Ollscoil Teicneolaíochta an Atlantaigh

Atlantic Technological University

Year 4

Ignats Lozko

Bachelor of Engineering (Honours) in Software and Electronic Engineering

Atlantic Technological University

2022/2023

Figure 1-1 Graphic

Declaration

This project is presented in partial fulfilment of the requirements for the degree of Bachelor of Engineering (Honours) in Software and Electronic Engineering at Galway-Mayo Institute of Technology.

This project is my own work, except where otherwise accredited. Where the work of others has been used or incorporated during this project, this is acknowledged and referenced.

_______________________Igants Lozko_______________________

Acknowledgements

I would like to express my sincere gratitude to all those who have contributed to the completion of this project.

First and foremost, I am deeply thankful to Michelle Lynch for her invaluable guidance, support, and encouragement throughout the duration of this project. Her expertise and insights have been instrumental in shaping the direction of this work.

I am also grateful to Atlantic Technical University for providing the necessary resources and facilities for the successful completion of this project.

I would also like to acknowledge creators and providers if online resources that helped me complete and shape my project.

1. Dave Gray Mern Stack tutorial – For the informative content. MERN Stack Full Tutorial & Project | Complete All-in-One Course | 8 Hours (youtube.com)
2. Programming with Mosh – Information about node technology. Node.js Tutorial for Beginners: Learn Node in 1 Hour (youtube.com)
3. Chris Blakely – Mern stack project Hotel booking app - Complete MERN Stack Project: Build a Hotel Booking App Like a Pro Developer Step-by-Step Course 2024 (youtube.com)
4. Coding with Dawid - Build a Fullstack Booking App using MERN (mongo, express, react, node) (youtube.com)
5. Lama Dev - React Node.js Booking App Full Tutorial | MERN Stack Reservation App (JWT, Cookies, Context API) (youtube.com)
6. Codeevolution – Providing MUI tutorials. React Material UI Tutorial - 1 - Introduction (youtube.com)
7. Stripe Developers - Authentication with stripe-node (youtube.com)

8. Web Dev Simplified - https://www.youtube.com/watch?v=mI1tbIXQI&t=966s&pp=ygUPU3RyaXBlIGFwaSBub2Rl
9. Darwin Tech - https://www.youtube.com/watch?v=XKWJFpZYVAQ&t=526s&pp=ygUPU3RyaXBlIGFwaSBub2Rl
10. Lama Dev – Mapbox tutorial React Node Travel Map App | MERN Stack Full Course Using Mapbox and React Hooks (youtube.com)
11. Code like a Pro – Map and layout. React MERN Project with Node, Express, Mongodb, Firebase, Mapbox. couch surfing or renting rooms (youtube.com)
12. Code with projects - Getting Started with JWT | MERN Authentication & Authorization | Node JS, React & MongoDB | Part 1 (youtube.com)
13. Mehul - Webflow + Attentive 60Sec cc (youtube.com)
14. Projects with Saha - Full-Stack MERN Auth project: Build & Deploy (Reactjs, json web token, jwt, redux toolkit, cookie) (youtube.com)
15. Code with Antonio - Full Stack Airbnb Clone with Next.js 13 App Router: React, Tailwind, Prisma, MongoDB, NextAuth 2023 (youtube.com)

Table of Contents

1 Summary

The aim of this project was to develop a functioning gear rental website, utilizing the MERN stack, Mapbox API, Firebase and Stripe API. Gear Depo provides a seamless rental of outdoor gear, for any demographic, either for personal or businesses. Scope encompassed user authentication, listing creation, search functionality, booking management, and secure payment processing.

Gear Depo tries to bridge the gap between expensive and underutilized outdoor gear with people who are unable to afford expensive gear and businesses who are unwilling to buy or create a rental platform for their business.

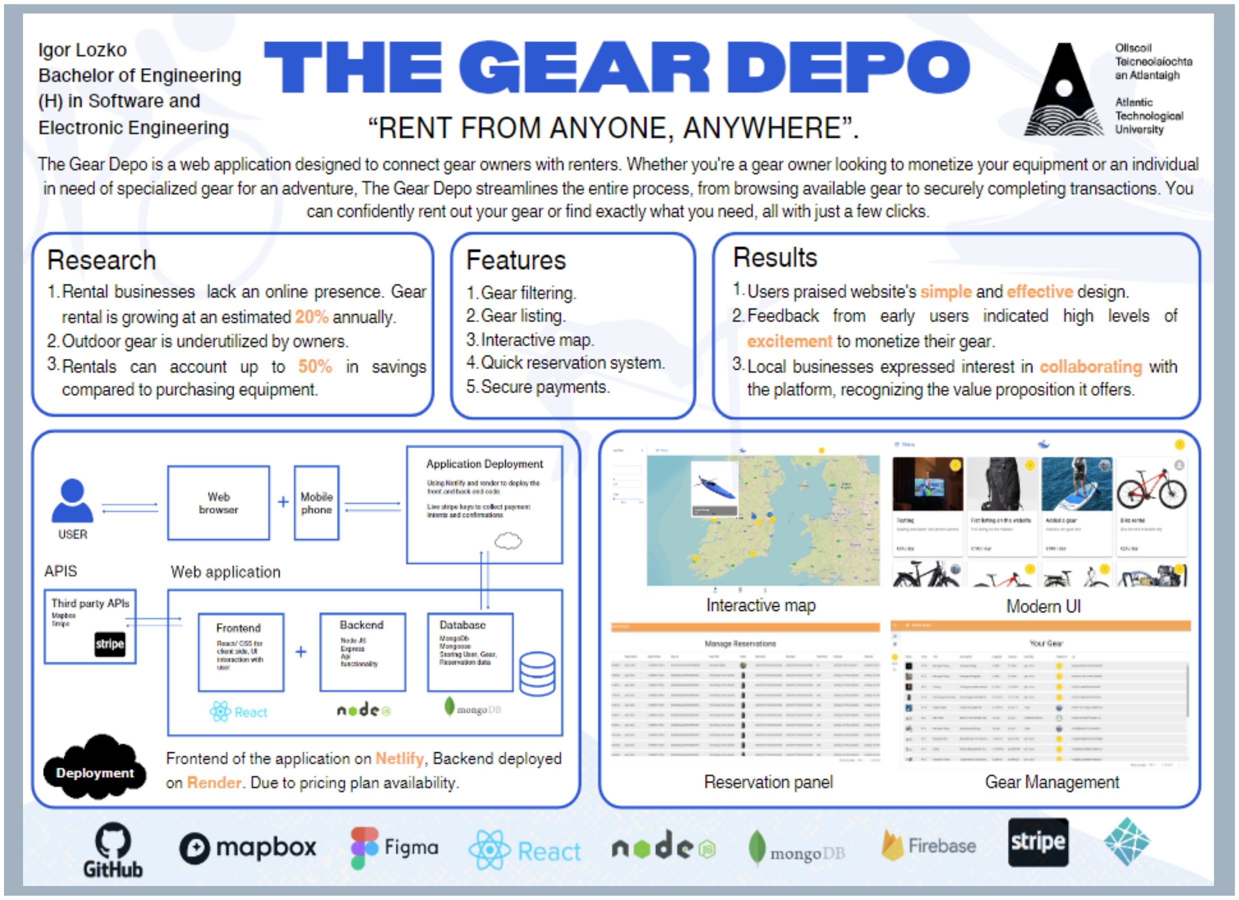
Key features include an intuitive user interface for both renters and gear owners, interactive map integration for location-based searches, and a robust payment system for secure transactions. Catering for all users with a simple and effective Ui design and a coherent structure.

Technologies that I used to develop this platform were MongoDB, Express.js, React.js, and Node.js for the backend part of the project. Mapbox API provided geolocation services and Stripe API provided a secure payment processing function. I've also used Firebase for uploading and storing files.

I successfully implemented a fully functional website enabling users to browse, search, book, and pay for gear rentals. User feedback during testing phases guided refinements to enhance usability and functionality.

This platform focuses on removing a roadblock for many adventures, by offering a solution for affordable and trustworthy outdoor gear that is provided by either other users or businesses.

2 Poster



Igor Lozko
Bachelor of Engineering (H) in Software and Electronic Engineering

THE GEAR DEPO

"RENT FROM ANYONE, ANYWHERE".

Ollscoil Teicneolaíochta an Atlantaigh
Atlantic Technological University

The Gear Depo is a web application designed to connect gear owners with renters. Whether you're a gear owner looking to monetize your equipment or an individual in need of specialized gear for an adventure, The Gear Depo streamlines the entire process, from browsing available gear to securely completing transactions. You can confidently rent out your gear or find exactly what you need, all with just a few clicks.

Research

1. Rental businesses lack an online presence. Gear rental is growing at an estimated 20% annually.
2. Outdoor gear is underutilized by owners.
3. Rentals can account up to 50% in savings compared to purchasing equipment.

Features

1. Gear filtering.
2. Gear listing.
3. Interactive map.
4. Quick reservation system.
5. Secure payments.

Results

1. Users praised website's simple and effective design.
2. Feedback from early users indicated high levels of excitement to monetize their gear.
3. Local businesses expressed interest in collaborating with the platform, recognizing the value proposition it offers.

USER
Web browser
Mobile phone
Application Deployment
Using Netlify and render to deploy the front and back end code.
Use stripe keys to collect payment intents and confirmations.
APIS
Third party APIs
Mapbox
Stripe
Web application
Frontend
React/ CSS for client side, UI interaction with user
Backend
Node JS
Express
Api functionality
Database
MongoDb
Mongoose
Storing User, Gear, Reservation data
Deployment

Frontend of the application on Netlify, Backend deployed on Render. Due to pricing plan availability.

3 Introduction

My goal for this project was to develop a functioning and user-friendly outdoor gear rental website that would be able to handle gear listing creation, gear reservation and secure payment. The motivation behind building this website was to meet the growing demand for outdoor gear rental services and to bridge the gap between expensive gear and people who are not able to afford it. The aim was to create a convenient and intuitive website for a wide range of people from young to old and provide gear owners with a means to monetize their equipment from just a few simple steps.

As gathered from researching rental spaces and markets that 40% of bought outdoor gear is underutilized [6]. Therefore, providing an opportunity for potential gear owners to monetize their gear to people who are seeking to travel or rent gear for a select period. Another beneficial outcome of this website is that it would reduce the need to buy outdoor gear resulting is a lesser production of outdoor gear leading to a greener environment.

During the development of the website new technologies were researched and used. Utilizing MongoDB, Express.js, React.js, and Node.js, alongside integration with Mapbox API for geolocation services, Firebase for image uploading/storing files and Stripe API for secure payment processing. Key functionalities included user authentication with Google and JWT tokens, gear listing creation, search functionality, reservation functionality, and secure payment processing.

This report details the journey from coming up with the idea to researching, outlining the technologies, methods, and tasks employed to achieve my project objectives. Providing insights into the development process, challenges faced, solutions, and outcomes achieved. Additionally, offer an overview of the layout, organization, and structure of the platform, providing a user manual for the website.

4 Research

First stage of the project was the research. This consisted of exploring different sources on the internet and asking friends for information. I focused on finding as much information as possible on different rental websites. Conducting market research proved vital in shaping the project. Technology research was vital in guiding the direction of the project. Researching the web for user experience design, database structures and much more.

4.1 Research and findings.

1. **Rental Business Growth:** According to industry experts, the gear rental market has been experiencing steady growth, with estimates an annual growth rate of around 15-20% in recent years (source: Outdoor Industry Association). [7]
2. **Online Access:** A survey conducted by Gear Trade found that approximately 40% of rental businesses lack a robust online presence, limiting their ability to reach a wider audience and facilitate to the growing demand for online rental services. [8]
3. **Underutilization of Outdoor Gear:** Studies conducted by outdoor gear manufacturers indicate that a significant portion of outdoor gear is not utilized to its full potential by owners. Estimates suggest that up to 40% of outdoor gear remains unused for extended periods, leading to a growing interest in rental and sharing options (source: The North Face). [6]
4. **Cost Savings:** Renting outdoor gear can lead to significant cost savings compared to purchasing new equipment. On average, renters can save up to 50% of the retail price by opting for rental options, making outdoor activities and hobbies more accessible to a broader audience (source: REI Co-op).[8]
5. **Sustainability Impact:** Research by the Outdoor Industry Association (OIA) indicates that the outdoor gear industry contributes significantly to environmental pollution and waste. Approximately 70% of outdoor gear ends up thrown away, highlighting the need for sustainable solutions such as gear rental and sharing initiatives to reduce environmental impact.

6. **Seasonal gear:** Outdoor gear is seasonal and may not be utilized to its fullest potential. As a result, expensive gear could result in sitting collecting dust, therefore and rental platform would resolve this issue by allowing owners to rent their underutilized gear.

After analysing the gathered data mentioned. There is a staggering percentage of underutilized gear, growing trends in minimalism and the impact that rental platforms have on the environment. After careful research and findings, it was clear that perusing this project would be both beneficial for the economic and environmental factors.

4.2 Research Questions:

Several key research questions that guided and shaped my objectives. These questions include:

1. What are the essential features and functionalities required for a successful gear rental platform?
2. How can trust and reliability be established among users in a peer-to-peer rental marketplace?
3. What are the potential challenges and opportunities associated with implementing a gear rental platform using the MERN stack?

With these research question I was able to point myself into the right direction:

1. **Develop a user-friendly** and intuitive platform for renting outdoor gear, incorporating essential features such as listings, search functionality, booking systems, and user profiles.
2. Implement robust **security measures** and trust-building mechanisms to ensure safe and reliable transactions between gear owners and renters.
3. Explore the capabilities of the **MERN** stack in building scalable and efficient web applications for the sharing economy.

THE GEAR DEPO holds a significant implication for outdoor enthusiasts, adventure seekers, and individuals looking to access specialized gear for recreational activities. By providing a convenient and reliable platform for gear rental, with the aim to promote sustainability, accessibility, and community engagement in outdoor recreation.

5 Project Architecture

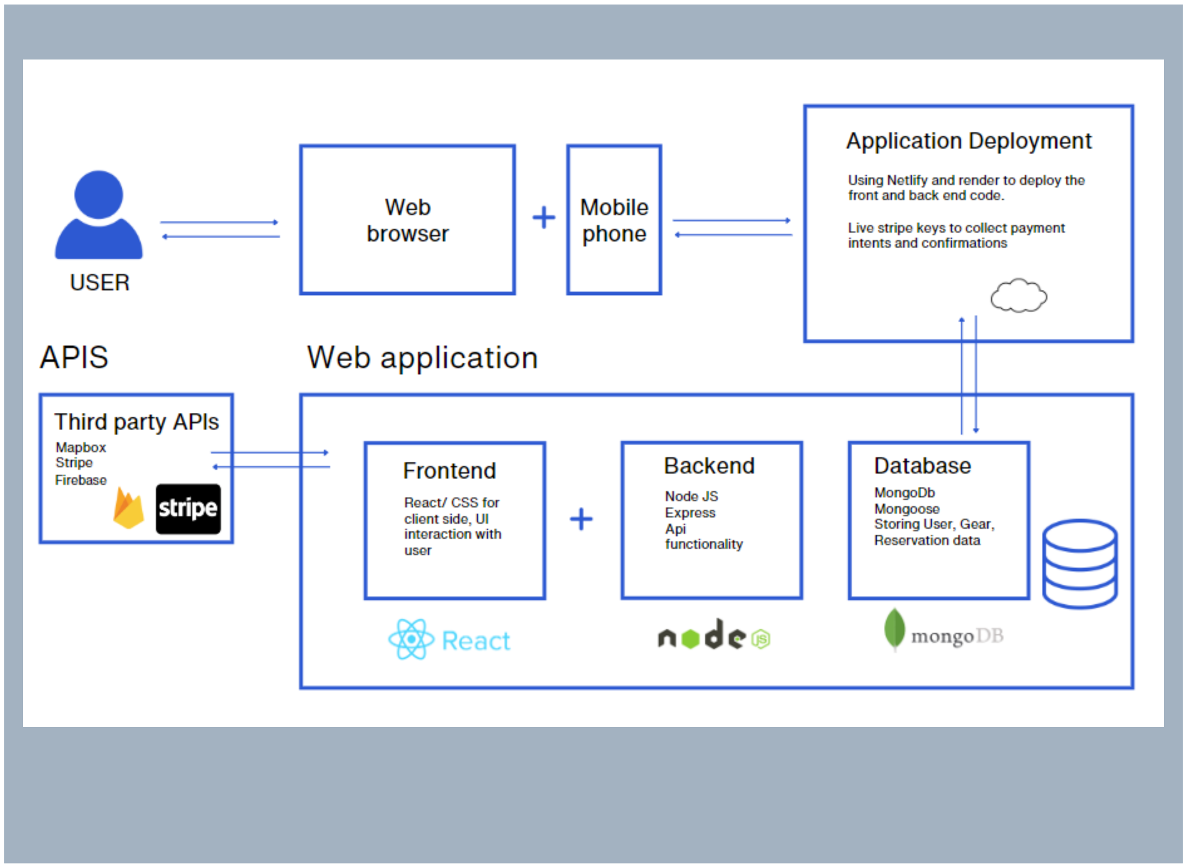

USER
Web browser
+
Mobile phone
Application Deployment
Using Netlify and render to deploy the front and back end code.
Live stripe keys to collect payment intents and confirmations
APIS
Web application
Third party APIs
Mapbox
Stripe
Firebase
stripe
Frontend
React/ CSS for client side, UI interaction with user
+
Backend
Node JS
Express
Api functionality
Database
MongoDb
Mongoose
Storing User, Gear, Reservation data
React
node
mongoDB

Figure 3-1 Architecture Diagram

Client and Server-side folders:

The separation of client and server-side code allows for a clear distinction between the frontend (client-side) and backend (server-side) components. Client-side folder contains code that runs in the user's web browser and handles the presentation layer, including user interface components and interactions. The server-side folder contains code that runs on a web server and handles backend logic, data processing, and communication with external services.

MongoDB Atlas Database:

MongoDB Atlas is a fully managed cloud database service that provides a scalable and reliable storage solution for the application's data. MongoDB Atlas performs CRUD (Create, Read, Update, Delete) operations, such as storing user profiles, gear listings, reservation details, and transaction records.

Map-box API Integration:

The Map-box API is integrated to provide interactive mapping features, such as displaying the location of gear listings, search functionality based on geographic criteria. Client-side code interacts with the Map-box API to fetch map data, render map components, and handle user interactions with the map interface.

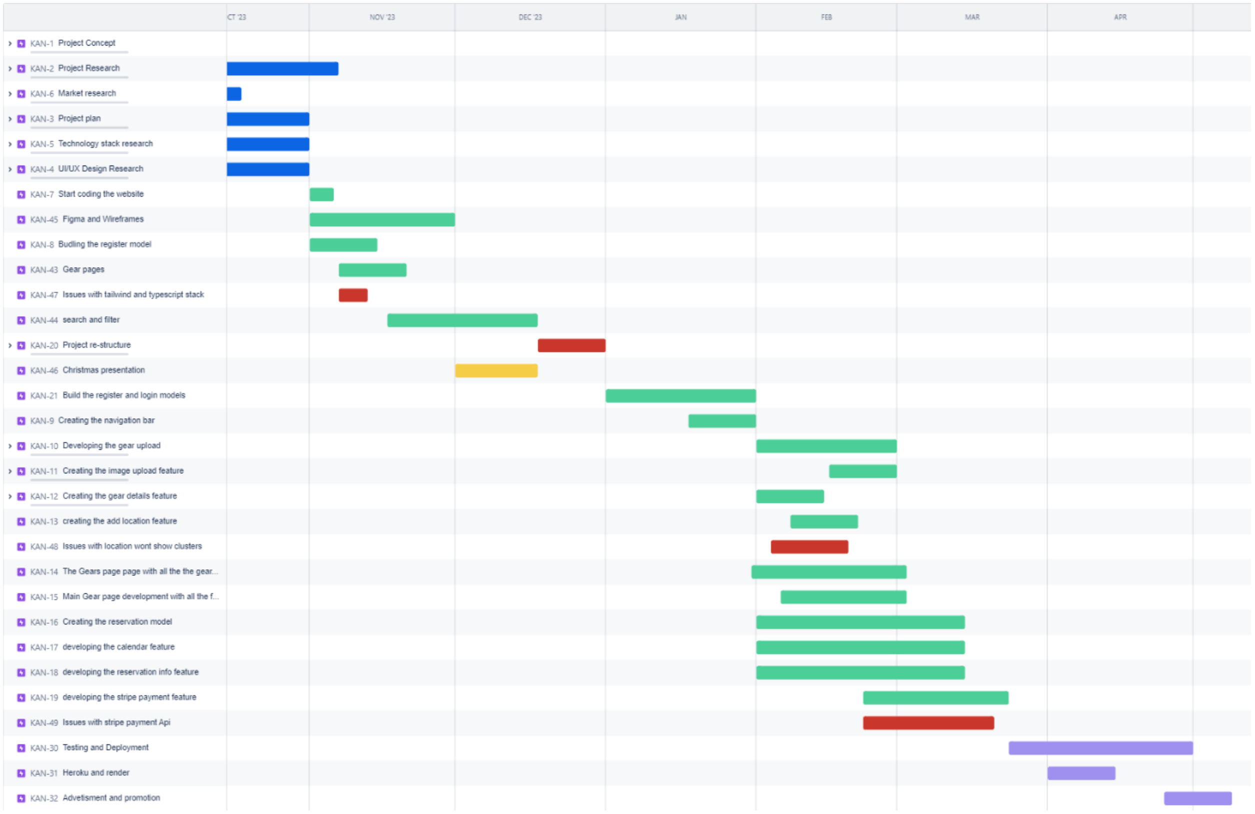
Stripe API Integration:

The Stripe API is integrated to facilitate secure payment processing for gear rentals and transactions. Server-side code interacts with the Stripe API to handle payment authentication, charge creation, and transaction management, ensuring a seamless and reliable payment experience for users.

Deployment on Render or Netlify:

The deployment of application on Render and Netlify enables it to be accessible to users over the internet. Render hosts application's back-end code, providing a scalable and reliable infrastructure for running and managing application in the cloud.

6 Project Plan



KAN-1 Project Concept
KAN-2 Project Research
KAN-6 Market research
KAN-3 Project plan
KAN-5 Technology stack research
KAN-4 UI/UX Design Research
KAN-7 Start coding the website
KAN-45 Figma and Wireframes
KAN-8 Budling the register model
KAN-43 Gear pages
KAN-47 Issues with tailwind and typescript stack
KAN-44 search and filter
KAN-20 Project re-structure
KAN-46 Christmas presentation
KAN-21 Build the register and login models
KAN-9 Creating the navigation bar
KAN-10 Developing the gear upload
KAN-11 Creating the image upload feature
KAN-12 Creating the gear details feature
KAN-13 creating the add location feature
KAN-48 Issues with location wont show clusters
KAN-14 The Gears page page with all the the gear...
KAN-15 Main Gear page development with all the f...
KAN-16 Creating the reservation model
KAN-17 developing the calendar feature
KAN-18 developing the reservation info feature
KAN-19 developing the stripe payment feature
KAN-49 Issues with stripe payment Api
KAN-30 Testing and Deployment
KAN-31 Heroku and render
KAN-32 Advetisment and promotion

Figure 4-1 Graphic

The gear depo began with a plan, first broke the project into 3 phases. As seen on the Jira software.

1. **Phase 1** - Planning and Research. (Blue)
2. **Phase 2** - Design and development. (Green)
3. **Phase 3** - Testing and Deployment. (Purple)

6.1 Phase 1

The Planning and Research phase of the project commenced on September 25th and concluded on October 30th, spanning approximately five weeks. This phase played a crucial role in laying the groundwork for the successful execution of the project, encompassing various activities aimed at developing the idea, conducting comprehensive research, and formulating a robust plan for project development.

1. **Idea Conceptualization:** Brainstorming sessions to develop the project concept.
2. **Project Research:** Literature review and analysis of existing solutions to define project scope.
3. **Market Research:** Studying target market trends, user needs, and competitor analysis.
4. **Project Plan Development:** Creating a detailed project plan on Jirra and on journals outlining objectives, timeline, and resource allocation.
5. **Technology Research:** Exploring suitable technologies and frameworks for project implementation.
6. **Design Research:** Investigating UI/UX design principles and creating wireframes.

6.2 Phase 2

Design and development phase lasted between October 30th to March 29th approximately 22 weeks. During this phase the functionality and the design of the website were constructed and tested. Listed below are the objectives that had to be completed during this phase.

Main features that I wanted to include, reverse engineering the entire process I wrote out all the features and functionalities my website had to contain.

1. An interactive map with gear clusters which are interactive and display an image.
2. A quick and easy log in / sign up functionality.
3. A main gear page with details of al the gears, have a grid block layout.
4. A place where a user can add and upload their gear, enter detail's location and images.
5. The main page for the gear containing all the information about the said gear (Including: images, title, price, calendar, and location, contact details and reviews).
6. A functional reservation system with a calendar and reserved dates I grey which cannot be clicked. (Within a range).
7. A functional payment system with stripe.
8. A filter option to filter all the gears by title price and location.
9. An admin page to display all the user, gear, reservation info.
10. A confirmation email for successful stripe payment.
11. Deployment to Heroku or render or aws.

After writing out all the features that had to be implemented it simplified the process and gave this project structure which could be followed and tracked.

6.3 Phase 3

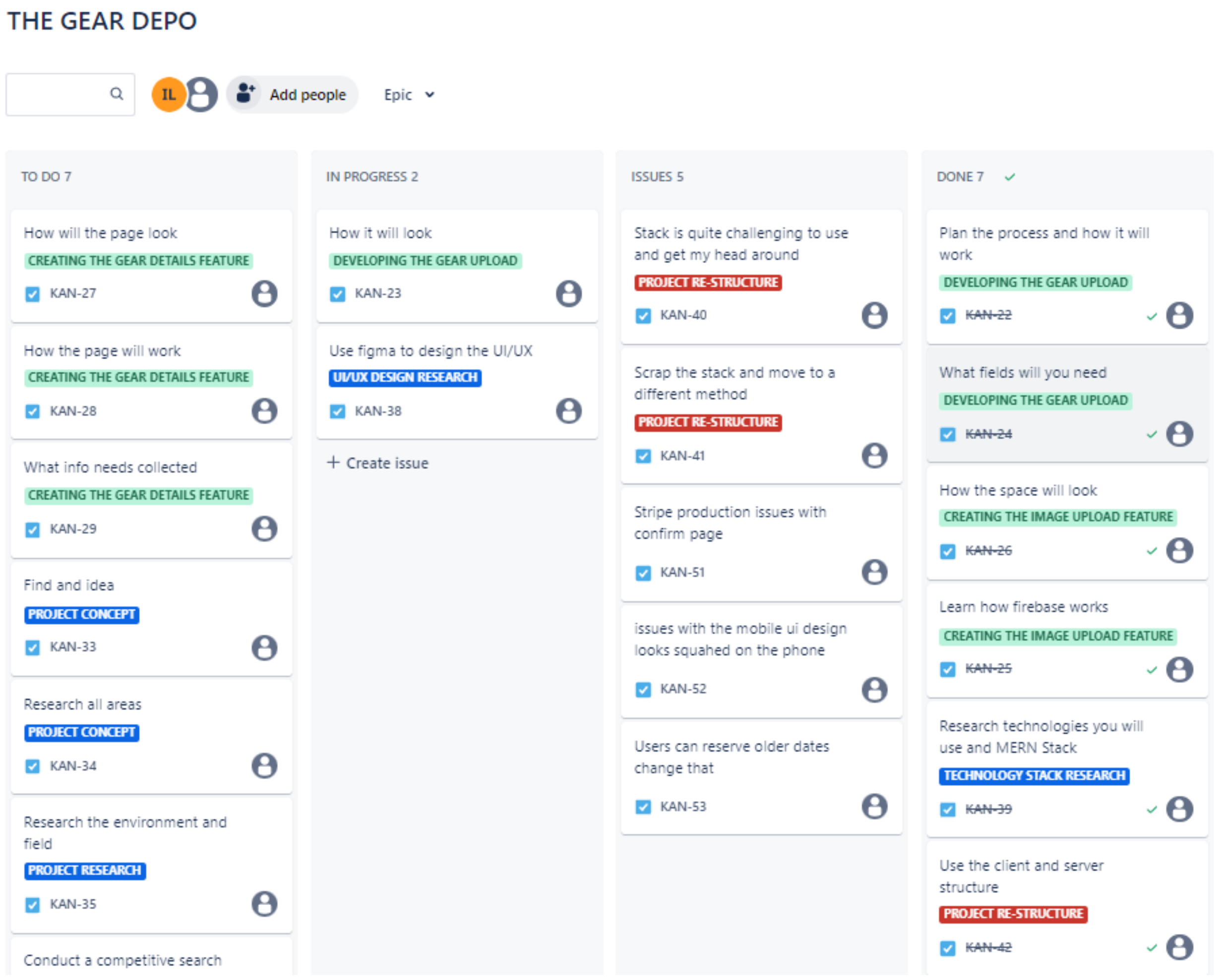
Testing and Deployment. This phase lasted 30th March to April 20th. In this part of the project further testing and fixing was carried out. Deployment took place where the website went live. The backend code was deployed on Render and the frontend code deployed on Netlify.

Deployment stage was conducted with minimal errors and quickly. By integrating the GitHub account with the two platforms and amending the applications codes routes, website was able to go live with ease.

Several methods during the project had been used to monitor and record purposes:

1. **Jira software:** For timeline and issue tracking.
2. **Note taking journaling:** Quick on hand notes.
3. **OneNote:** Collaboration and journaling more in-depth issues and solutions, documenting research.

While the project was ongoing, issues were noted on Jira and progress was monitored. The process was split up into 4 fields, to do, in process, issues and done. This can be seen in figure 5-1.



THE GEAR DEPO

IL Add people Epic

TO DO 7

How will the page look
CREATING THE GEAR DETAILS FEATURE
KAN-27

How the page will work
CREATING THE GEAR DETAILS FEATURE
KAN-28

What info needs collected
CREATING THE GEAR DETAILS FEATURE
KAN-29

Find and idea
PROJECT CONCEPT
KAN-33

Research all areas
PROJECT CONCEPT
KAN-34

Research the environment and field
PROJECT RESEARCH
KAN-35

Conduct a competitive search

IN PROGRESS 2

How it will look
DEVELOPING THE GEAR UPLOAD
KAN-23

Use figma to design the UI/UX
UI/UX DESIGN RESEARCH
KAN-38

+ Create issue

ISSUES 5

Stack is quite challenging to use and get my head around
PROJECT RE-STRUCTURE
KAN-40

Scrap the stack and move to a different method
PROJECT RE-STRUCTURE
KAN-41

Stripe production issues with confirm page
KAN-51

issues with the mobile ui design looks squahed on the phone
KAN-52

Users can reserve older dates change that
KAN-53

DONE 7 ✓

Plan the process and how it will work
DEVELOPING THE GEAR UPLOAD
~~KAN-22~~ ✓

What fields will you need
DEVELOPING THE GEAR UPLOAD
~~KAN-24~~ ✓

How the space will look
CREATING THE IMAGE UPLOAD FEATURE
~~KAN-26~~ ✓

Learn how firebase works
CREATING THE IMAGE UPLOAD FEATURE
~~KAN-25~~ ✓

Research technologies you will use and MERN Stack
TECHNOLOGY STACK RESEARCH
~~KAN-39~~ ✓

Use the client and server structure
PROJECT RE-STRUCTURE
~~KAN-42~~ ✓

Figure 5-2 Graphic

7 Technologies:

The development of the web application involved a strategic approach to utilize various technologies, frameworks, and tools to achieve the project objectives effectively. The methods employed encompassed various stages.

7.1 Technology Stack:

The project was developed using the MERN stack, which comprises of MongoDB, Express.js, React.js, and Node.js. This stack was chosen for its versatility, scalability, and compatibility with the project requirements.

7.1.1 Mern Stack:

1. **MongoDB:** A NoSQL database used for storing and managing data related to user profiles, gear listings, rental transactions, and other application data.
2. **Express.js:** A backend framework for Node.js used to develop the server-side logic, RESTful APIs, and middleware for handling requests and responses.
3. **React.js**: A frontend JavaScript library used for building interactive user interfaces, components, and views for the web application.
4. **Node.js:** A JavaScript runtime environment used for server-side scripting, event-driven architecture, and asynchronous I/O operations.

7.1.2 Reasons for Choosing MERN:

1. **Single Language:** All components of the MERN stack (JavaScript) use the same language, simplifying development and reducing the need for context switching between different programming languages.
2. **Full-stack JavaScript:** With Node.js on the server-side and React.js on the client-side, can work seamlessly across the entire application stack, promoting code reusability and consistency.

3. **Community Support:** The MERN stack has a large and active developer community, providing access to a wealth of resources, tutorials, libraries, and frameworks that can accelerate development and troubleshooting.
4. **Flexibility and Scalability:** MongoDB's flexible document-based data model allows for easy schema changes and scalability, making it well-suited for agile development and handling large volumes of data.
5. **Rich Frontend Development:** React.js offers a component-based architecture, and efficient rendering, enabling to create dynamic and interactive user interfaces with ease.
6. **RESTful API Development:** Express.js simplifies the development of RESTful APIs by providing a lightweight and flexible framework for routing, middleware, and request handling, facilitating seamless integration with frontend components.
7. **Performance and Speed:** Node.js's non-blocking, event-driven architecture enables high-performance, asynchronous I/O operations, resulting in fast and responsive web applications capable of handling concurrent requests efficiently.
8. **Modern Development Practices:** The MERN stack embraces modern development practices such as modularization, componentization, and microservices architecture, promoting code maintainability, testability, and scalability.
9. **Cloud Compatibility:** The MERN stack is well-suited for cloud deployment and integration with cloud-based services and platforms such as AWS, Google Cloud Platform, and Microsoft Azure, enabling scalable and cost-effective hosting solutions.

7.2 API Integration:

1. **Mapbox API:** Integrated Mapbox API for implementing location-based features, such as geocoding, mapping, and route planning. This API enabled users to visualize gear rental listings on interactive maps and obtain directions to rental locations.

React map components and the geolocator were used to get the user precise location.

Fetch ipapi was function used to get the users Ip address and scroll the map to it so to have the user displayed and allow them to see the gear nearby.

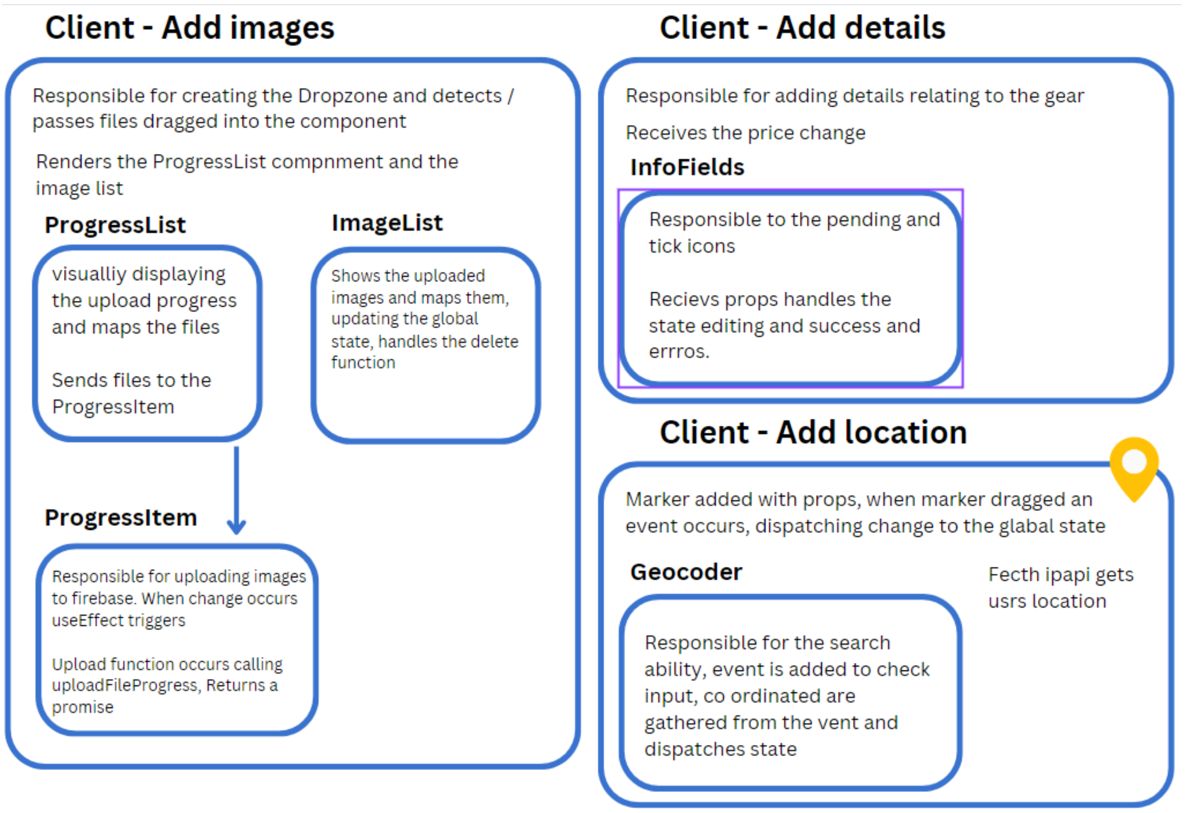

Client - Add images
Responsible for creating the Dropzone and detects / passes files dragged into the component
Renders the ProgressList compnment and the image list
ProgressList
visualliy displaying the upload progress and maps the files
Sends files to the ProgressItem
ImageList
Shows the uploaded images and maps them, updating the global state, handles the delete function
ProgressItem
Responsible for uploading images to firebase. When change occurs useEffect triggers
Upload function occurs calling uploadFileProgress, Returns a promise
Client - Add details
Responsible for adding details relating to the gear
Receives the price change
InfoFields
Responsible to the pending and tick icons
Recievs props handles the state editing and success and erros.
Client - Add location
Marker added with props, when marker dragged an event occurs, dispatching change to the glabal state
Geocoder
Responsible for the search ability, event is added to check input, co ordinated are gathered from the vent and dispatches state
Fecth ipapi gets usrs location

Figure 5-2 Graphic

As seen from the graphic above map box was used in the listing creation part of the project. It was vital to extract the longitude and the latitude of the marker which the user selected by either dragging the marker or by searching the search bar on the right-hand side.

2. **Stripe API:** Integrated Stripe API for implementing secure payment processing functionality, enabling users to make and receive payments for gear rentals seamlessly. This API facilitated secure transactions using industry-standard encryption and compliance with PCI DSS standards.

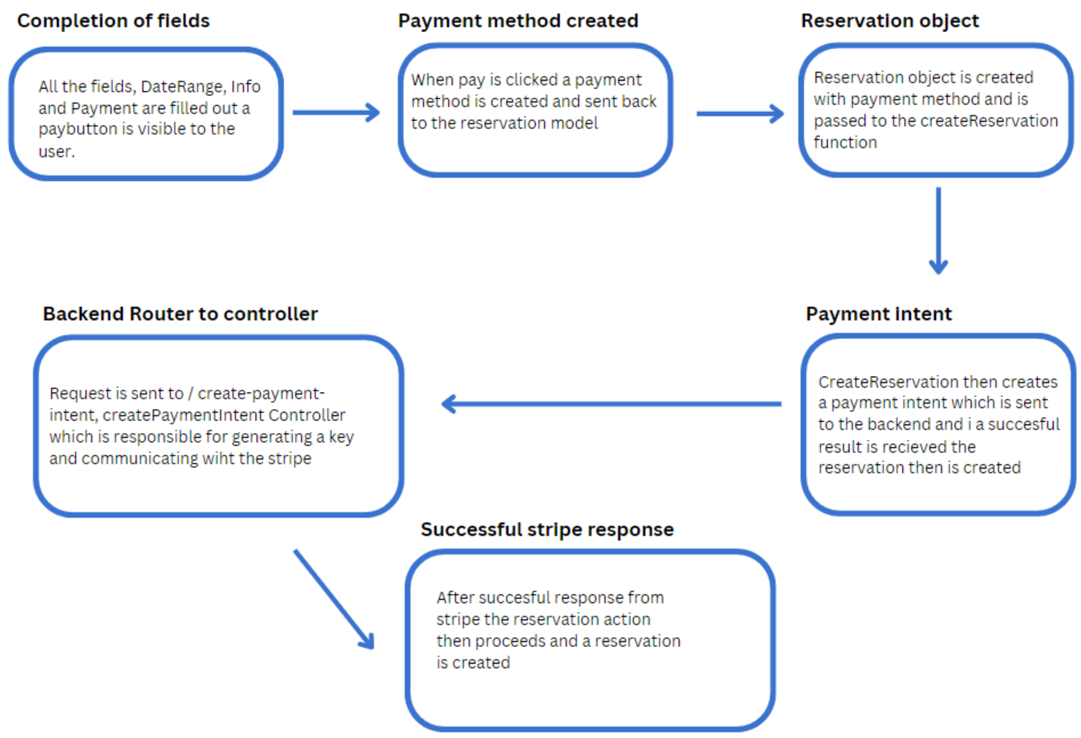

Completion of fields
All the fields, DateRange, Info and Payment are filled out a paybutton is visible to the user.
Payment method created
When pay is clicked a payment method is created and sent back to the reservation model
Reservation object
Reservation object is created with payment method and is passed to the createReservation function
Payment intent
CreateReservation then creates a payment intent which is sent to the backend and i a succesful result is recieved the reservation then is created
Backend Router to controller
Request is sent to / create-payment-intent, createPaymentIntent Controller which is responsible for generating a key and communicating wiht the stripe
Successful stripe response
After succesful response from stripe the reservation action then proceeds and a reservation is created

Figure 5-3 Graphic

As can be seen from the image above this was the architecture for the stripe and reservation system. Once all the fields had been created in the 3-step reservation process, A payment method was created inside the Payment fields component. Here the component extracted all the card and user information and passed it back to the main reservation where an object could be created.

After the creation on the object, information was sent to the gear function in actions where a request to the backend was made. If a payment intent came back successful from the backend only then the reservation could continue and create the booking for the chosen gear. This methodology was chosen as to limit scams and users creating bookings without paying for them.

7.3 Development Tools:

1. **Visual Studio Code:** Utilized Visual Studio Code as the primary (IDE) for coding, debugging, and managing project files.

2. **GitHub:** GitHub repository for collaborative development, code management, and version tracking. This facilitated a platform where code could be easily managed and maintained. Unfortunately, when uploading the project sometimes it would fail as the size of the files were to large resulting in me deleting the node modules folder and uploading the project to a ne repository each time.

7.4 Frontend Development:

1. **React.js:** Leveraged React.js to develop the frontend components, user interfaces, and views for the web application. This included creating reusable components, managing state and props, and implementing client-side routing for navigation between pages.

2. **CSS**: Using CSS styling user interfaces and implementing responsive design principles to ensure compatibility across devices and screen sizes.

3. **JavaScript/ES6:** Used JavaScript/ES6 to add interactivity, dynamic content, and client-side functionality to the web application, such as form validation, event handling, and asynchronous data fetching.

7.5 Backend Development:

1. **Node.js and Express.js:** Developed the backend server using Node.js and Express.js, implementing RESTful APIs, middleware, and business logic for handling user requests, authentication, authorization, and data processing. Using basic HTTP requests it was possible to create, delete, update and edit the information on the website.

2. **MongoDB:** Integrated MongoDB as the database management system for storing and querying application data, utilizing MongoDB Atlas for cloud-based database hosting and management. Implemented Mongoose ORM for schema validation, data modelling, and CRUD operations.

7.6 Deployment and Hosting:

The front end of the application was deployed on **Netlify** while the backend of the code was deployed with **render**.

Both the front end and backend deployment were seamless. By connecting the GitHub repository to the chosen providers. Building configuration settings and setting environment variables. The process was nearly complete. After changing the links within my code with the backend URL for connectivity the application was deployed.

The application would drop and stop working after 30 minutes of inactivity.

Other tools used:

1. **Firebase:** Integrated Firebase into the project to provide a path for the cloud image storing within the project. Allowing the user to upload and change their photos.
2. **JWT/Google login:** Jason be Token were implemented inside the application to provide an authentication feature. Google login was also implemented to speed up account creation.

8 Design of website

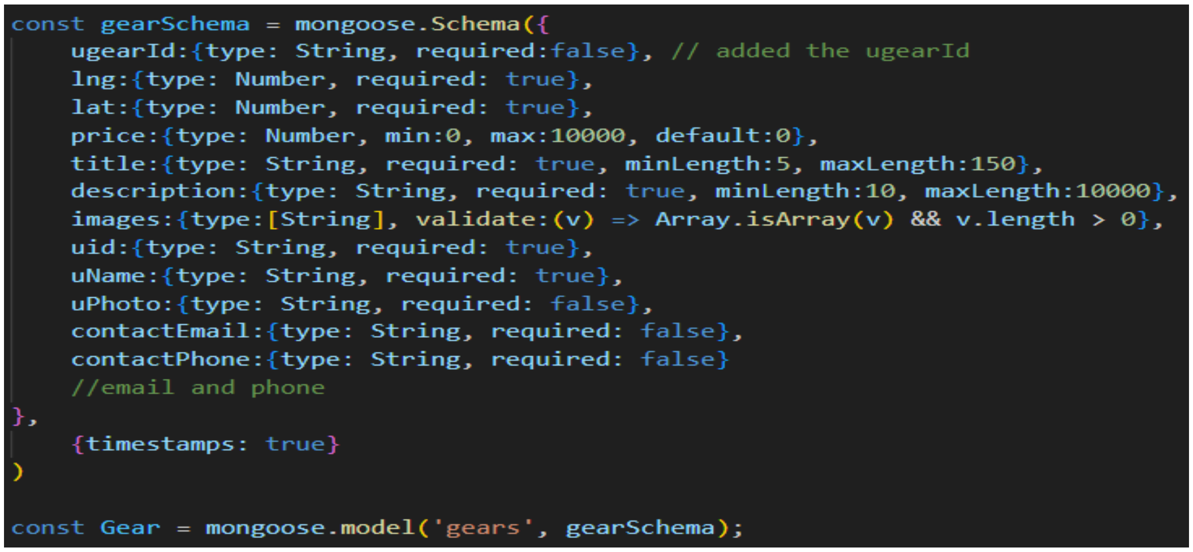
This part of the project will focus on UI sketches, designing the database schematics and writing out routes.

8.1 Database schema

3 fields of data are collected the user information, reservation, and the gear information.

1. **Gear**

User data block collects all the necessary information needed to create a listing for the web application. Some fields are marked as required meaning that they must be included in the listing creation process.



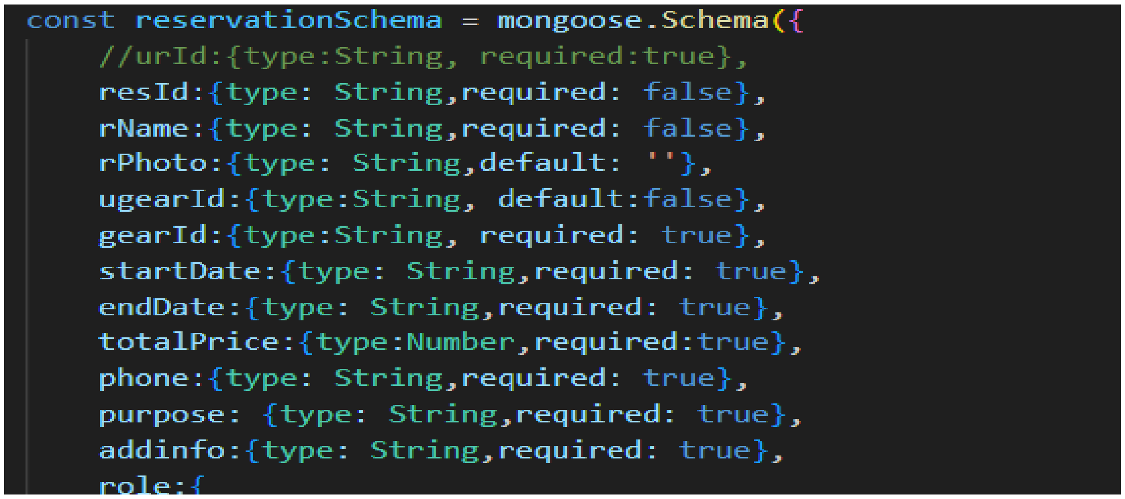
```
const gearSchema = mongoose.Schema({
    ugearId:{type: String, required:false}, // added the ugearId
    lng:{type: Number, required: true},
    lat:{type: Number, required: true},
    price:{type: Number, min:0, max:10000, default:0},
    title:{type: String, required: true, minLength:5, maxLength:150},
    description:{type: String, required: true, minLength:10, maxLength:10000},
    images:{type:[String], validate:(v) => Array.isArray(v) && v.length > 0},
    uid:{type: String, required: true},
    uName:{type: String, required: true},
    uPhoto:{type: String, required: false},
    contactEmail:{type: String, required: false},
    contactPhone:{type: String, required: false}
    //email and phone
},
    {timestamps: true}
)

const Gear = mongoose.model('gears', gearSchema);
```

Figure 6-1 Database Gear

2. Reservation

Reservation block collects all the necessary information for creating a reservation. A reservation block I not activated until a stripe payment is complete therefore a stripe success must be passed for this block to be activated.

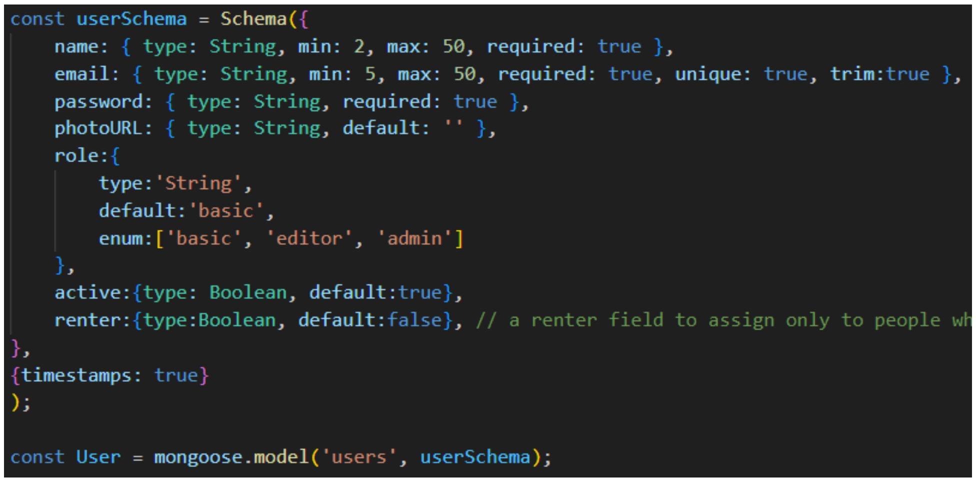


```
const reservationSchema = mongoose.Schema({
    //urId:{type:String, required:true},
    resId:{type: String,required: false},
    rName:{type: String,required: false},
    rPhoto:{type: String,default: ''},
    ugearId:{type:String, default:false},
    gearId:{type:String, required: true},
    startDate:{type: String,required: true},
    endDate:{type: String,required: true},
    totalPrice:{type:Number,required:true},
    phone:{type: String,required: true},
    purpose: {type: String,required: true},
    addinfo:{type: String,required: true},
    role:{
```

Figure 7-2 Database Reservation

3. User

Collects all the necessary information for the creation of a user's account. Passing fields such as password and role.

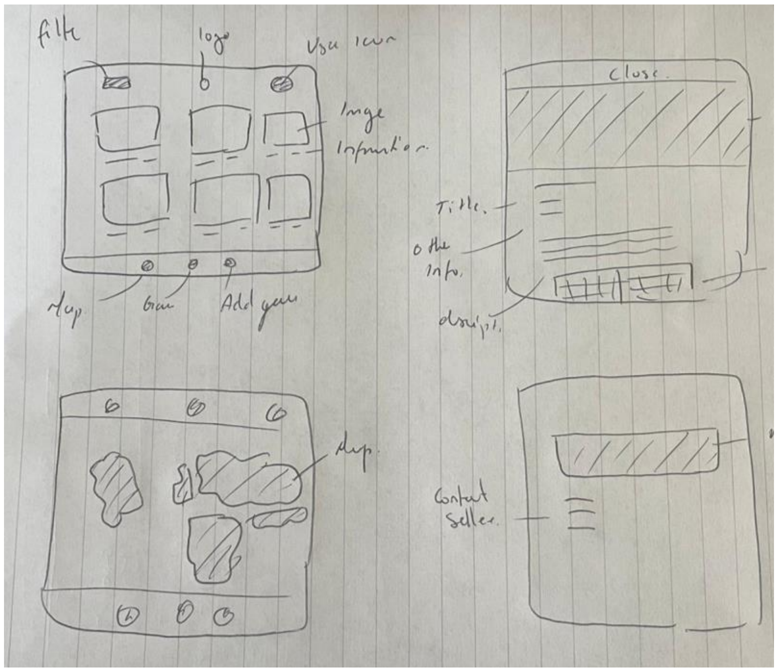


```
const userSchema = Schema({
    name: { type: String, min: 2, max: 50, required: true },
    email: { type: String, min: 5, max: 50, required: true, unique: true, trim:true },
    password: { type: String, required: true },
    photoURL: { type: String, default: '' },
    role:{
        type:'String',
        default:'basic',
        enum:['basic', 'editor', 'admin']
    },
    active:{type: Boolean, default:true},
    renter:{type:Boolean, default:false}, // a renter field to assign only to people wh
},
{timestamps: true}
);

const User = mongoose.model('users', userSchema);
```

Figure 8-3 Database User

8.2 UI mock-ups



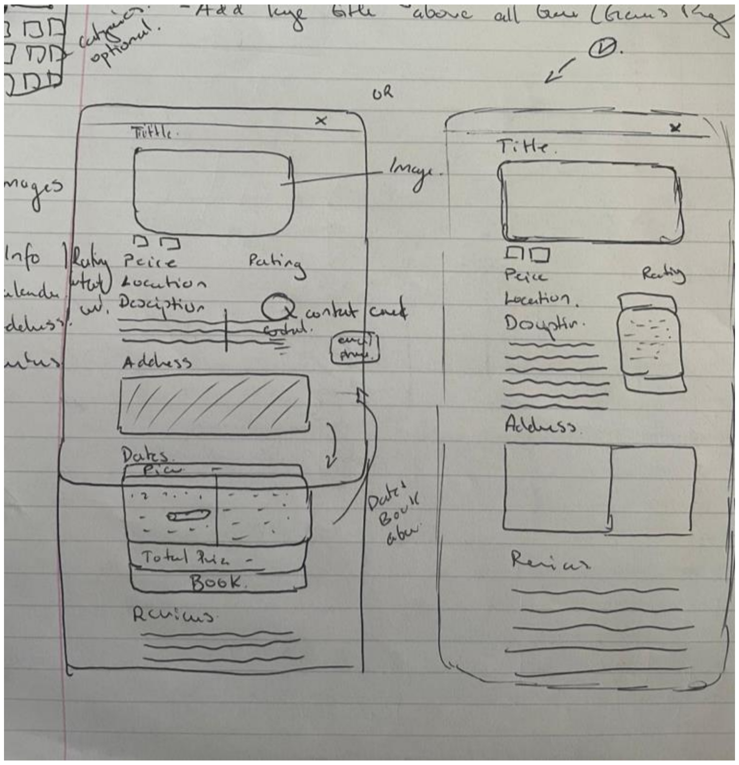


Figure 9-4 UI Drawings

Free hand drawings of the UI in the development stage. The final design had been altered to speed up the development process to achieve the deadline.

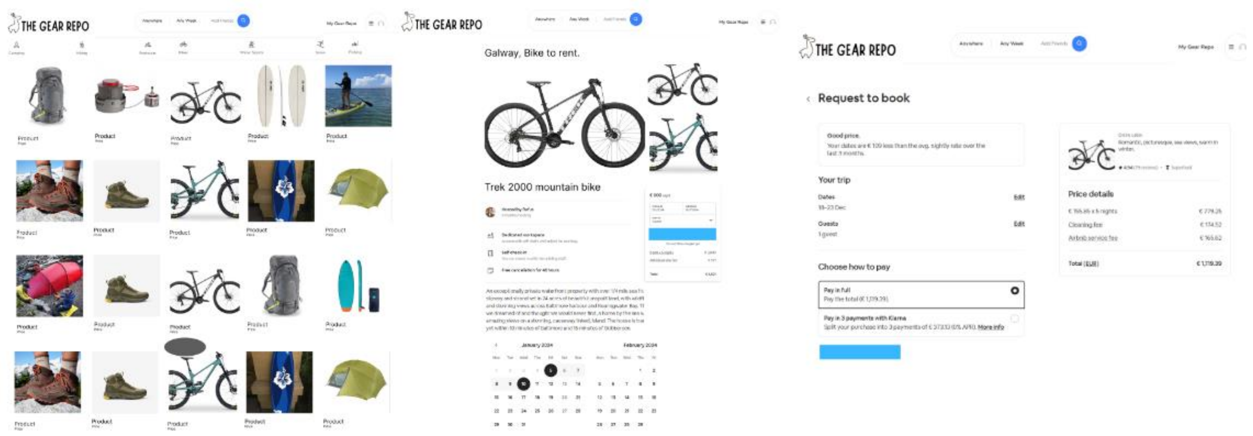
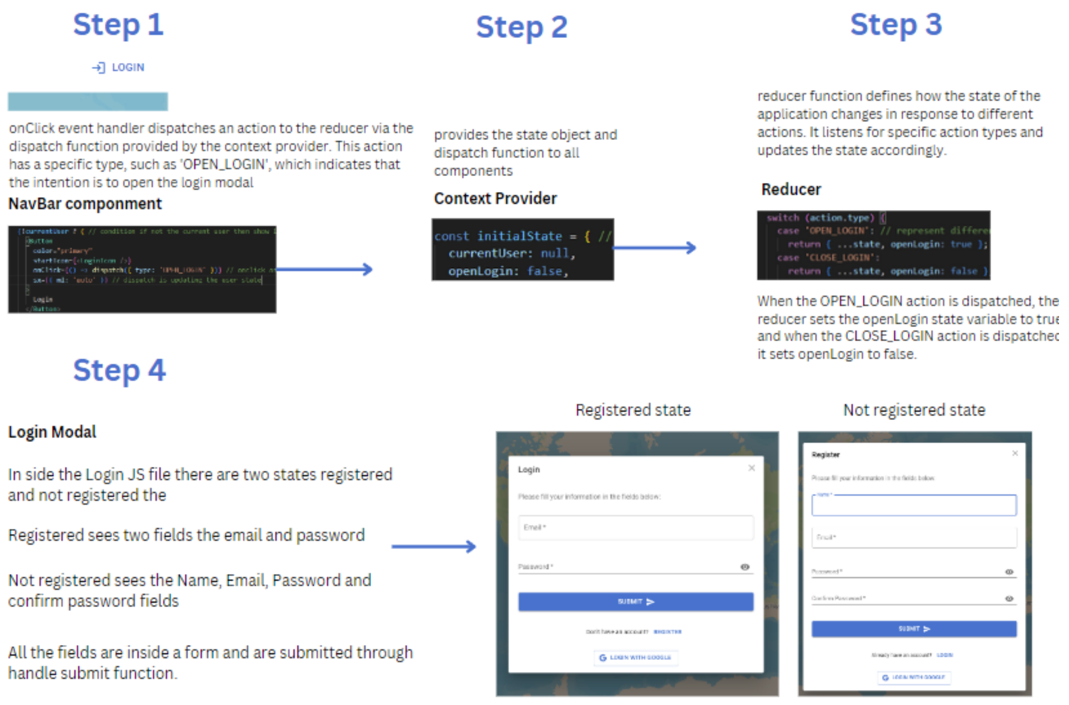


Figure 10-5 Figma

Picture above is a screenshot from Figma of how the initial UI was to design as but it was altered to meet the deadline.

8.3 Login in functionality



onClick event handler dispatches an action to the reducer via the dispatch function provided by the context provider. This action has a specific type, such as 'OPEN_LOGIN', which indicates that the intention is to open the login modal

NavBar componment

```
{!currentUser ? ( // condition if not the current user then show l
  <Button
    color="primary"
    startIcon={<LoginIcon />}
    onClick={() => dispatch({ type: 'OPEN_LOGIN' })} // onclick a
    sx={{ ml: 'auto' }} // dispatch is updating the user state
  >
    Login
  </Button>
```

provides the state object and dispatch function to all components

Context Provider

```
const initialState = { //
  currentUser: null,
  openLogin: false,
```

reducer function defines how the state of the application changes in response to different actions. It listens for specific action types and updates the state accordingly.

Reducer

```
switch (action.type) {
  case 'OPEN_LOGIN': // represent differe
    return { ...state, openLogin: true };
  case 'CLOSE_LOGIN':
    return { ...state, openLogin: false }
```

When the OPEN_LOGIN action is dispatched, the reducer sets the openLogin state variable to true and when the CLOSE_LOGIN action is dispatched it sets openLogin to false.

Login Modal

In side the Login JS file there are two states registered and not registered the

Registered sees two fields the email and password

Not registered sees the Name, Email, Password and confirm password fields

All the fields are inside a form and are submitted through handle submit function.

Figure 11-6 Login Functionality

Pictured above is the process used in building and understanding the functionality of the login modal used in the project. Breaking down and following the steps taken to achieve the login modal. MUI components helped to build and shape the websites design and functionality.

9 Implementation:

9.1 Coding Practices:

1. **Modular Development:** Adopted a modular approach to development, breaking down the project into smaller components and modules to facilitate code organization, reusability, and maintainability. For example, I analysed all the relevant components I needed like the Reservation model, then broke that into 3 features – Date selection, Details field and Stripe payment. This was then broken down into even smaller components and developed. I was able to reuse features like text fields in other components.
2. **Component-based Architecture:** Leveraged component-based architecture in React.js to create reusable UI components, such as forms, modals and password fields promoting consistency and efficiency in frontend development. Functions such as fetchData and hooks like useCehcktoken were all reused inside the application speeding up the process.
3. **RESTful API Design:** Designed and implemented RESTful APIs using Express.js to enable communication between the frontend and backend components of the web application. Followed REST principles for resource naming, HTTP methods, and data representation to ensure clarity, consistency, and scalability in API design.
4. **Database Schema Design:** Designed MongoDB database schemas using Mongoose, defining data models, relationships, and validation rules to ensure data integrity, efficiency, and scalability in data storage and retrieval. Specified relevant fields which had to be entered and stored on the Atlas.

9.2 Challenges Faced:

1. **Integration Complexity:** Faced challenges in integrating third-party APIs, such as Mapbox and Stripe, due to differences in documentation, authentication mechanisms, and data formats. Had numerous of issues when trying to implement the Stripe Api with payment intents not being successful or extracting the card data incorrectly. Overcame this challenge by thoroughly studying API documentation, testing API endpoints, and

implementing error handling and fallback mechanisms to handle integration issues gracefully. I also used the console.log method to figure out and trace back where the issue was occurring a method that proved to be quite useful. Organisation of components and clear coding proved useful when solving this issue.

2. **Performance Optimization:** Encountered performance bottlenecks, such as slow database queries and frontend rendering, leading to suboptimal user experience and application responsiveness. Addressed this challenge by implementing lazy loading techniques to optimize database queries and frontend rendering, improving application performance and scalability.
3. **Security Vulnerabilities:** Faced numerous issues with security and data protection, had to figure out a correct way with dealing with breaches or scams. Decided to implement user rights so only select few can post gear ads. In Addition, implemented a limit of 30mb of json object data that can communicate with he backend to avoid data overload.
4. **Rendering reservation model:** Faced an issue with the reservation model being available for a second payment for the same dates which caused double booking. Fixed this issue by implementing a new state open and close and clear for the reservation model. The model would close once a successful payment would be created.
5. **Double booking and date selection:** Faced an issue where users could double book same dates. Fixed this issue by creating reserved dates, a function get reserved dates would gather all the reserved dates within the database relevant to the gear id. Then a function in the calendar section would render and filter the dates and if it came back withing a reserved date range the dates would be blocked and unclickable.
6. **Adding images:** Faced issues when using firebase for image upload, images weren’t uploading correctly to the server causing a broken icon. This issue was fixed by studying the documentation and figuring out the correct URL folder point for the images and the storage location.
7. **Stripe payment total:** Stripe total payments were coming back in cents when a total price was displayed. After reading the documentation figured out that the total before

being sent to the stripe service had to be multiplied by 100 to convert from cents to euros. As noted, stripe API only accepts cent figures.

8. **Error handling:** Issues encountered that messages weren't displaying for users when something went wrong. Implementing a screen message to pop up when a user made the incorrect action was crucial for user experience.
9. **User authentication and authorisation:** Challenge: Implementing secure authentication and authorization mechanisms to verify user identities, protect sensitive resources, and enforce access control policies. Solution: Implemented user authentication using secure authentication protocols, such as OAuth 2.0 or JSON Web Tokens (JWT), to authenticate users securely and generate access tokens. Implemented role-based access control (RBAC) or attribute-based access control (ABAC) to define and enforce fine-grained authorization policies, ensuring proper access control and data protection.
10. **Project management**: Had issues recording and reporting project status on OneNote, overcame this issue by allowing myself to take notes on pen and paper when an issue was encountered and logging all the issues on a Sunday.
11. **Image Carousel:** Encountered several issues when trying to implement the swiper component from react, images weren't loading, tried solving the issue but I couldn't come to a resolution, decided to take actions, and change from swiper to carousel. Carousel worked and I scraped swiper.
12. **UI design:** During the development stage I encountered issues with my current design of the website had to change the UI to speed up the development while maintaining a user-friendly interface. Changed the booking form to have a 3-step process on the same page as the Gear page and changed the gear upload to spend less time developing a more coherent user interface. The initial design was too complex, and I was running into issues while using tailwind as it was a new technology reverting to CSS.
13. **Developing the website process or order:** To achieve a smooth and effective experience, A detailed plan and design had to take place to figure out the concept from the Users and Renters perspectives what would the user view and would need to see

and same for the renter. Drawing wireframes and deigning webpages took place to achieve a systematic result.

14. **Issues with main features backend:** When developing the main features (Gear, Reservation, User) encountered multiple issues when it came to structuring and implementing. To overcome this, I designed a system consisting of:
    a. **Controllers** – Responsible for handling incoming requests, processing data and sending back request,
    b. **Routes** – Act as entry points for incoming requests from the client side e.g. reservation router
    c. **Middleware** – Intercept incoming requests before they reach the routes auth,
    d. **Models** – Data structures which must be filed and later stored on mongo atlas.

    Request fist hits the middleware where it is checked and authorized, then if passes it reaches the router which matches the URL, HTTP method, controller execution is then carried out when method invokes controller, Controller interacts with model to perform operations.
15. **Security and user permissions:** Basic user, renters and admin all had to be separated and had to contain different rights. A user must not be allowed to create a gear to avoid scamming and false advertising. To overcome this user permissions were designed.
    a. **User Perm –** Defines specific permissions status,
    b. **Auth –** extracts the JWT token ad verifies validity if valid decodes token and payload to extract user info.
    c. **Check Access** responsible for enforcing access control based on user permissions.
16. **Saved data after entry:** After a user entered their information of the gear or reservation. The data was saved on the website. To resolve this issue a context provider and reducer were used and called after a successful creation the callback would reset the values.
17. **Gear and reservation identification:** For clarity utilizing the Atlas id and unique gear and reservation id for retrieval and identification of the listings was crucial for organisation.

The reserved dates are fetched by the relevant gear id, and then passed to the function blocking the dates.

10 Website Features and Functionalities:

This section will discuss all the features detailed on The Gear Depo website.

10.1 Interactive map

1. **Clusters -** Utilizes clustering to group nearby gear listings for improved map visualization and user experience.
2. **Map navigation -** Provides intuitive navigation controls, such as pan, zoom, and rotation, for users to explore the map.
3. **Pop up window -** Displays additional details of gear listings, such as title, location, and availability, in a pop-up window when users interact with map markers.

10.2 Filtering and searching

1. **Search by title -** Allows users to search for gear listings by title or keywords, facilitating quick and targeted searches.
2. **Search by location -** Enables users to search for gear listings based on location, such as city, neighbourhood, or specific geographical coordinates.
3. **Filter by price -** Offers filtering options to refine search results by price range, allowing users to find listings within their budget.

10.3 User authentication

1. **Log in model -** Provides a secure login model where registered users can log in using their email address and password.
2. **Sign up model -** Offers a user-friendly registration process with options to sign up using email or Google accounts. Using the Salt 12 method in JWT, making the passwords complexity and adding characters with every iteration.
3. **Google one tap login -** Enhances user convenience with Google One Tap Login, allowing users to log in quickly and securely with their Google credentials. Conducting all the

security checks for example making sure that the google token is more than 1000 characters.

10.4 Gear page.

1. **Image -** Displays high-quality images of gear listings to showcase their appearance and condition effectively. Using the Carousel feature in react.
2. **Information -** Presents comprehensive information about gear listings, including description, specifications, and rental terms.
3. **Location -** Indicates the geographical location of gear listings on the map, providing users with context and proximity information.
4. **Contact information -** Offers contact details of gear owners for users to inquire about rentals or arrange pick-up/delivery.

10.5 Reservation model

1. **Calendar with reserved dates -** Incorporates a calendar view displaying reserved dates for each gear listing, allowing users to check availability and plan bookings accordingly.
2. **Reservation details -** Provides users with detailed information about their reservations, including rental dates, pricing, and payment status.
3. **Stripe payment -** Facilitates secure payment processing through integration with the Stripe payment gateway, ensuring seamless transactions for gear rentals.

10.6 Gear grid.

Image and box styling Presents gear listings in a visually appealing grid layout with optimized image display and styled boxes for enhanced presentation.

10.7 Gear listing.

1. **Gear location -** Indicates the geographical location of gear listings on the map, providing users with visual context and navigation assistance.
2. **Details -** Offers detailed information about each gear listing, including specifications, features, and rental terms, to help users make informed decisions.
3. **Images -** Showcases high-quality images of gear listings to attract users' attention and provide visual representation of the items.

10.8 Profile management.

Allows users to manage their profiles, update personal information, and customize account settings according to their preferences.

11 Testing and Quality Assurance:

During the development, Testing and Quality Assurance (QA) were vital for ensuring the reliability and performance of the website. Testing involves various methodologies, from unit tests to end-to-end tests, to verify functionality. QA focuses on establishing processes and standards to maintain quality. Together, they form a comprehensive framework for delivering high-quality software, reducing defects, and enhancing the application.

Testing and quality assurance were accomplished by:

1. Unit testing
2. Performance testing
3. Compatibility testing
4. API testing
5. User feedback

11.1 Unit testing

Unit testing was achieved by identifying individual units or components of code. Isolated pieces of functionality such as functions, methods, or classes. For instance, when testing the reservation model. Using the web browser console, it was notable to see the process of the code execution. When an error was met, I used console log to figure out at what part the code was failing this gave me a better understanding as to where the code was nonfunctional.

Creating a testing with false data, creating test components proved to be helpful when reaching the desired outcome. Writing specific scenarios or conditions that verify the behaviour and output of the unit under test.

11.2 Performance testing

Testing the speed and responsiveness of the application. It involves simulating scenarios to measure how the system performs and behaves under different loads and stress levels. Performance testing helps identify potential bottlenecks, resource constraints, and performance issues that could affect the overall user experience.

To achieve performance testing, several key steps

1. **Design Test Scenarios:** I thought about different ways people might use my app, like browsing, making transactions, or doing heavy tasks. Then, I made scenarios to simulate those situations.
2. **Monitoring system:** While running the tests, I kept an eye on things like how much the computer's brain (CPU) and memory were being used and how long it took for the app to respond. Utilising the web browser.
3. **Look at Test Results**: After the tests, I looked at what happened. I checked if the app did what it was supposed to and if it was as fast as I wanted it to be. If something wasn't right, I tried to figure out why.
4. **Make Things Better and Test Again**: If I found things that needed fixing, I made changes to the app and tested it again. I kept doing this until the app worked the way I wanted it to.

The results of performance testing provide valuable insights into the performance characteristics of the software application under test. These insights enabled me to make decisions about system scalability, capacity planning, and performance optimization strategies. Additionally, performance testing results help ensure that the application can handle expected loads and provide a satisfactory user experience under peak usage conditions.

11.3 Compatibility testing

Testing that ensures the compatibility of an application or system across different platforms, devices, browsers, and environments. The goal of compatibility testing is to verify that the software functions correctly and displays properly across various configurations, thereby providing a consistent user experience to all users. Here's how compatibility testing is achieved and its results:

1. The first step in compatibility testing was to identify the devices my application would run on most. I chose desktop and mobile phone. This may include popular platforms such as Windows, macOS, iOS, Android, Chrome, Firefox, Safari, and Internet Explorer.
2. During the development constant testing and resizing of the web browser was carried out to maintain user friendly UI. Changing the web browser screen after creating each component or function proved useful. For example, when creating the calendar component half of the calendar was cut off on the mobile phone browser therefore removing some of the features on the left hand side and resizing solved this issue.
3. All the UI issues were recorded and documented. Such as calendar being cut off, the navbar being too low and the footer being placed in the wrong position. These issues were fixed later in the project.
4. Collaborate with lecturers and other students, showcase the UI and looked for improvements and where necessary changes should have been made. For example, the Standard Mapbox map UI was changed from standard to dusk to be more pleasing on the eye.
5. After implementing fixes, retest the application across the affected test environments to ensure that the compatibility issues have been resolved and the application had been improved.

The results of compatibility testing provide valuable insights into the application's ability to reach a wide audience and deliver a consistent user experience across diverse environments. By ensuring compatibility across multiple platforms and devices, can enhance user satisfaction and minimize the risk of user abandonment due to technical issues or usability challenges.

12 Deployment:

Deployment of the website was achieved with two services Render and Netlify. The frontend of the code which is under the client folder name was published on Netlify and the backend part of the application under the server folder was published on the render service. Website can be accessed by users by following this link.

https://thegeardepo.netlify.app/

12.1 Why render and Netlify were chosen.

1. **Ease of Use:**

Render and Netlify both offer intuitive user interfaces that simplify the deployment and management of web applications. Their dashboards provide clear navigation and easy access to essential features, making it straightforward to configure and monitor their deployments. Additionally, both platforms offer comprehensive documentation and tutorials to quickly and troubleshoot any issues they encounter.

2. **Scalability:**

Render and Netlify use modern cloud infrastructure that can automatically scale resources up or down based on traffic patterns and application demands. This ensures that the application can handle sudden spikes in traffic without performance degradation or downtime, providing a seamless experience for users during peak usage periods.

3. **High Availability:**

Both Render and Netlify distribute systems to maximize uptime and minimize service disruptions. They deploy applications across multiple data centres and availability zones, ensuring that the application remains accessible even in the event of hardware failures or network outages.

Additionally, they offer robust monitoring and alerting systems to proactively detect and mitigate any issues that may arise.

4. **Built-in Features:**

Render and Netlify provide a rich set of features, including HTTPS support, custom domain configuration, automatic SSL certificate provisioning, and integration with version control systems like Git. These features eliminate the need to manually configure and manage infrastructure components, allowing to focus on building and improving applications.

5. **Cost-effectiveness:**

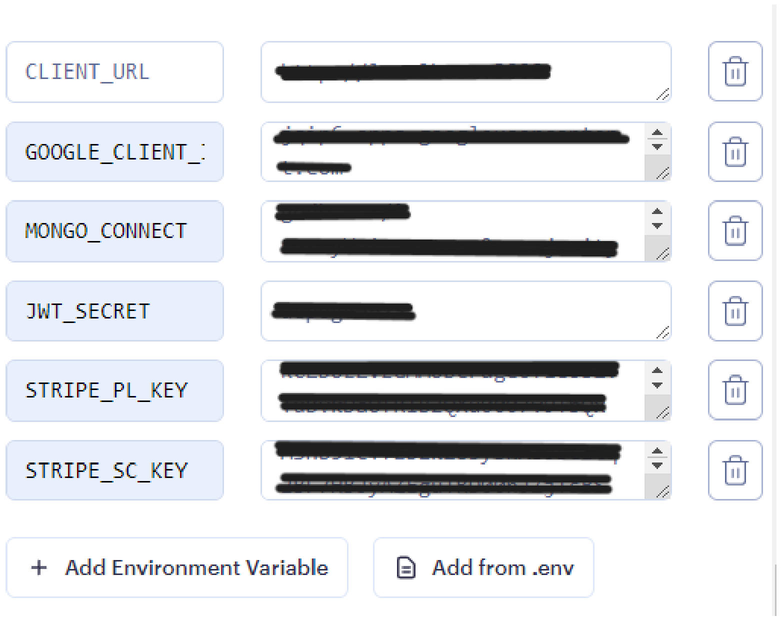
Both Render and Netlify offer transparent and flexible pricing plans that cater to a wide range of budgets and usage patterns. They provide free tiers with generous usage limits for small projects and startups, making it easy to get started without incurring significant costs. For larger applications, they offer pay-as-you-go pricing models with transparent billing and no long-term commitments, allowing to scale infrastructure cost-effectively as application grows.

6. **Community Support:**

Render and Netlify have vibrant communities of developers, engineers, and support staff who actively engage with users through forums, chat channels, and community events. These communities provide valuable resources, share knowledge, and learn best practices for deploying and managing web applications on their respective platforms. Both platforms offer dedicated support channels and documentation to assist developers with troubleshooting and optimization, ensuring that they have the resources they need to succeed.

12.2 Deployment process

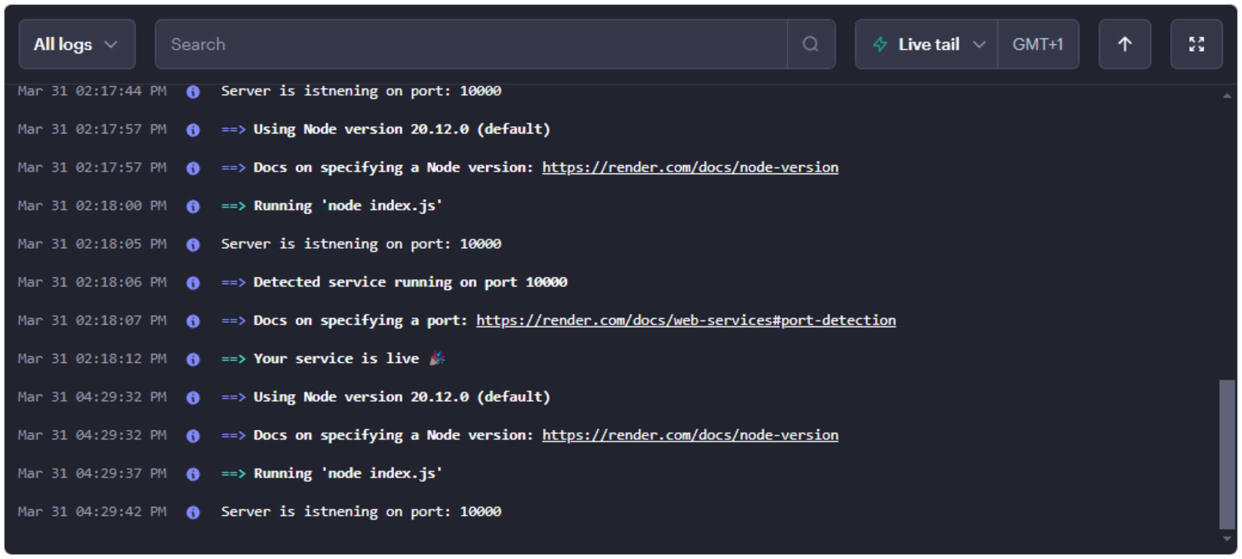
Deployment was achieved by creating a free account on render and Netlify. Connecting the websites GitHub repository. After successfully creating free accounts and connecting the repository, environment variables were added. As it can be seen below.



CLIENT_URL	[redacted]
GOOGLE_CLIENT_I	[redacted]
MONGO_CONNECT	[redacted]
JWT_SECRET	[redacted]
STRIPE_PL_KEY	[redacted]
STRIPE_SC_KEY	[redacted]

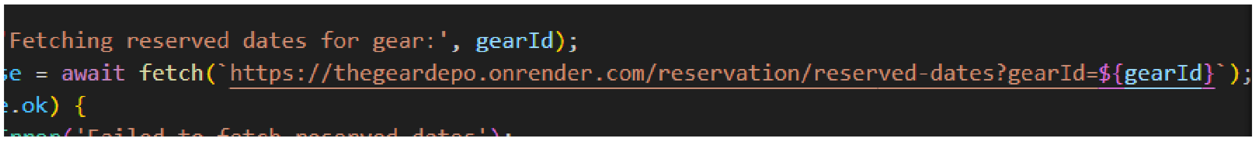
Figure 12-1 Graphic

After filling all the necessary fields and environment variables, the application's build was initiated and console appeared as seen below. After a successful build the website was live.



```
Mar 31 02:17:44 PM   Server is istnening on port: 10000
Mar 31 02:17:57 PM   ==> Using Node version 20.12.0 (default)
Mar 31 02:17:57 PM   ==> Docs on specifying a Node version: https://render.com/docs/node-version
Mar 31 02:18:00 PM   ==> Running 'node index.js'
Mar 31 02:18:05 PM   Server is istnening on port: 10000
Mar 31 02:18:06 PM   ==> Detected service running on port 10000
Mar 31 02:18:07 PM   ==> Docs on specifying a port: https://render.com/docs/web-services#port-detection
Mar 31 02:18:12 PM   ==> Your service is live 🎉
Mar 31 04:29:32 PM   ==> Using Node version 20.12.0 (default)
Mar 31 04:29:32 PM   ==> Docs on specifying a Node version: https://render.com/docs/node-version
Mar 31 04:29:37 PM   ==> Running 'node index.js'
Mar 31 04:29:42 PM   Server is istnening on port: 10000
```

Figure 13-2 Graphic



```
Fetching reserved dates for gear:', gearId);
e = await fetch(`https://thegeardepo.onrender.com/reservation/reserved-dates?gearId=${gearId}`);
.ok) {
rror('Failed to fetch reserved dates');
```

Changed all the local host URL mentions to the backend link, in order to link the frontend of the application Netlify to the backend end on render.

13 User Documentation:

13.1 Homepage

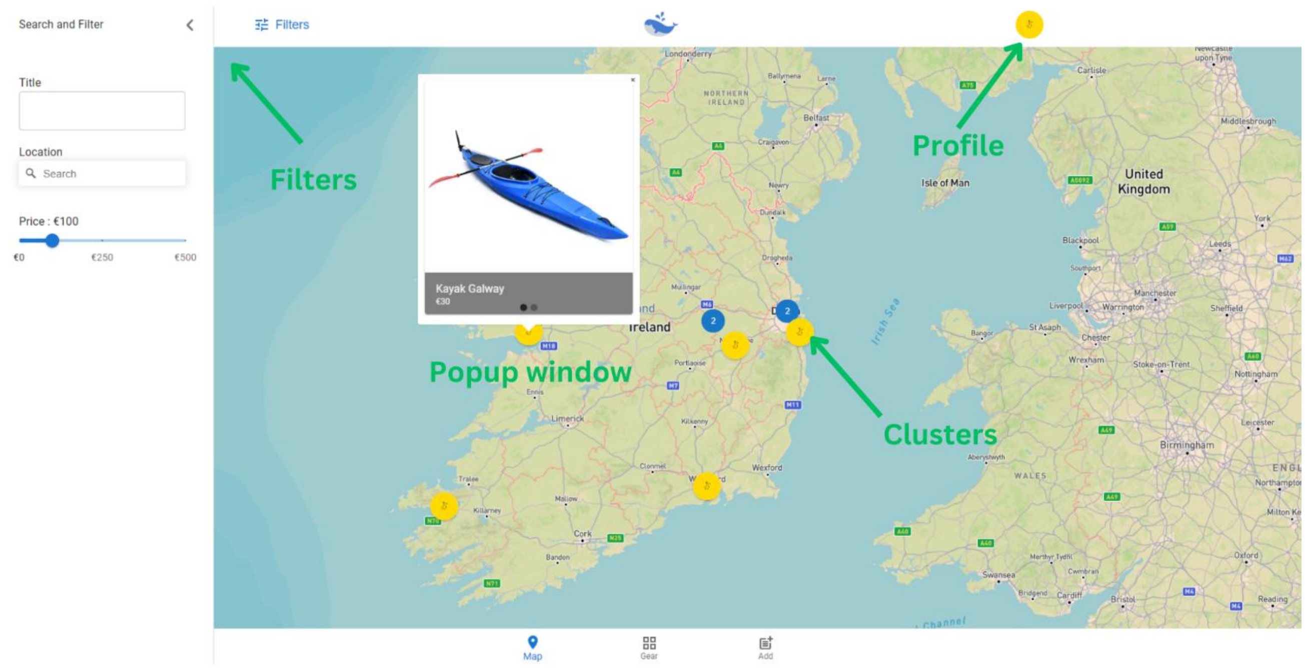


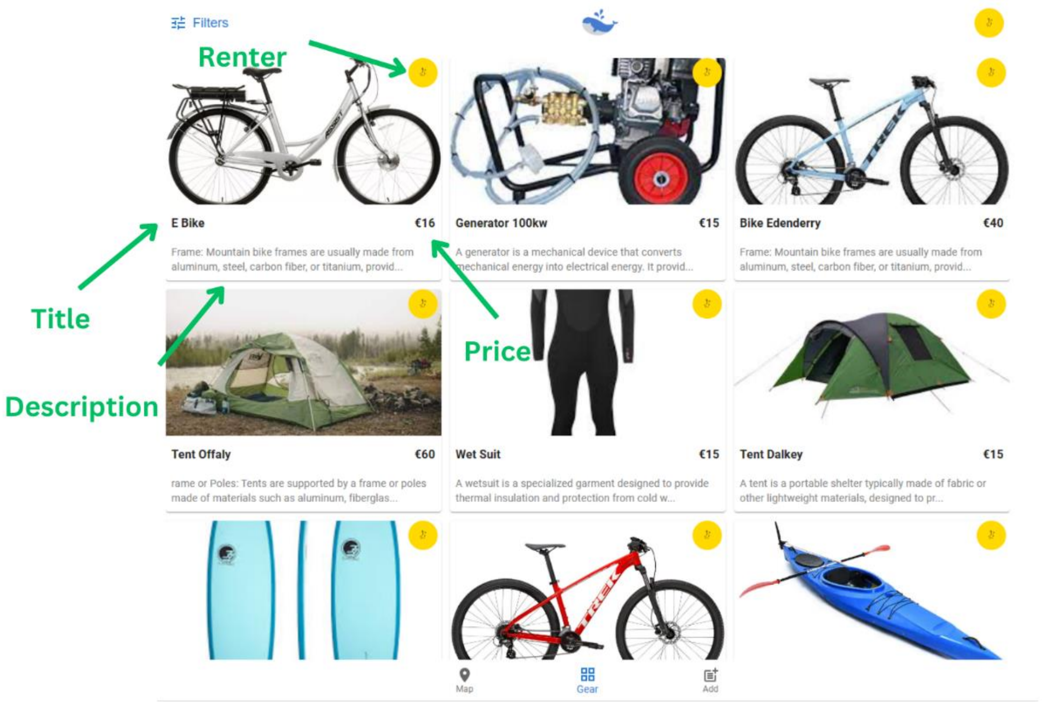
Figure 14-1 Graphic

Home screen contains an interactive map where a user can navigate around the world looking for their specific gear. Easily zooming in and out with fast rendering. When clicking on a gear a pop-up window is displayed with a picture, title, and price of selected gear.

On the right hand-side located are the filtering options, gears can be filtered by title, location, and price.

On the bottom of the website is the bottom navigation where a user can toggle between display preferences and add a listing.

13.2 Gears page



E Bike €16
Frame: Mountain bike frames are usually made from aluminum, steel, carbon fiber, or titanium, provid...

Generator 100kw €15
A generator is a mechanical device that converts mechanical energy into electrical energy. It provid...

Bike Edenderry €40
Frame: Mountain bike frames are usually made from aluminum, steel, carbon fiber, or titanium, provid...

Tent Offaly €60
rame or Poles: Tents are supported by a frame or poles made of materials such as aluminum, fiberglas...

Wet Suit €15
A wetsuit is a specialized garment designed to provide thermal insulation and protection from cold w...

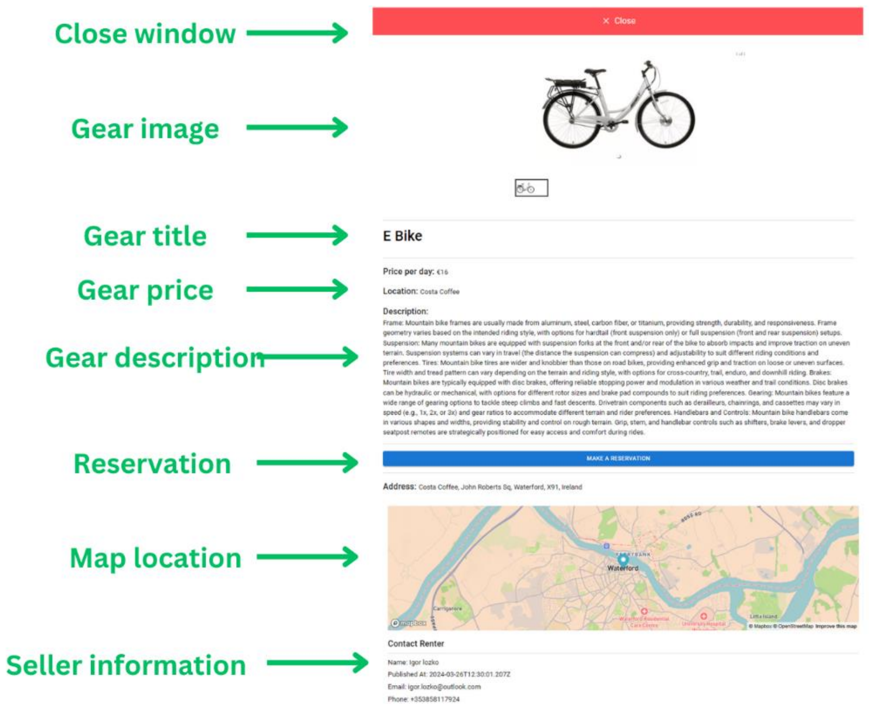
Tent Dalkey €15
A tent is a portable shelter typically made of fabric or other lightweight materials, designed to pr...

Figure 15-2 Graphic

When the Gear page is selected a user is greeted with a grid of gears listed on the website, user can filter through the gears to find their desired gear.

The layout of the page consists of a grid layout with each box containing the image, title, description, and price of the gear.

13.3 Selected gear page



E Bike

Price per day: €16

Location: Costa Coffee

Description:
Frame: Mountain bike frames are usually made from aluminum, steel, carbon fiber, or titanium, providing strength, durability, and responsiveness. Frame geometry varies based on the intended riding style, with options for hardtail (front suspension only) or full suspension (front and rear suspension) setups. Suspension: Many mountain bikes are equipped with suspension forks at the front and/or rear of the bike to absorb impacts and improve traction on uneven terrain. Suspension systems can vary in travel (the distance the suspension can compress) and adjustability to suit different riding conditions and preferences. Tires: Mountain bike tires are wider and knobbier than those on road bikes, providing enhanced grip and traction on loose or uneven surfaces. Tire width and tread pattern can vary depending on the terrain and riding style, with options for cross-country, trail, enduro, and downhill riding. Brakes: Mountain bikes are typically equipped with disc brakes, offering reliable stopping power and modulation in various weather and trail conditions. Disc brakes can be hydraulic or mechanical, with options for different rotor sizes and brake pad compounds to suit riding preferences. Gearing: Mountain bikes feature a wide range of gearing options to tackle steep climbs and fast descents. Drivetrain components such as derailleurs, chainrings, and cassettes may vary in speed (e.g., 1x, 2x, or 3x) and gear ratios to accommodate different terrain and rider preferences. Handlebars and Controls: Mountain bike handlebars come in various shapes and widths, providing stability and control on rough terrain. Grip, stem, and handlebar controls such as shifters, brake levers, and dropper seatpost remotes are strategically positioned for easy access and comfort during rides.

Address: Costa Coffee, John Roberts Sq, Waterford, X91, Ireland

Contact Renter

Name: Igor lozko
Published At: 2024-03-26T12:30:01.207Z
Email: igor.lozko@outlook.com
Phone: +353858117924

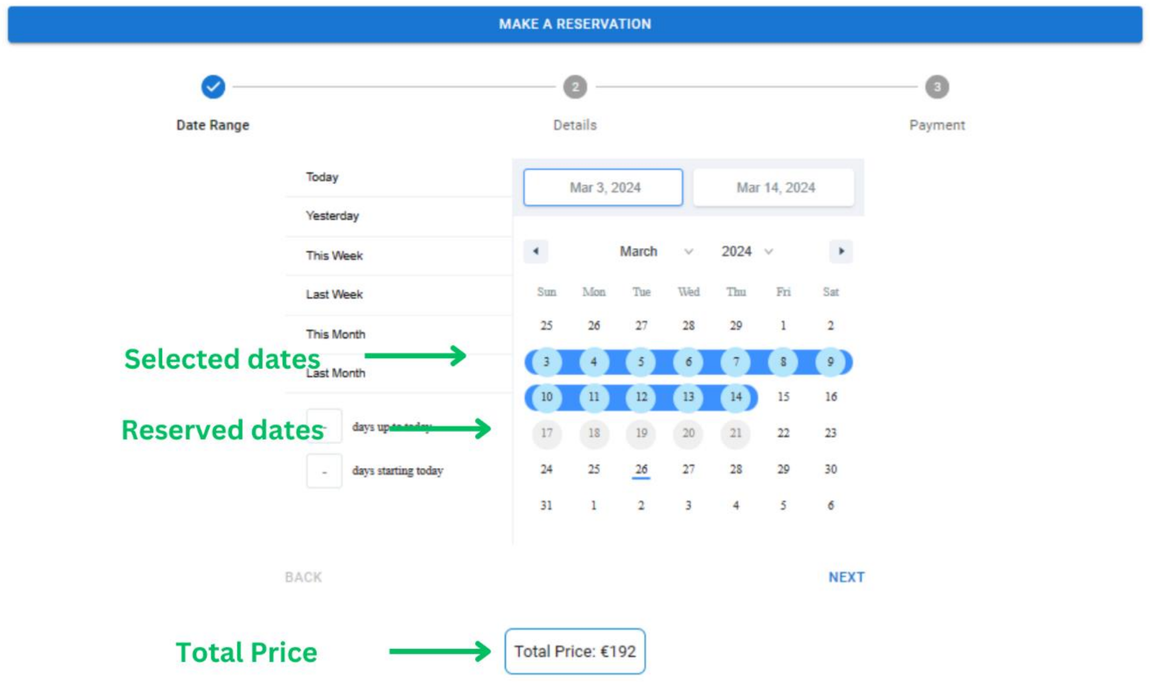
Figure 16-3 Graphic

This is the main gear page, here all the necessary information for the user is displayed. The gear page can be easily and quickly closed by clicking the red close bar. Gear image displays a carousel of gear images. Gear title, price. location and description are available to view by the user.

A reservation can be made by selecting the button make a reservation. Once selected a reservation model is triggered and opens with a calendar.

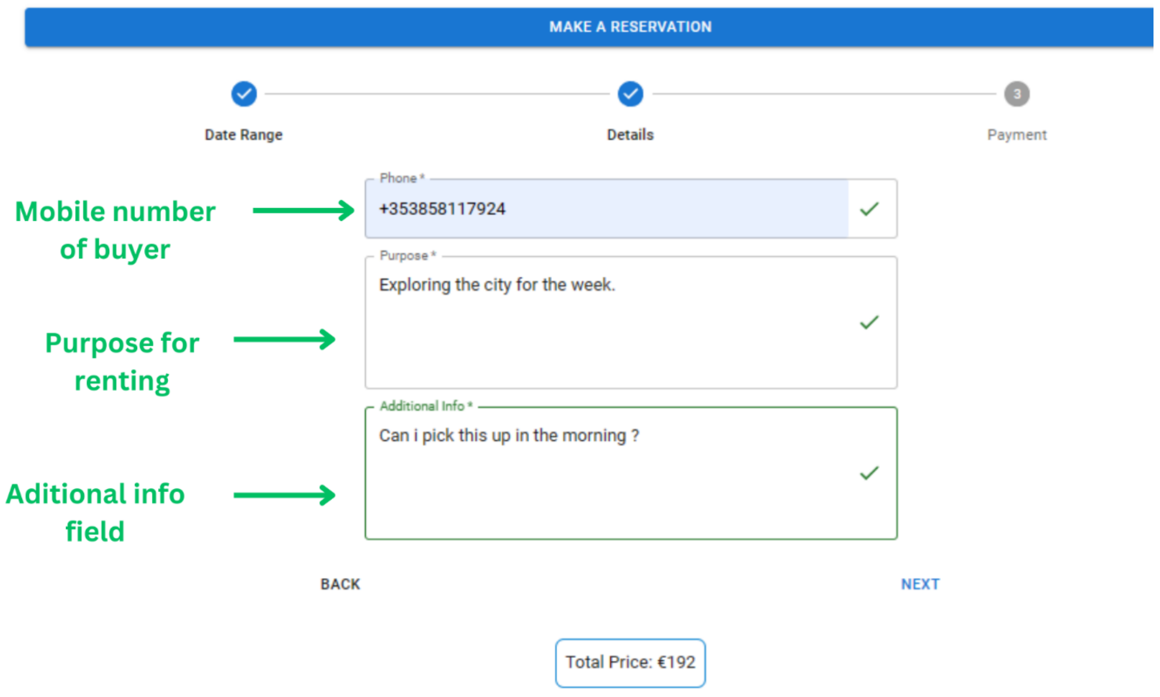
Below the reservation model is the map location of the gear, with sellers' information and contact details.

13.4 Reservation



Mar 3, 2024 – Mar 14, 2024

Total Price: €192



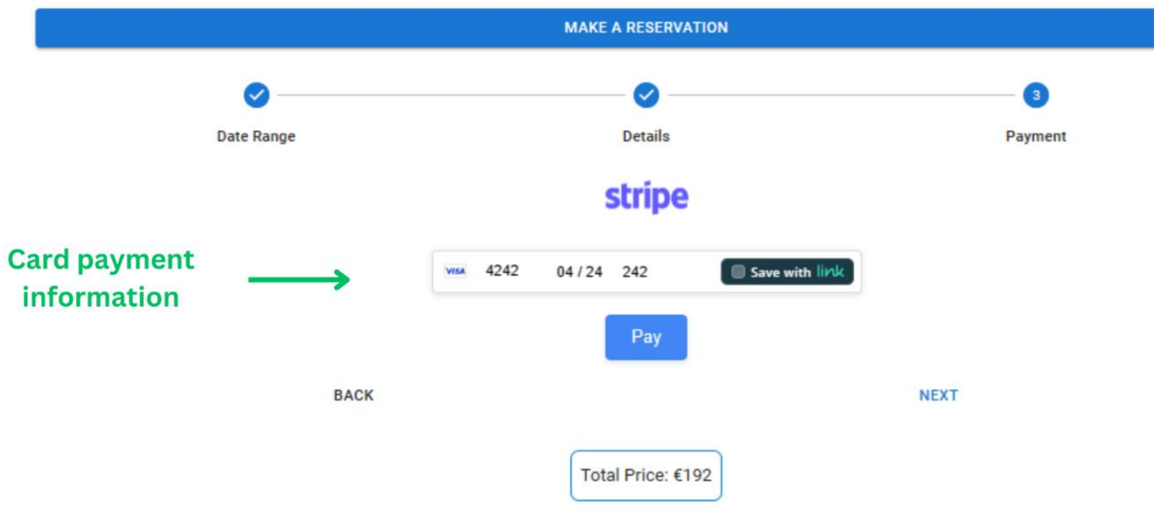
Phone *: +353858117924

Purpose *: Exploring the city for the week.

Additional Info *: Can i pick this up in the morning ?

Total Price: €192

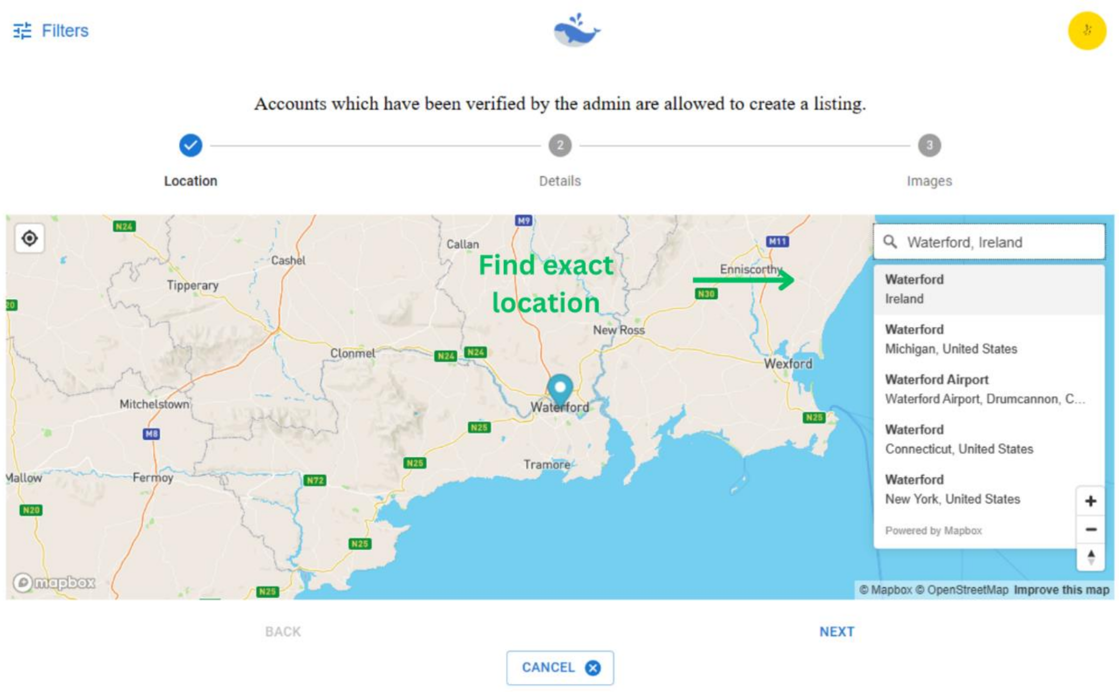
Figure 17-4 Graphic



Once the user toggles the button “Make a reservation”, the reservation model opens displaying 3 stages, 1 – Select dates, 2- Enter details, 3- payment.

1. **Stage 1:** User selects the dates they want to reserve the gear for, note that the dates in grey are already reserved dates and are not selectable. User can select the date range by clicking on the start and end date.
2. **Stage 2:** Once the dates are selected, user enters their phone number, purpose of reservation and additional info which will be displayed on the reservation for the renter.
3. **Stage 3:** Last stage if the payment a user enters their card information and clicks the pay button if the payment is successful, and reservation is made and logged on the database.

13.5 Listing creation



Accounts which have been verified by the admin are allowed to create a listing.

Waterford, Ireland

Waterford
Ireland

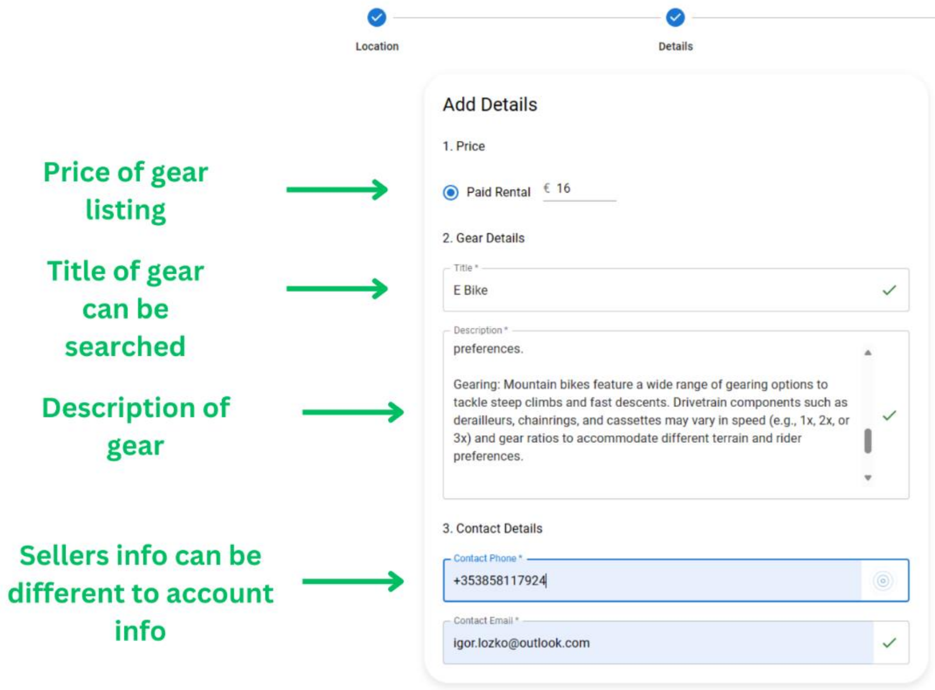
Waterford
Michigan, United States

Waterford Airport
Waterford Airport, Drumcannon, C...

Waterford
Connecticut, United States

Waterford
New York, United States

Powered by Mapbox


Location
Details
Images
Price of gear listing
Title of gear can be searched
Description of gear
Sellers info can be different to account info

Add Details

1. Price

Paid Rental € 16

2. Gear Details

Title *
E Bike

Description *
preferences.

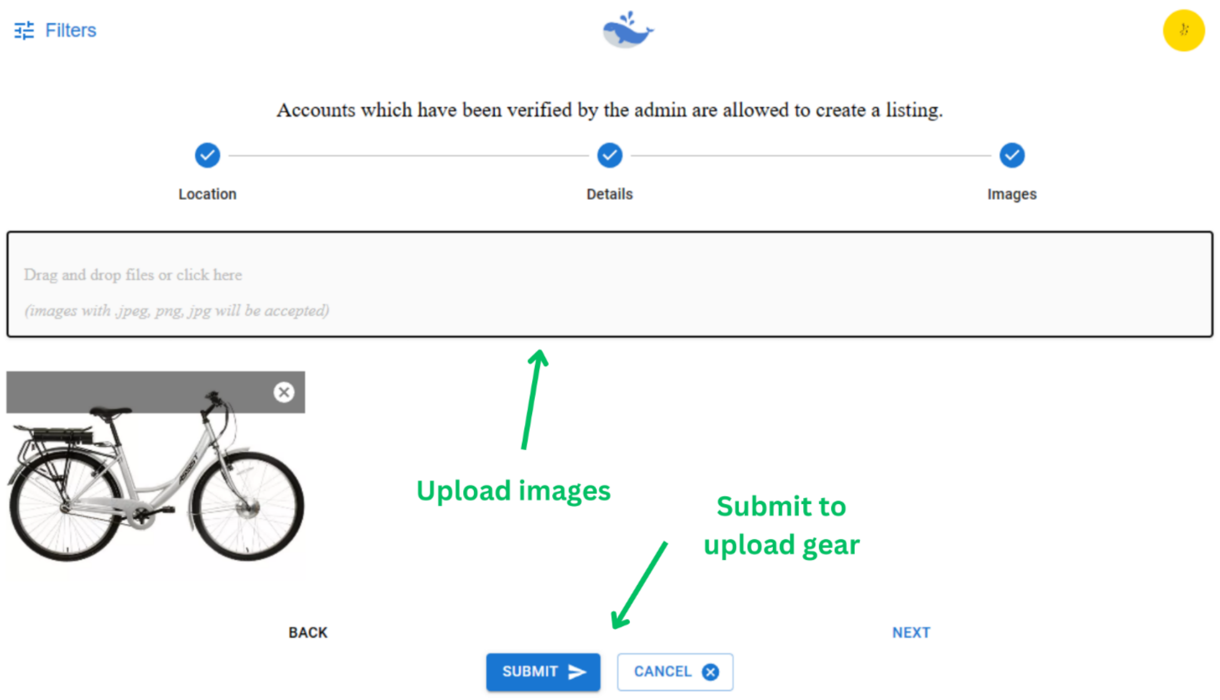
Gearing: Mountain bikes feature a wide range of gearing options to tackle steep climbs and fast descents. Drivetrain components such as derailleurs, chainrings, and cassettes may vary in speed (e.g., 1x, 2x, or 3x) and gear ratios to accommodate different terrain and rider preferences.

3. Contact Details

Contact Phone *
+353858117924

Contact Email *
igor.lozko@outlook.com

Figure 18-5 Graphic


Filters
Accounts which have been verified by the admin are allowed to create a listing.
Location
Details
Images
Drag and drop files or click here
(images with .jpeg, png, jpg will be accepted)
Upload images
Submit to upload gear
BACK
NEXT
SUBMIT
CANCEL

Only verified user can pots an add on the website, Users who are able to post have been verified by the admin. Listing creation is a 3-step process.

1. **Adding location:** Seller can search the map for their exact location by entering their town, region or Eircode into the search bar on the top right-hand corner.
2. **Entering gear/ seller details:** Once the location has been selected, Seller must enter the price, title, description, phone number and email address into a form.
3. **Uploading images:** Last stage in listing creation is image upload a seller can select from images on their device or dragging and dropping images, wait until the images have been uploaded and then press submit to create a listing.

14 Evaluation:

14.1 Performance:

The performance tests involved measuring response times for user interactions, such as searching for gear, viewing gear listings, and making reservations.

Results from performance testing indicated that the website was meeting performance checks, with response times well within the aims. Even under peak load scenarios, the application-maintained responsiveness, ensuring a smooth user experience.

System monitoring tools were utilized to track resource utilization metrics, including CPU usage, memory consumption, and network bandwidth. The monitoring data revealed efficient resource management, with no significant spikes or bottlenecks observed during testing.

14.2 Usability:

Users who consisted of friends and students were asked to perform common tasks, such as searching for gear, making reservations, and updating user profiles.

Feedback from usability testing was overwhelmingly positive, with users praising the intuitive layout, clear navigation paths, and logical organization of content. Participants reported minimal friction in completing tasks and expressed satisfaction with the overall usability of the application.

14.3 Effectiveness:

User feedback surveys and interviews were conducted to gather insights into the effectiveness of the web application in facilitating gear rentals and transactions. People were asked about their satisfaction with the rental process, the quality of gear listings, and the reliability of transactional features.

Users praised the seamless browsing experience, the comprehensive gear listings, and the convenience of the reservation and payment process.

The integration of interactive maps, detailed gear descriptions, and seamless UI was found to enhance the effectiveness of the application in helping users find and select the right gear for their needs. Users reported feeling confident and informed when making a rental, one improvement was brought up which was not implemented due to the timeline constraints which was email confirmation when a booking was made. Overall positive experience. The filtering feature proved to be popular with users being able to find what they were looking for faster and easily.

In conclusion, user feedback and testing proved to be beneficial and supporting. The positive reception from users shows the project in delivering a valuable and user-centric solution for accessing outdoor equipment. Ongoing monitoring efforts will continue to enhance the application's performance, usability, and effectiveness.

15 Ethics

Ethical considerations play a crucial role in ensuring the responsible and ethical use of technology. It is important to be mindful of the potential ethical implications of projects and the broader impact it may have.

Privacy and Data Protection:

User privacy to ensure secure handling of personal data. Implementing robust security measures to protect against unauthorized access, breaches, and data leaks. For example, using JWT tokens, checking token expiration by comparing the dates is vital for ensuring user protection and applications security.

Transparency and Accountability:

Inside the application is the terms and conditions link, this is a provisional link and will change. Provide clear and accessible privacy policies and terms of service to users. Take responsibility for the actions and consequences of software and be accountable for any ethical lapses or violations. By having the user on the website, they agree to follow terms and conditions.

Accessibility and Usability:

Ensuring that software is accessible to all users, for example this website was developed with aged people in mind or someone who isn’t too comfortable with using technology, a seamless and easy to use system with readable text and simple navigation.

Sustainability and Environmental Impact:

Optimizing code efficiency, minimize waste, and adopt sustainable development practices to reduce environmental harm.

Intellectual Property Rights:

Respecting intellectual property rights and adhere to copyright, patent, and licensing regulations when using third-party libraries, frameworks, or open-source software in your projects. Give proper attribution to original creators and obtain necessary permissions for using copyrighted material.

16 Conclusion

In conclusion, the outcome of this project is the successful development of a web-based platform for renting outdoor gear and equipment. Using the MERN (MongoDB, Express.js, React.js, Node.js) stack along with Stripe and Map box APIs, the website provides users with a seamless and intuitive experience for browsing, renting, and managing outdoor gear rentals.

Through extensive planning, research, and implementation phases, the project has resulted in a somewhat functional state that demonstrates some of the key features such as interactive mapping, gear listings, user authentication, reservation management, and secure payment processing.

Moving forward, there are several opportunities for further development and enhancement of the website. Future improvements and enhancements could focus on expanding the range of available gear, implementing advanced search and filtering options with different categories, and integrating social features for community engagement. A gear review feature could be a beneficial addition to the website.

Overall, this project represents a significant achievement, offering a valuable solution for outdoor enthusiasts and businesses alike. With continued refinement and innovation, the platform has the potential to become a leading destination for convenient and sustainable outdoor gear rentals in the digital age.

17 References

[2
] [Online].

[3
] MakeSigns, "Scientic Posters Tutorial," [Online]. Available: https://www.makesigns.com/tutorials/scientific-poster-parts.aspx. [Accessed 09 02 2021].

[5] Outdoor Industry Association. (Year). "Annual Growth Rate of Gear Rental Sector." [Online]. Available: [insert URL or source of information]. [Accessed Date].

[6] GearTrade. (Year). "Survey on Online Presence of Rental Businesses." [Online]. Available: [insert URL or source of information]. [Accessed Date].

[7] The North Face. (Year). "Study on Underutilization of Outdoor Gear by Owners." [Online]. Available: [insert URL or source of information]. [Accessed Date].

[8] REI Co-op. (Year). "Comparative Analysis of Cost Savings through Gear Rental." [Online]. Available: [insert URL or source of information]. [Accessed Date].]

[9] Dave Gray Mern Stack tutorial – For the informative content. MERN Stack Full Tutorial & Project | Complete All-in-One Course | 8 Hours (youtube.com)

[10] Programming with Mosh – Information about node technology. Node.js Tutorial for Beginners: Learn Node in 1 Hour (youtube.com)

[11] Chris Blakely – Mern stack project Hotel booking app - Complete MERN Stack Project: Build a Hotel Booking App Like a Pro Developer Step-by-Step Course 2024 (youtube.com)

[11] Coding with Dawid - Build a Fullstack Booking App using MERN (mongo, express, react, node) (youtube.com)

[12]Lama Dev - React Node.js Booking App Full Tutorial | MERN Stack Reservation App (JWT, Cookies, Context API) (youtube.com)

[13] Codeevolution – Providing MUI tutorials. React Material UI Tutorial - 1 - Introduction (youtube.com)

[14] Stripe Developers - Authentication with stripe-node (youtube.com)

[15] Web Dev Simplified - https://www.youtube.com/watch?v=mI1tbIXQI&t=966s&pp=ygUPU3RyaXBlIGFwaSBub2Rl

[16] Darwin Tech - https://www.youtube.com/watch?v=XKWJFpZYVAQ&t=526s&pp=ygUPU3RyaXBlIGFwaSBub2Rl

[17] Lama Dev – Mapbox tutorial React Node Travel Map App | MERN Stack Full Course Using Mapbox and React Hooks (youtube.com)

[18] Code like a Pro – Map and layout. React MERN Project with Node, Express, Mongodb, Firebase, Mapbox. couch surfing or renting rooms (youtube.com)

[19] Code with projects - Getting Started with JWT | MERN Authentication & Authorization | Node JS, React & MongoDB | Part 1 (youtube.com)

[20] Mehul - Webflow + Attentive 60Sec cc (youtube.com)

[21] Projects with Saha - Full-Stack MERN Auth project: Build & Deploy (Reactjs, json web token, jwt, redux toolkit, cookie) (youtube.com)

[22]Code with Antonio - Full Stack Airbnb Clone with Next.js 13 App Router: React, Tailwind, Prisma, MongoDB, NextAuth 2023 (youtube.com)