ParkMe | MU

Interim Report

Student: Redon Ferizi (21358591) (Redon.ferizi.2022@mumail.ie)

Supervisor: Dr Aidan Mooney

Overall Project Objectives

The primary objective of this project is to develop "ParkMe | MU," a comprehensive web application designed to streamline parking management at Maynooth University. The app aims to:

- Provide **real-time parking availability** across campus.
- Facilitate **easy check-ins and check-outs** for users.
- Offer **navigation assistance** to available parking spots.
- Deliver **analytics and insights** to help users plan their commute.
- Enhance the overall parking experience for **students**, **staff**, and **visitors**.
- Reduce parking congestion and improve the efficiency of campus parking

Table of Contents

Description of Work Completed

Description of Work Completed

Substantial progress has been achieved during the initial project phases, following the set schedule and objectives. Key milestones include establishing the overall framework, setting up both the frontend and backend infrastructure, and developing core features that bring real-time functionality and user interactivity.

Evidence of Work Completed and Issues Encountered

Requirements Gathering:

- ***Completed:*** Created a requirements document detailing functional and non-functional requirements to guide development.
- ***Issues Encountered:*** Initially struggled to capture all necessary requirements, as the complexity of real-time parking data required input from potential users and stakeholders. This feedback took time to gather, and adjusting requirements to meet practical expectations while keeping the scope realistic was challenging.

System Design:

- ***Completed:*** Developed system diagrams outlining the architecture of frontend and backend components and created user interface wireframes in Figma to illustrate layout and design (see Appendix B).
- ***Issues Encountered:*** Designing a seamless user interface that balanced functionality with simplicity proved challenging. Translating the wireframes into interactive features required numerous revisions to avoid clutter and keep the layout user-friendly. Additionally, coordinating frontend and backend integration was complex, requiring multiple design iterations to ensure data flowed smoothly across the system.

Frontend Development:

- ***Completed****:* Built the user interface using HTML5, CSS3, and JavaScript. Implemented responsive design using Bootstrap 5, ensuring compatibility with various devices. This codebase contains approximately 4,500 lines across HTML, CSS, and JavaScript files (see Appendix A).
- ***Issues Encountered:*** Integrating Bootstrap effectively while maintaining unique design elements presented some styling challenges. Consistency across different screen sizes was difficult to maintain, and managing CSS conflicts (especially with the interactive map) required extensive debugging to keep the layout intact.

Backend Development:

- ***Completed:*** Set up a server using Node.js and Express for efficient request handling and developed RESTful APIs to manage parking data retrieval, check-ins, check-outs, and real-time updates (see Appendix B).
- ***Issues Encountered:*** Initially struggled with configuring Express routing and managing API requests effectively, particularly when handling real-time data updates. MongoDB connection errors and port configuration issues also slowed development, as debugging these errors required familiarity with server environments and fine-tuning configurations for stability.

Database Setup:

- ***Completed:*** Configured a MongoDB database to store user data, parking spot information, and log management. Designed schemas for users, parking spots, and transactions, which helped standardize data handling across the system (see Appendix C).
- ***Issues Encountered:*** Designing efficient schemas was initially a challenge, especially when balancing flexibility with database efficiency. Implementing relationships between data types (such as users and parking spots) created complexity in database structure, leading to performance issues when querying large datasets. Debugging these issues involved optimizing queries and fine-tuning schema configurations.

Feature Implementation:

- Real-Time Availability: Integrated Chart.js for dynamic data visualization, providing real-time availability insights.
- Check-In/Check-Out System: Created a system to update parking availability based on user input.
- Interactive Map: Utilized Leaflet to allow users to view real-time parking availability on a campus map.
- Navigation Assistance: Integrated Google Maps API to guide users to available parking spots.
- User Feedback System: Added a feedback form to gather user feedback and drive improvements.

Issues Encountered*:* Integrating each of these features came with unique challenges:

- *Real-Time Availability:* Achieving accurate real-time updates involved frequent data refreshes, which impacted performance and required adjustments to API calls and Chart.js configuration.
- *Check-In/Check-Out System:* Implementing check-in and check-out logic was complex, as user input had to update availability without data conflicts, leading to extensive debugging.
- *Interactive Map:* Leaflet integration faced compatibility issues with CSS, which disrupted the layout on smaller screens. Aligning Leaflet's color-coded markers with accessibility requirements (e.g., colourblind-friendly colours) required multiple adjustments.

Literature review

Background and Motivation

Inspired by challenges in finding parking at Maynooth University, the project aims to improve parking management for students, staff, and visitors.

Prior Work in Web Development and Parking Systems

- **Internship at Electric Ireland:** Gained expertise in CMS and web systems, directly applicable to this project.
- **Academic Studies:** Web development modules provided skills in HTML, CSS, JavaScript, and modern frameworks.

Technical Resources and Inspirations

The project utilises several state-of-the-art technologies and practices. The following resources and prior work form the technical foundation of this project:

1. **Mapping Systems:** Leaflet and Google Maps API for dynamic maps and navigation.
2. **Real-Time Visualisation:** Chart.js for real-time data representation.
3. **Scalable Backend:** Node.js, Express, and MongoDB for real-time data management.
4. **Accessibility:** Designed for inclusivity with colourblind-friendly features and intuitive UI.

Overview of Learning Gained Through GenAI

Generative AI tools like **ChatGPT** and **GitHub Copilot** have been invaluable resources throughout the development of the ParkMe | MU project.

Understanding Best Practices in Web Development

- **Usage:** Leveraged GenAI to gain insights into industry-standard best practices for both frontend and backend development.

Exploring and Integrating New Technologies

- **Usage:** Utilised GenAI to explore and understand new technologies and APIs relevant to the project's requirements.

Enhancing UI/UX Design Skills

- **Usage:** Employed GenAI to improve user interface and user experience design by receiving suggestions on layout, colour schemes, and responsive design techniques.

Outline of Future Work

Complete Feature Implementation:

Finish developing features like push notifications, user history, and advanced analytics.

Testing Phase:

Conduct thorough unit, integration, and user acceptance testing.

Address any bugs or performance issues identified.

Deployment Preparation:

Optimise the application for deployment on a scalable hosting platform (e.g., Heroku or AWS).

Ensure security best practices are in place.

Comprehensive Documentation:

Develop user manuals, API documentation, and technical guides.

Final Presentation and Submission:

Prepare presentation materials, including slides and a live demo

Appendix - Additional Evidence

Appendix A: Screenshots

Figure 0: *landing page*

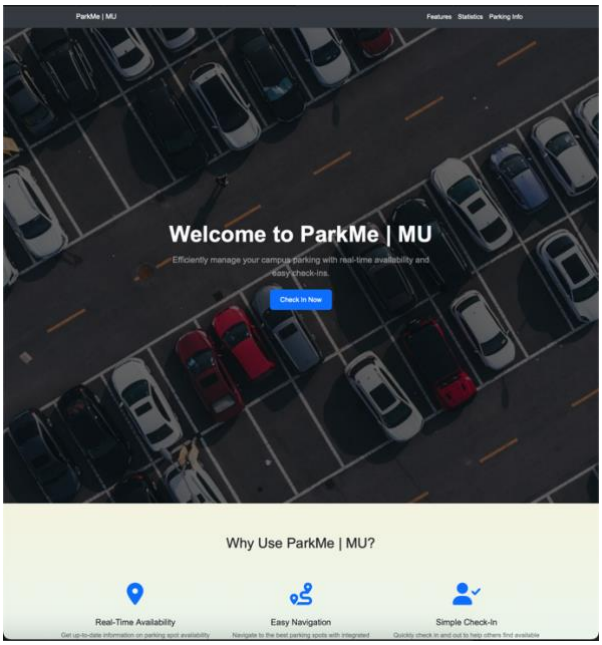


Figure 1: *Features under landing*

Figure 2: *Occupancy Graph under features*

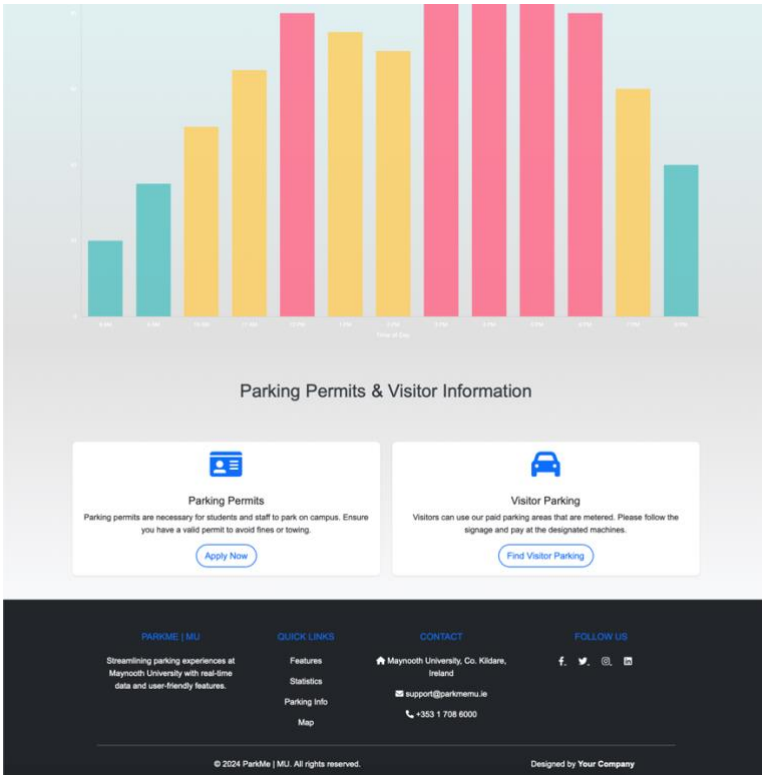


Figure 3: *Interactive App with Realtime data with legend*

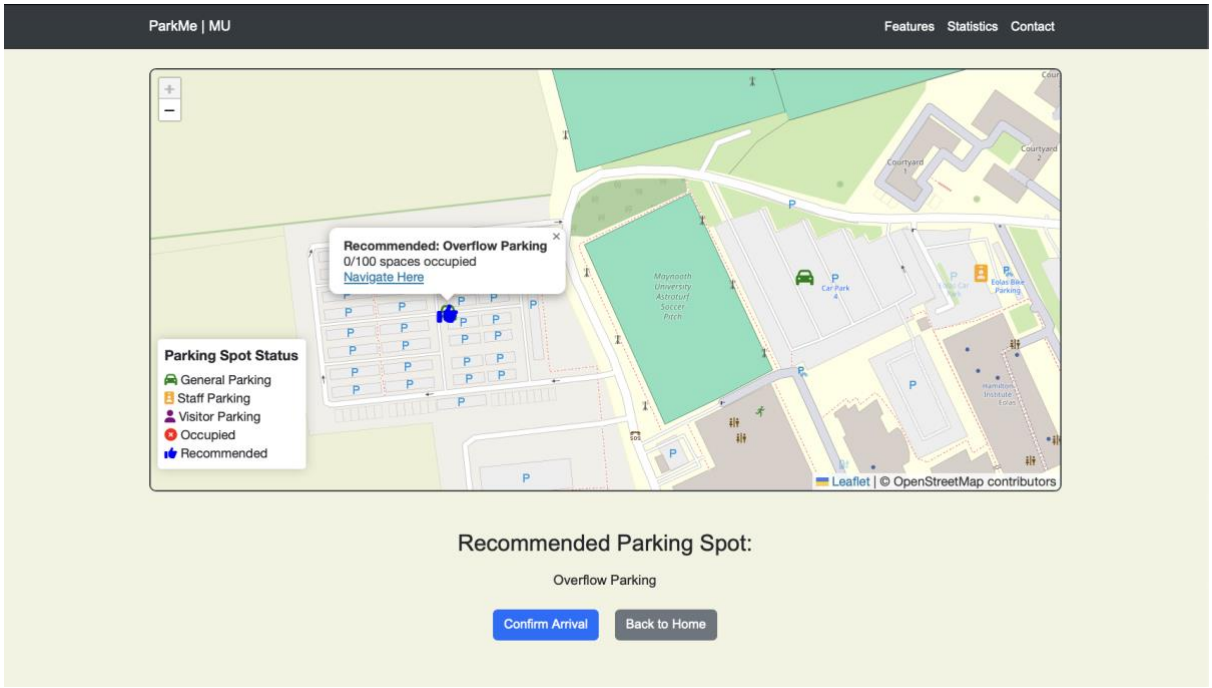


Figure 4: *Admin page*

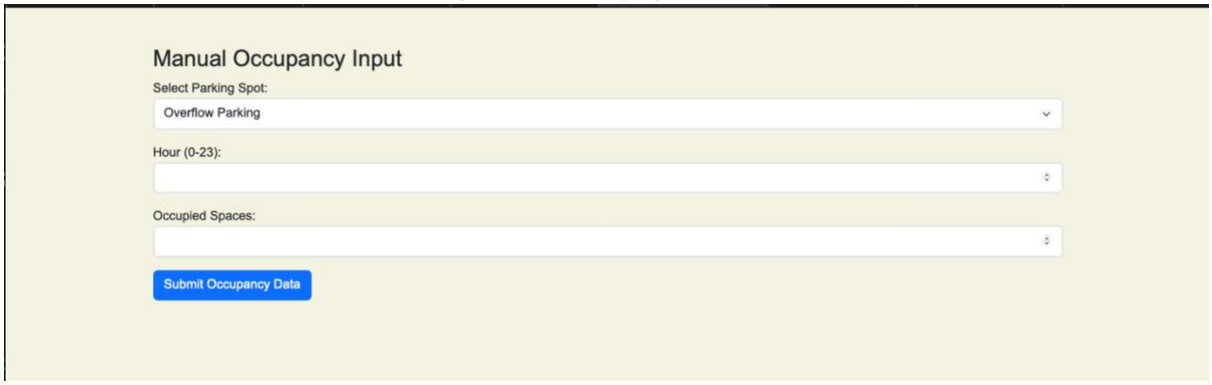
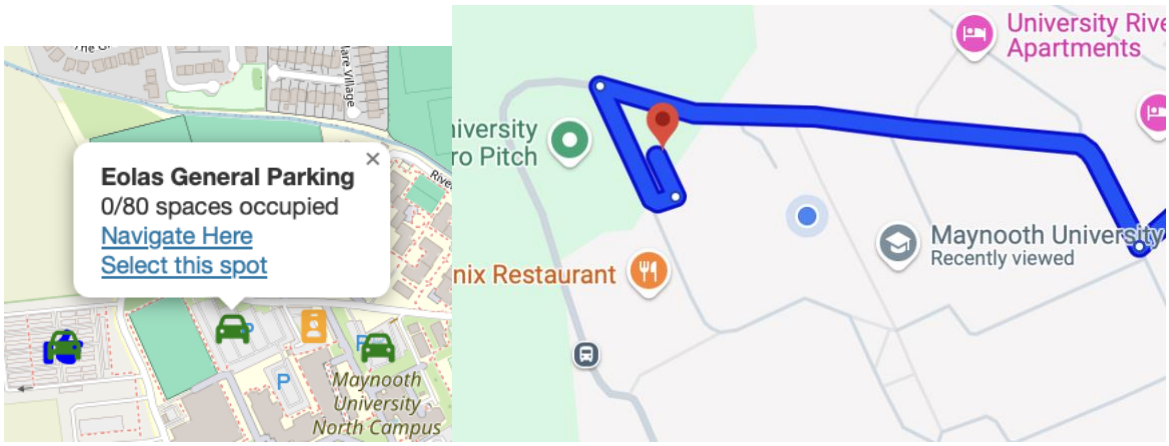
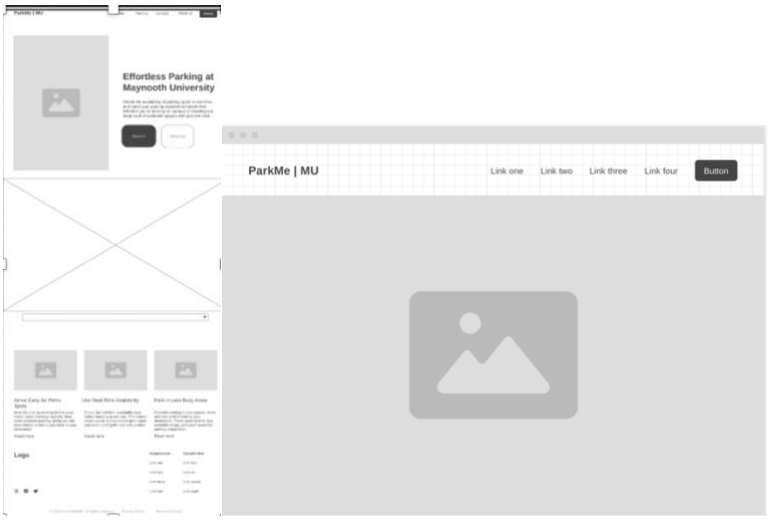


Figure 5: *Google maps integration with different icons*



Appendix B: Wireframe

Figure 6: *Initial Wireframes*



Appendix C: Backend

Figure 7: *MongoDB Dashboard*

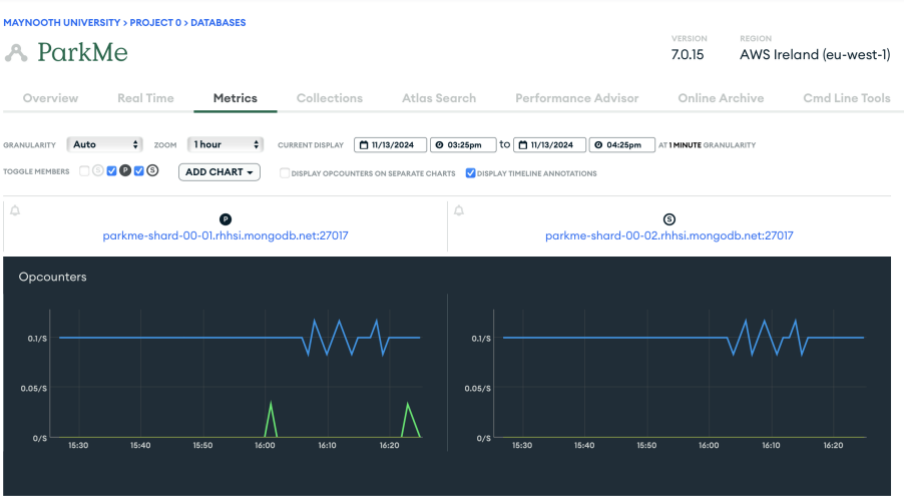


Figure 8: *MongoDB Collection Sample*

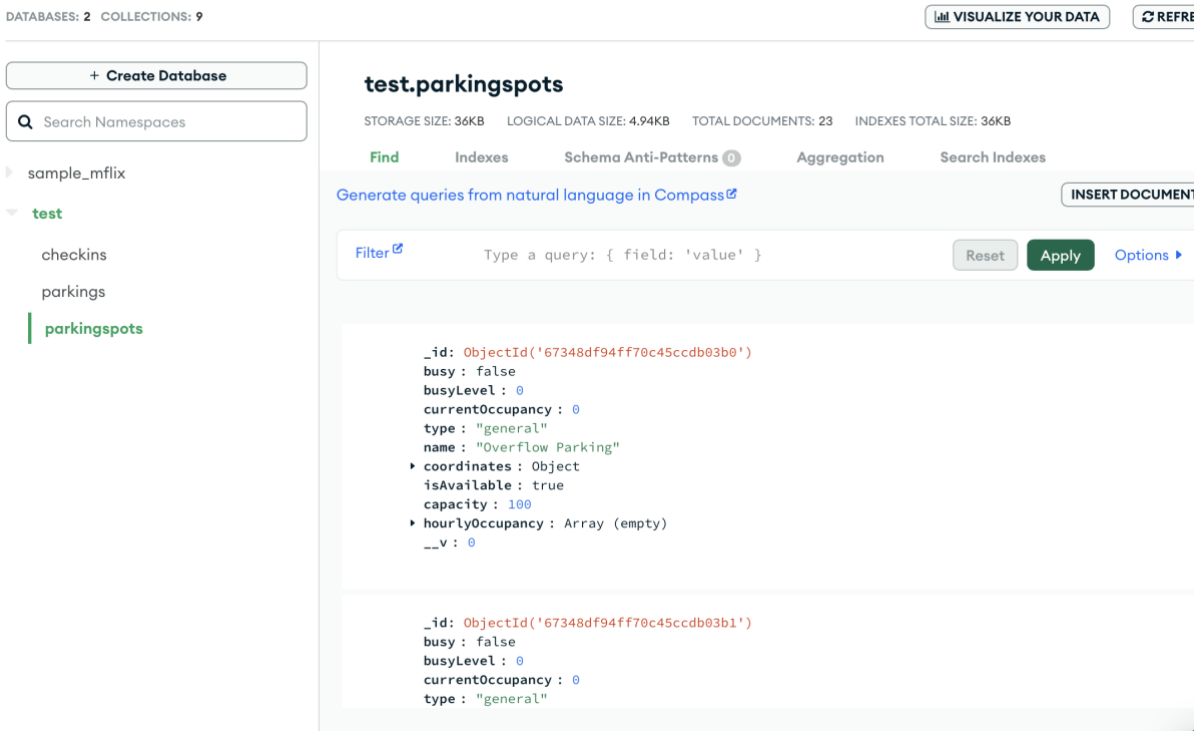
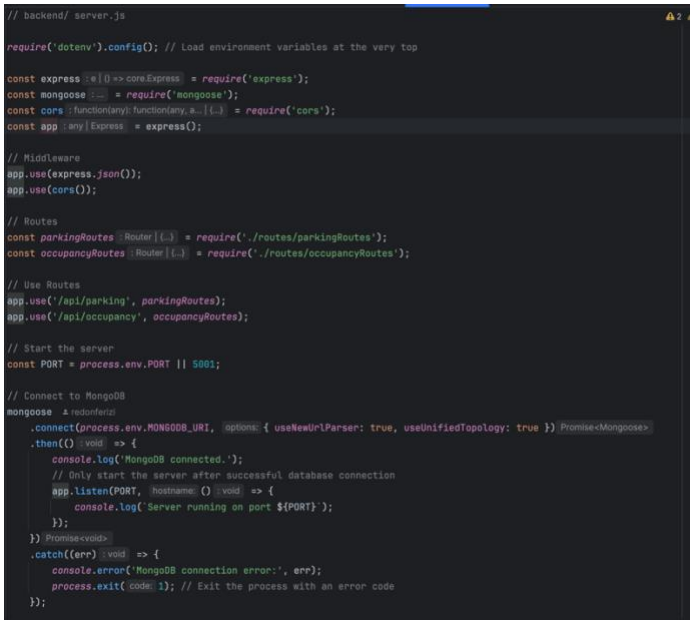
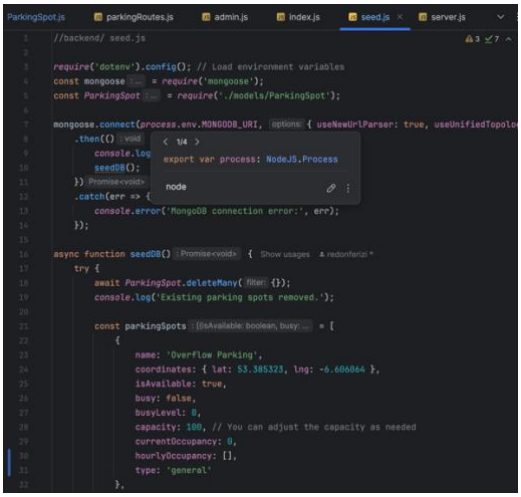


Figure 9: *Server.js Config*



```
// backend/ server.js

require('dotenv').config(); // Load environment variables at the very top

const express = require('express');
const mongoose = require('mongoose');
const cors = require('cors');
const app = express();

// Middleware
app.use(express.json());
app.use(cors());

// Routes
const parkingRoutes = require('./routes/parkingRoutes');
const occupancyRoutes = require('./routes/occupancyRoutes');

// Use Routes
app.use('/api/parking', parkingRoutes);
app.use('/api/occupancy', occupancyRoutes);

// Start the server
const PORT = process.env.PORT || 5001;

// Connect to MongoDB
mongoose
    .connect(process.env.MONGODB_URI, { useNewUrlParser: true, useUnifiedTopology: true })
    .then(() => {
        console.log('MongoDB connected.');
        // Only start the server after successful database connection
        app.listen(PORT, () => {
            console.log(`Server running on port ${PORT}`);
        });
    })
    .catch((err) => {
        console.error('MongoDB connection error:', err);
        process.exit(1); // Exit the process with an error code
    });
```

Figure 10: *Seed.js with parking spot data for mongoDB*

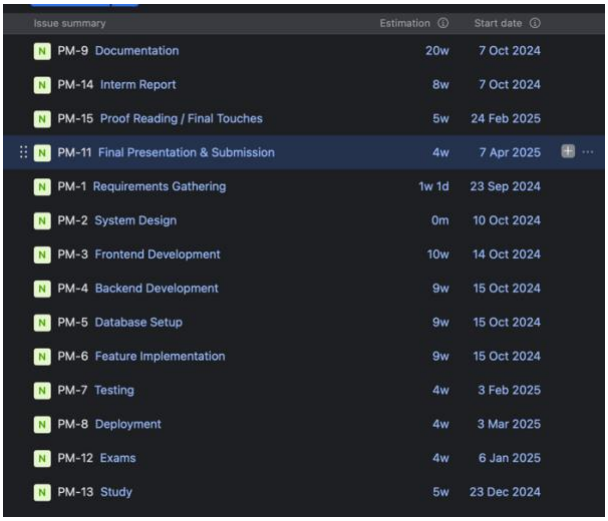


```
ParkingSpot.js | parkingRoutes.js | admin.js | index.js | seed.js | server.js

//backend/ seed.js

require('dotenv').config(); // Load environment variables
const mongoose = require('mongoose');
const ParkingSpot = require('./models/ParkingSpot');

mongoose.connect(process.env.MONGODB_URI, { useNewUrlParser: true, useUnifiedTopolo
    .then(() => {
        console.log
        seedDB();
    })
    .catch(err => {
        console.error('MongoDB connection error:', err);
    });

async function seedDB() {
    try {
        await ParkingSpot.deleteMany({});
        console.log('Existing parking spots removed.');

        const parkingSpots = [
            {
                name: 'Overflow Parking',
                coordinates: { lat: 53.385323, lng: -6.606064 },
                isAvailable: true,
                busy: false,
                busyLevel: 0,
                capacity: 100, // You can adjust the capacity as needed
                currentOccupancy: 0,
                hourlyOccupancy: [],
                type: 'general'
            },

export var process: NodeJS.Process
node
```

Appendix D: Gantt Chart

Figure 11: *YouTrack Gantt Chart*

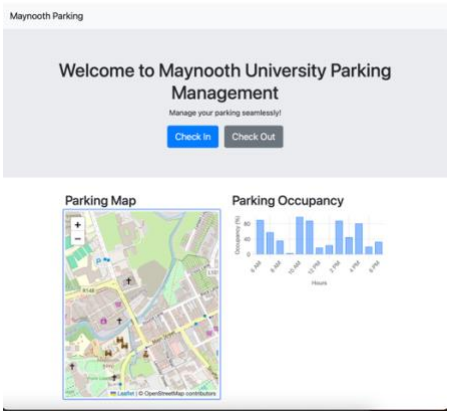
Figure 12: *You track gantt chart issues/items*



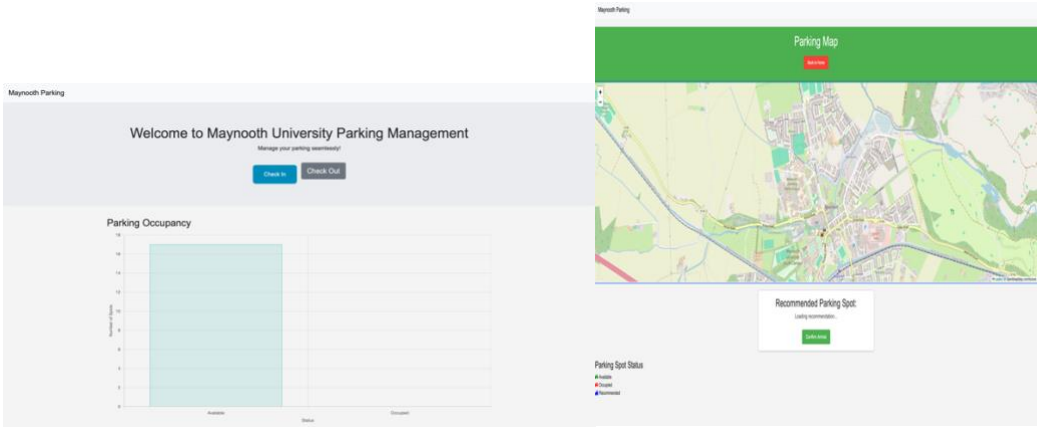
Issue summary	Estimation	Start date
PM-9 Documentation	20w	7 Oct 2024
PM-14 Interm Report	8w	7 Oct 2024
PM-15 Proof Reading / Final Touches	5w	24 Feb 2025
PM-11 Final Presentation & Submission	4w	7 Apr 2025
PM-1 Requirements Gathering	1w 1d	23 Sep 2024
PM-2 System Design	0m	10 Oct 2024
PM-3 Frontend Development	10w	14 Oct 2024
PM-4 Backend Development	9w	15 Oct 2024
PM-5 Database Setup	9w	15 Oct 2024
PM-6 Feature Implementation	9w	15 Oct 2024
PM-7 Testing	4w	3 Feb 2025
PM-8 Deployment	4w	3 Mar 2025
PM-12 Exams	4w	6 Jan 2025
PM-13 Study	5w	23 Dec 2024

Appendix E: UI Iterations

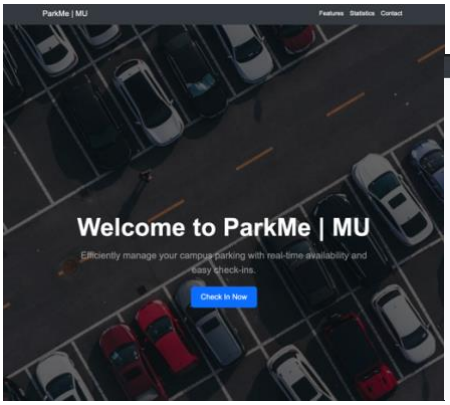
11th October

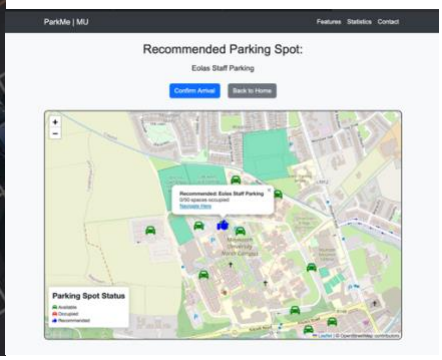


25th October



6th November





13th November