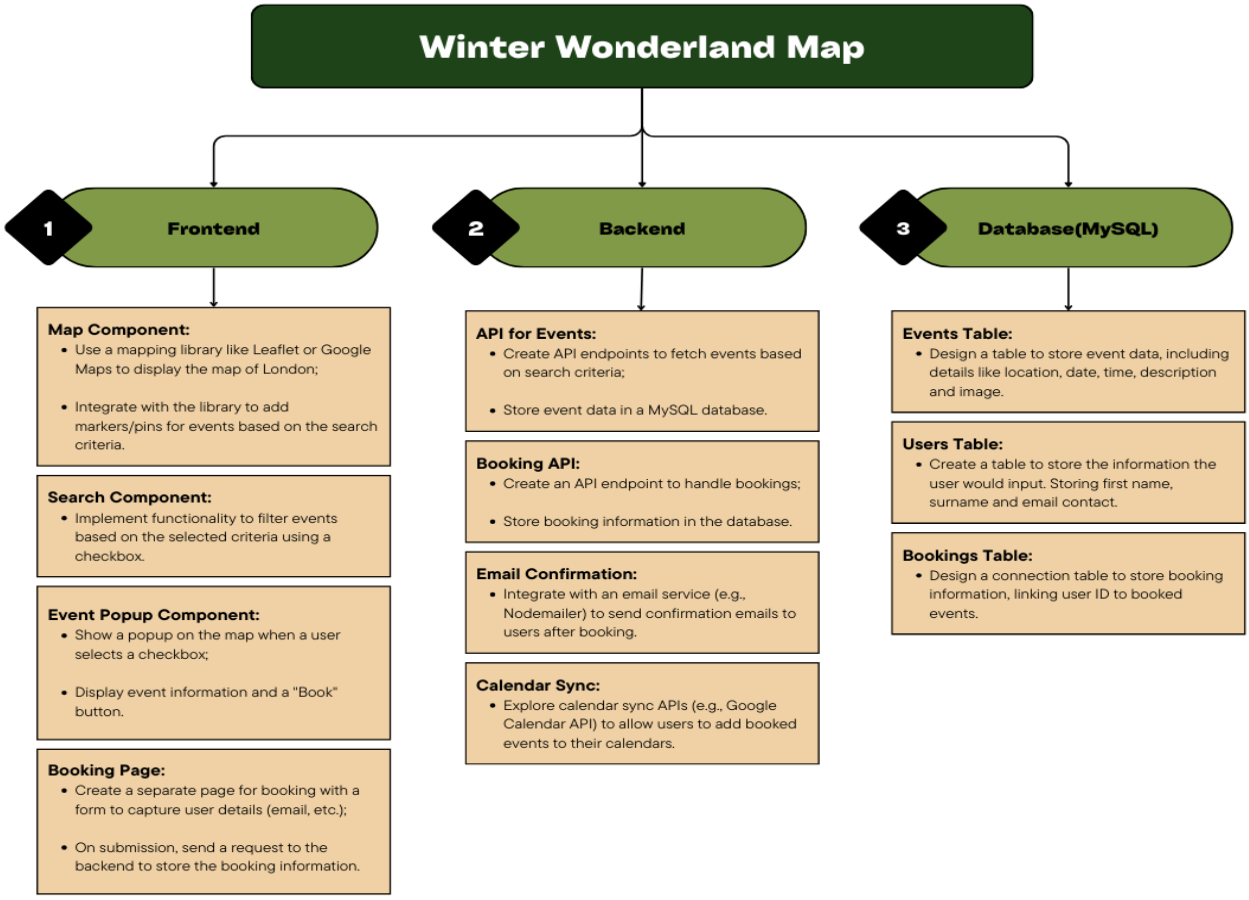


Fig. 2 - Design and Architecture diagram of Wonderland Map Website.

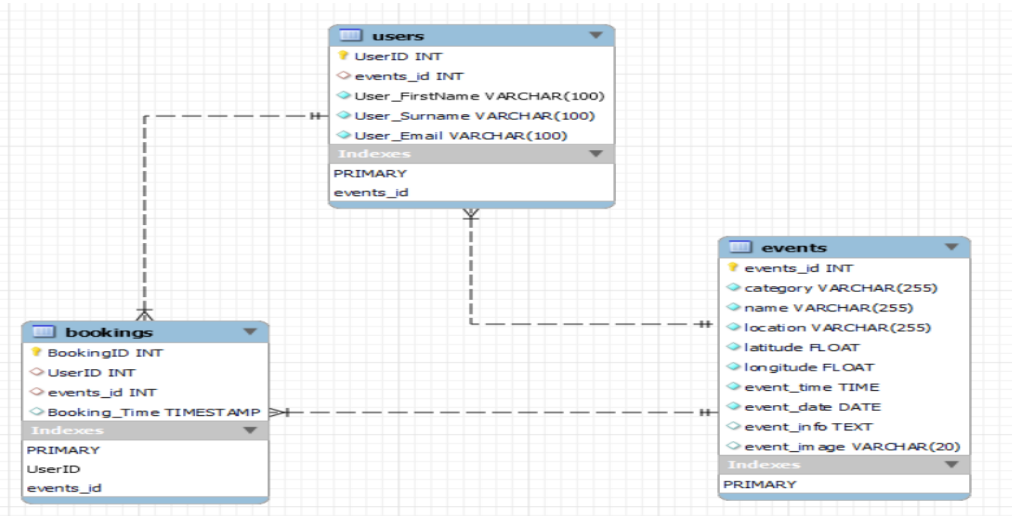


Fig. 3 - Database scheme of Wonderland Map Website.

4. IMPLEMENTATION AND EXECUTION

Our development approach embraced collaboration, with each team member assuming specific roles. Tools and libraries such as Node, VS Code, React, Python, JS, CSS, and Redux were instrumental in the project's implementation, along with Trello for task management. The iterative process involved weekly Zoom meetings to discuss achievements, challenges, and progress. Agile elements, including refactoring, were employed to enhance code quality. For instance, the original backend file was refactored into separate database and app.js files to keep the code more organized and readable. To aid the project management of the task, team members were assigned specific tasks, ensuring progress was properly coordinated and monitored (fig. 4).

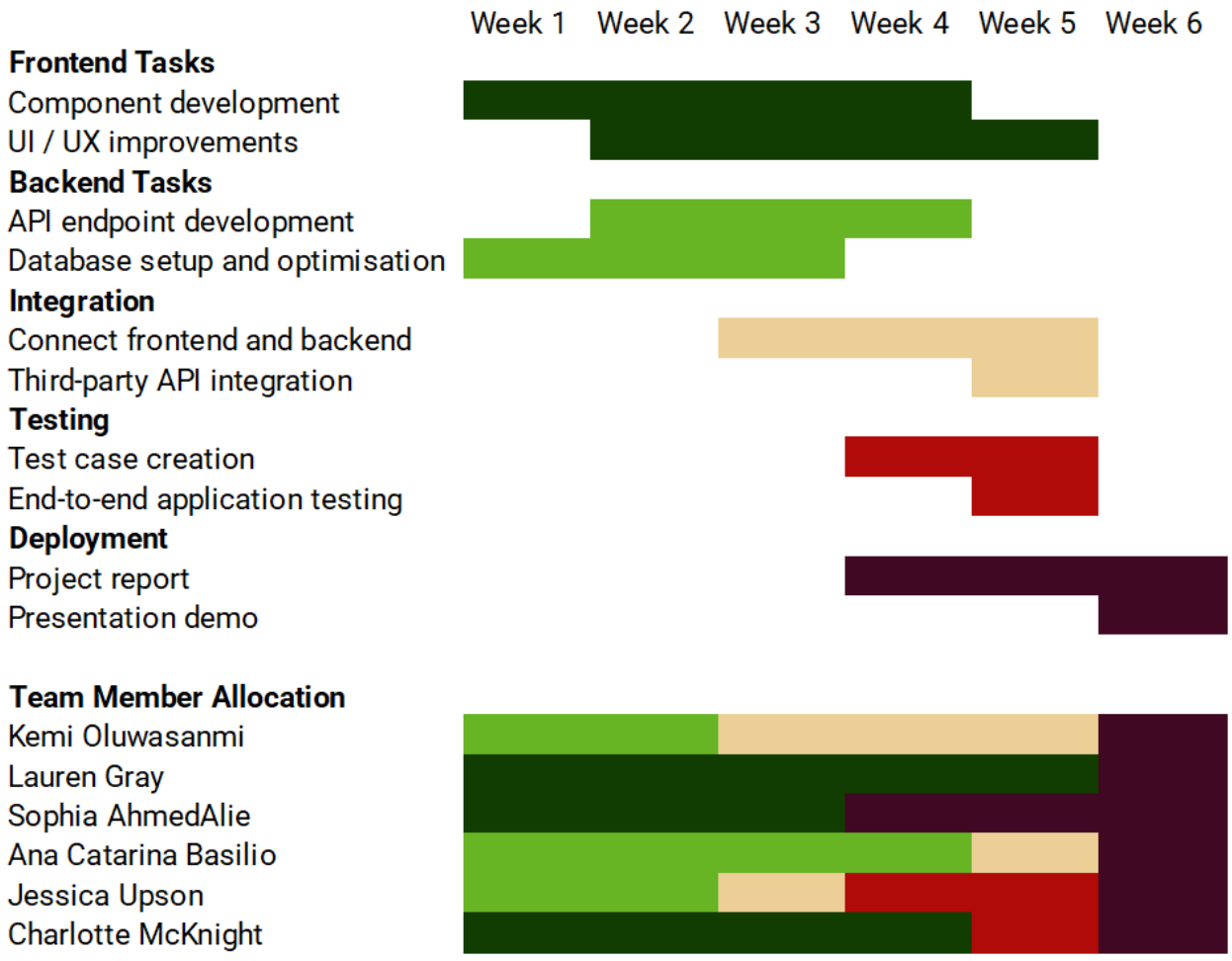


Figure 4 - Breakdown of project tasks and resource allocation.

Overcoming challenges required team effort. Here's how we used and coordinated each tool/section:

Github

We used GitHub as our main version control system, supplemented with the Git Bash terminal integrated into VS Code. We designated our master branch as the 'production-ready' branch and used an additional branch for development. New features would be developed on separate branches before being merged into the development branch. Pull requests required reviewing by a team member prior to merging and we used detailed yet concise commit messages for every commit to allow appropriate tracking of application development.

API

The front-end was developed using React, incorporating Hooks such as useEffect for fetching the API at the first render and useState for state management. Despite initial intentions to use Python for the backend, complications in fetching API endpoints led to a swift adaptation to Node.js. This shift required additional learning of Node.js while ensuring the production of clean, efficient code.

Leaflet.js

Implementing the Leaflet.js library to create an interactive map posed a significant challenge. It necessitated thoroughly reading documentation for understanding, and overcoming hurdles in displaying realistically located pins on the map. Notably, displaying event locations on the map proved intricate. To resolve this, the backend team created API endpoints to fetch category details from the database and display events on the map based on the event geolocation. Successfully implementing this feature marked a pivotal achievement, providing the foundational structure for the entire project. Learning additional React functions like map and filter, along with spread operators, became essential in achieving this milestone.

Database

Originally planning to use MySQL with Python, a transition to Node.js necessitated quick learning of connecting MySQL with Node.js and writing SELECT queries.

The SQL server played a vital role in creating API endpoints. One complex endpoint created involved a POST request wherein user-submitted data triggered a stored procedure inserting form data into the USERS and Bookings tables, sending an updated booking number to the front end for the confirmation page. This seemingly straightforward functionality turned out to be time-consuming to set up.

iCalender feature

We successfully integrated a dynamic QR code generator into the frontend of this project, allowing users to effortlessly save event details. Upon scanning the QR code, the information seamlessly transforms into the widely supported iCalendar format. This innovative feature caters specifically to iPhone users, enabling them to conveniently add event details directly to their calendars with just a scan.

REDUX

State management was implemented using Redux, with a reducer.js and an action.js file. The useDispatch function dispatched the state, and the useSelector fetched the action. Upon a user's initial login, Redux stored the user's name, displaying a customised greeting. The Redux store also retained selected event details and user information, all of which were accessible to use in the booking page and the confirmation page. The implementation of Redux facilitated seamless data access within the react components.

React Router

React Router was chosen for navigation, providing a structured approach over conditional rendering. This decision streamlined the management of different pages within the application, each accessible through individual navigation paths. For instance, localhost:3000/ displays the home page, while localhost:3000/map showcases the map component.

MailJet API

We began integrating a third-party API, MailJet, to allow us to send booking confirmation emails. Whilst this successfully ran within the independent backend-development environment, CORS and proxy restrictions hindered our efforts to fully integrate this into the project. This would be a key aspect of our focus in any future Winter Wonderland application development.

5. TESTING AND EVALUATION

The testing strategy included debugging techniques, manual and unit testing to ensure the Winter Wonder Map's robustness and usability. We employed Jest and React testing library for testing react components. The tests were ordered using the pattern arrange, act and assert. The tests checked that all appropriate elements were rendered and event handlers worked correctly.

As we used different frameworks we ran our application via localhost and manually tested the application to ensure everything was working as intended. We added console logs to record the journey of our program, it also enabled us to see if there was a viable connection between the front and back-end. User input was considered as a possible error case, so input validation was implemented using RegEx, alert prompts and helpful messages that guided the user to correct input.

Our project employed various testing methods to ensure comprehensive assessment. Component and unit testing were conducted to evaluate individual components independently, while system performance underwent thorough evaluation. Functional testing aimed to confirm the project's adherence to technical and functional specifications. Finally, end-to-end testing was employed to ensure seamless operation across the entire platform.

As part of any future rollout we would utilise user testing to gather feedback, seeking insights into potential system bugs and assessing user-friendliness - an integral goal of the project.

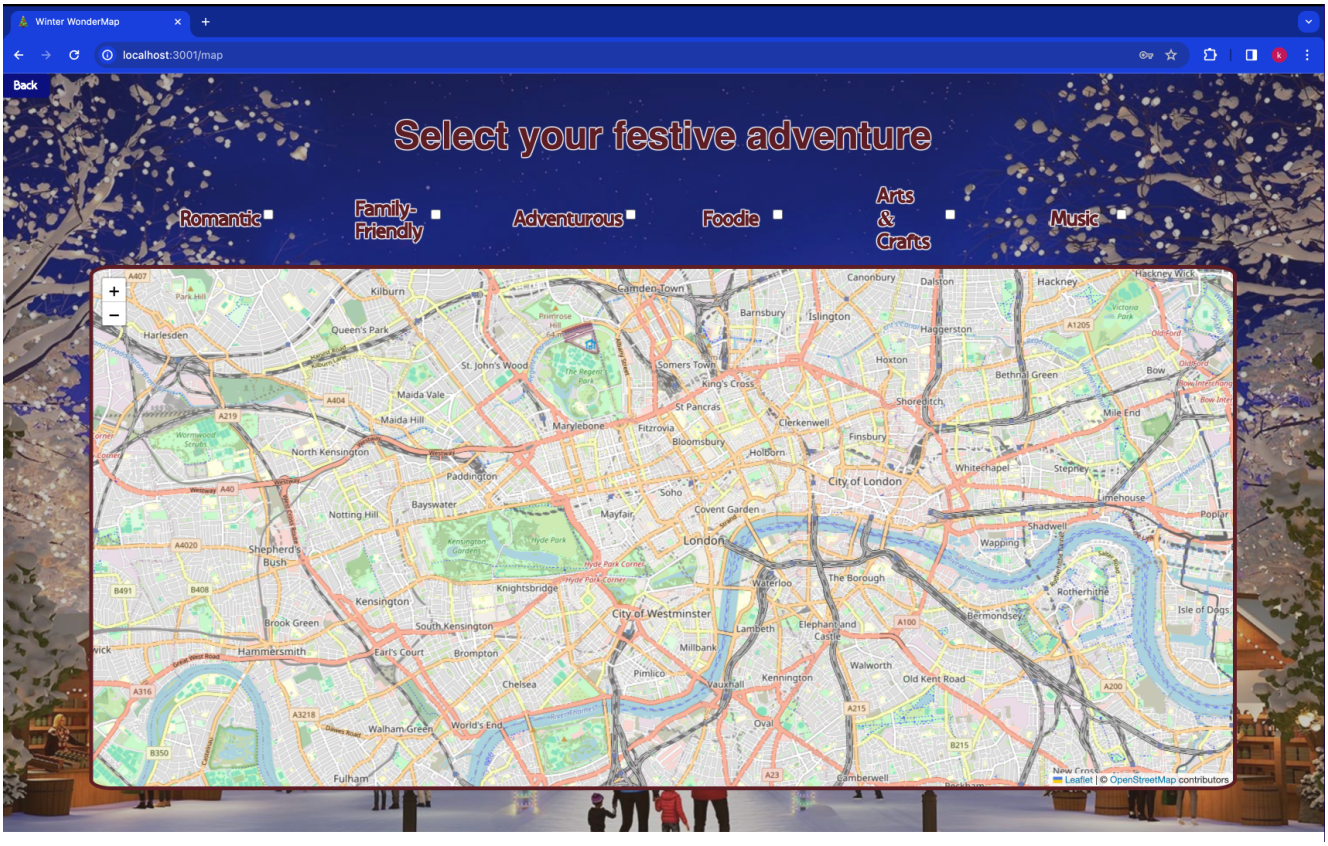
6. CONCLUSION

"The goal of our web application is to provide users with an interactive map, featuring festive events with mocked data. Users will input their information (name and email) to access the platform and use search criteria, such as the type of activity, to filter and explore events. Upon clicking on a specific event on the map, detailed information is presented, offering users the option to book the event. After a successful booking, users receive an email confirmation with the additional option to sync it to their calendar." **In 4CodeGirls System design document.**

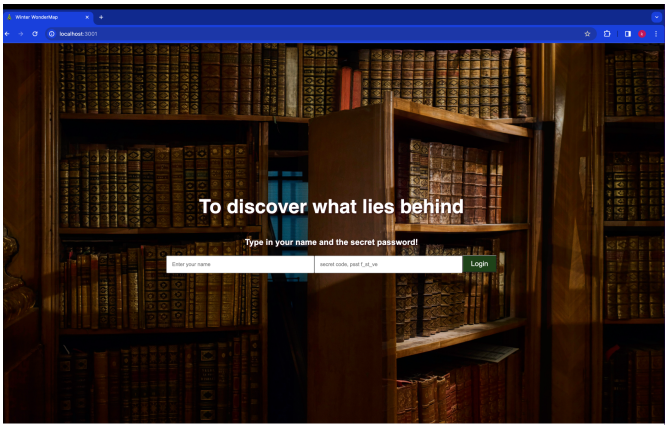
The Winter Wonder Map project successfully navigated from conceptualisation to implementation, overcoming challenges and delivering a user-friendly, festive experience. We managed to meet the objectives we initially set out for our project, with a few minor adjustments, allowing us to create a more feasible and user-friendly webpage. For example, implementing the option for users to sync the booking with their calendar via an email confirmation was adapted to syncing the booking with their calendar via a QR code. Time and system limitations caused us to adapt features like this from our initial plan to keep the application functioning smoothly as desired.

We discussed a range of ambitious features during meetings, and given a larger time frame, we would have continued to expand the functionality of our application and its features. Although time and system limitations led to adaptations, the combination of collaborative development, agile practices, and strategic testing contributed to the project's overall success. The report reflects our journey, detailing the intricacies of the Winter Wonder Map and capturing the essence of a winter celebration in London.

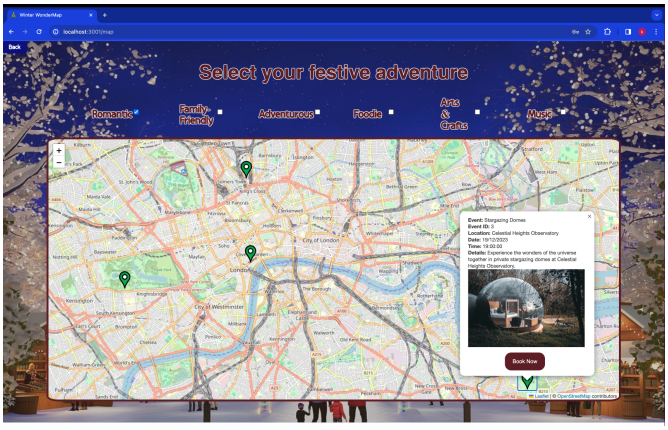
APPENDIX

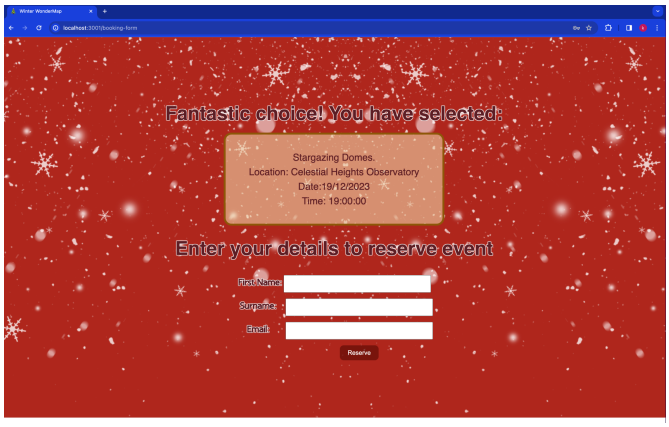


Apendix 1 - Map displayed in the Winter Wonderland Map website.

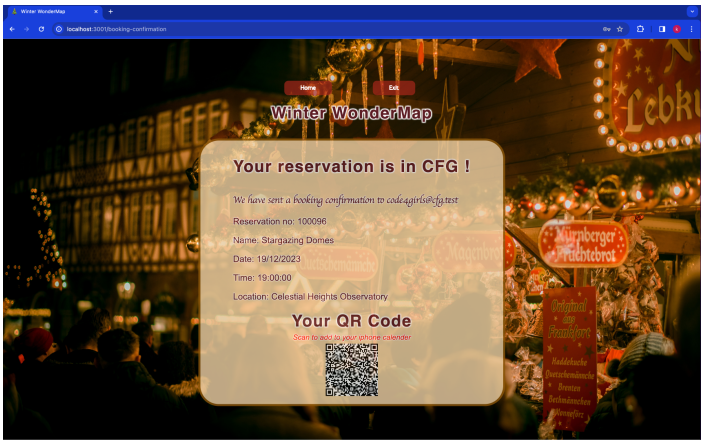


Apendix 2 - Lauching and welcome screens of the Winter Wonderland Map website.





Apendix 3 - Map with an event selected and Booking screen of the Winter Wonderland Map website.



Apendix 4 - Confirmation screen of the Winter Wonderland Map website.

Apendix 5 - Christmas colour palette selected and used on our Wonderland Map Website.