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

OVERALL PROJECT OBJECTIVES **2**

DESCRIPTION OF WORK COMPLETED **3**

DESCRIPTION OF WORK COMPLETED 3
EVIDENCE OF WORK COMPLETED AND ISSUES ENCOUNTERED 3
Requirements Gathering: *3*
System Design: *3*
Frontend Development: *3*
Backend Development: *4*
Database Setup: *4*
Feature Implementation: *4*

LITERATURE REVIEW **5**

BACKGROUND AND MOTIVATION 5
PRIOR WORK IN WEB DEVELOPMENT AND PARKING SYSTEMS 5
TECHNICAL RESOURCES AND INSPIRATIONS 5
CHALLENGES IN PRIOR WORK AND THEIR APPLICATION 6

OVERVIEW OF LEARNING GAINED THROUGH GENAI **6**

Understanding Best Practices in Web Development *6*
Exploring and Integrating New Technologies *6*
Enhancing UI/UX Design Skills *6*

OUTLINE OF FUTURE WORK **7**

Complete Feature Implementation: *7*
Testing Phase: *7*
Deployment Preparation: *7*
Comprehensive Documentation: *7*
Final Presentation and Submission: *7*

APPENDIX - ADDITIONAL EVIDENCE **8**

APPENDIX A: SCREENSHOTS 8
APPENDIX B: WIREFRAME 10
APPENDIX C: BACKEND 11
APPENDIX D: GANTT CHART 12
APPENDIX E: UI ITERATIONS 13

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

The **ParkMe | MU** project is inspired by personal and professional experiences that highlight the need for efficient parking systems. As a student who drives to Maynooth University, the challenges of securing parking spaces directly affect my daily routine. The difficulty in finding parking on campus motivated the idea of creating a solution to improve parking management, making it more accessible for students, staff, and visitors.

Prior Work in Web Development and Parking Systems

During my internship at **Electric Ireland**, I gained valuable experience working with content management systems (CMS) and their practical applications. My responsibilities included generating ideas, updating content for the new Electric Ireland website, making changes to their Switch site, and conducting User Acceptance Testing (UAT) for various designs. This experience deepened my understanding of web systems and design from both user and developer perspectives. These skills are directly applicable to the development of the ParkMe | MU web application.

In my second year at Maynooth University, I completed a **Web Development module** that provided a foundation in HTML, CSS, and JavaScript. This module introduced me to modern development frameworks, best practices, and debugging techniques, all of which have been invaluable in the creation of the ParkMe | MU application.

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. **Interactive Mapping Systems:** Tools like Leaflet and the Google Maps API have inspired the integration of dynamic maps to enable users to view parking availability and navigate to suggested parking spots. These technologies are widely used in urban planning and traffic management systems, aligning with the objectives of this project.
2. **Real-Time Data Visualisation:** Chart.js, a library for creating interactive charts, is employed to dynamically visualise parking data. Real-time updates are crucial for parking management systems, as demonstrated by existing smart parking solutions in urban centres.
3. **Database-Driven Web Applications:** My experience with databases such as MongoDB, combined with server-side frameworks like Node.js and Express, has enabled the creation of a scalable backend. These architectures are commonly used in applications requiring real-time data synchronisation, such as ride-sharing and logistics platforms.
4. **Accessibility and User-Centric Design:** My exposure to accessibility best practices, both academically and professionally, informs the project's commitment to

inclusivity. Features such as colourblind-friendly markers, language options, and intuitive navigation are designed to meet the needs of diverse users.

Challenges in Prior Work and Their Application

The technical and logistical challenges I encountered in previous roles, such as debugging content integrations for the Electric Ireland website and ensuring design consistency across CMS platforms, mirror some of the hurdles faced in this project. For example, optimising user interfaces for different screen sizes and ensuring seamless data flow between frontend and backend systems have been recurring themes.

Drawing on this experience, along with lessons from my academic studies, the project builds on a solid foundation of practical and theoretical knowledge. This integration of prior work and technical resources ensures that **ParkMe | MU**delivers a comprehensive and impactful solution.

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.
- **Example:** Used ChatGPT to learn about responsive design principles, which guided the creation of a mobile-friendly interface, ensuring that the application is accessible and user-friendly across various devices and screen sizes.

Exploring and Integrating New Technologies

- **Usage:** Utilised GenAI to explore and understand new technologies and APIs relevant to the project's requirements.
- **Example:** Consulted ChatGPT to learn how to integrate the Google Maps API effectively. This guidance was instrumental in implementing the navigation assistance feature, allowing users to receive accurate directions to available parking spots.

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.
- **Example:** Used ChatGPT to obtain recommendations for creating a user-friendly and aesthetically pleasing interface. Applied these suggestions to design the frontend layout, resulting in a more intuitive and engaging user experience.

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*

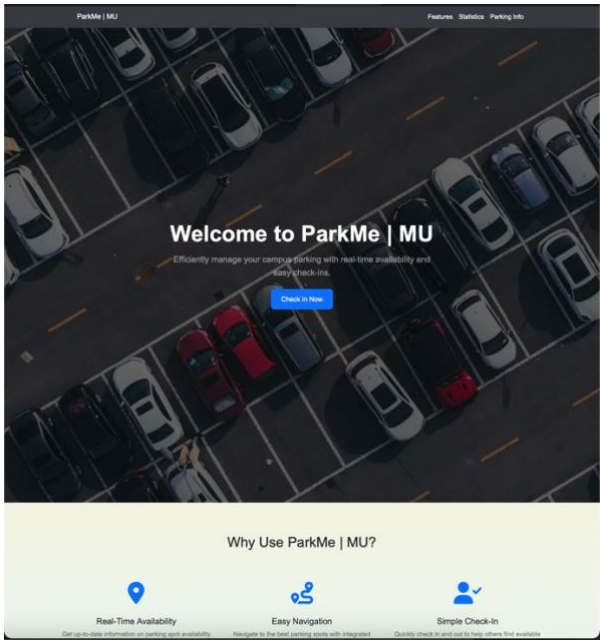


Figure 1: *Features under landing*

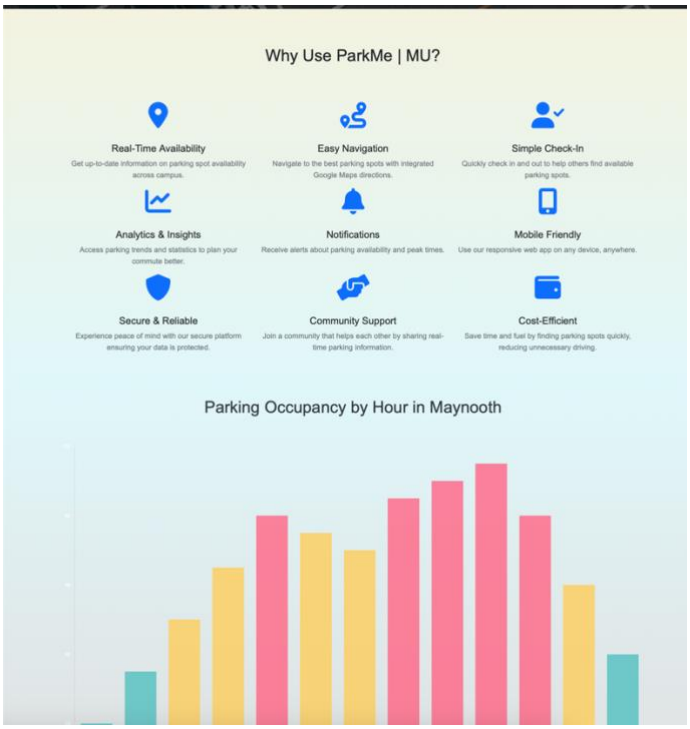


Figure 2: *Occupancy Graph under features*

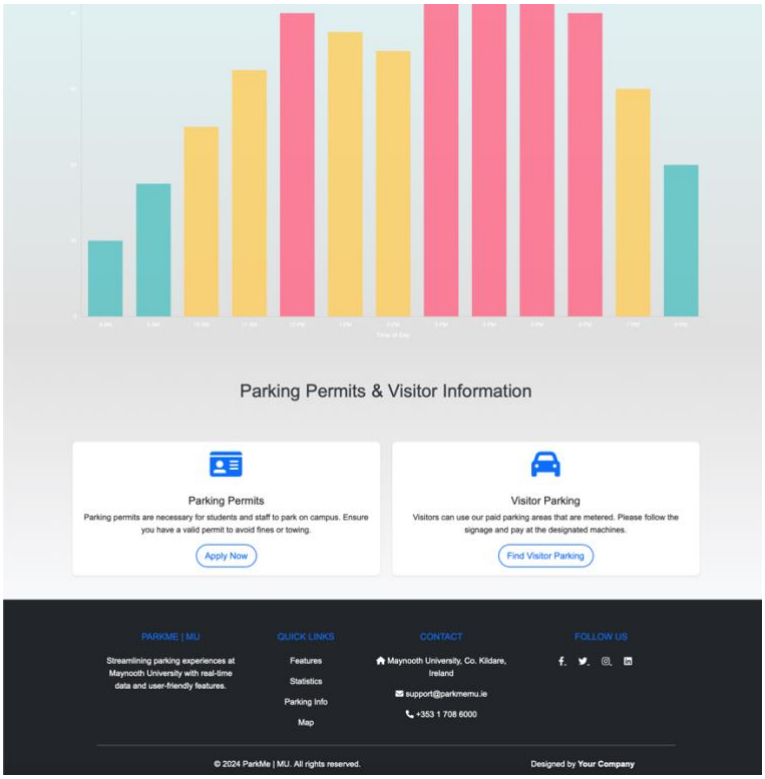


Figure 3: *Interactive App with Realtime data with legend*

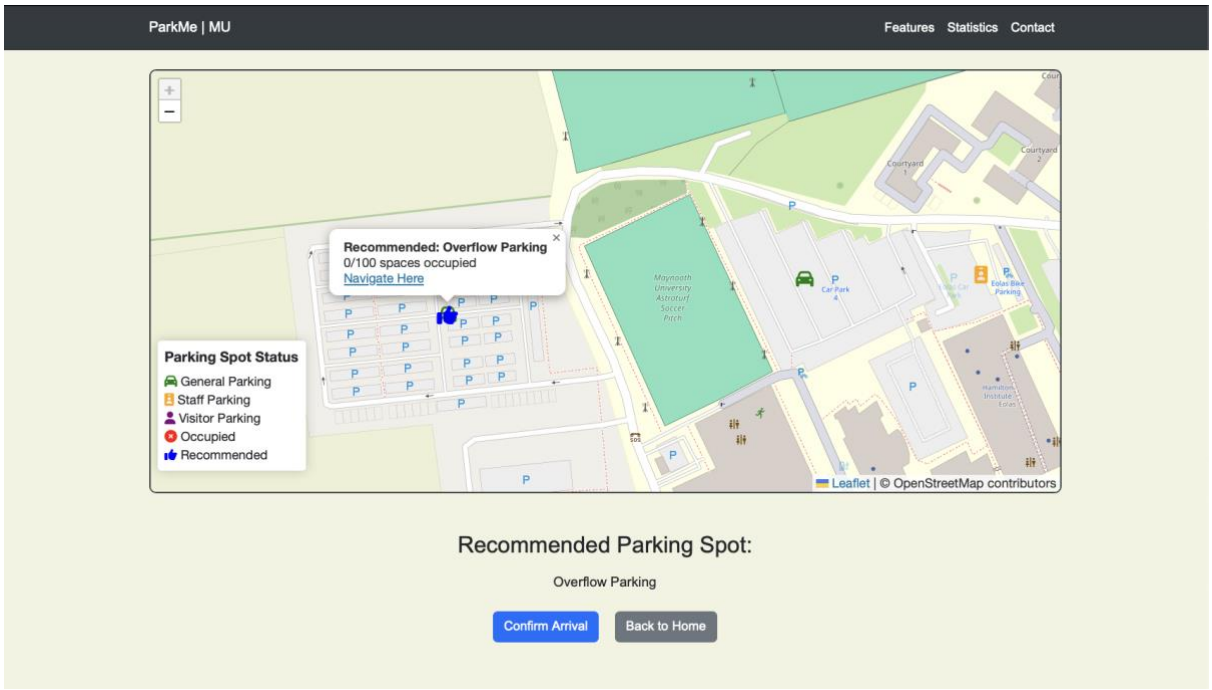


Figure 4: *Admin page*

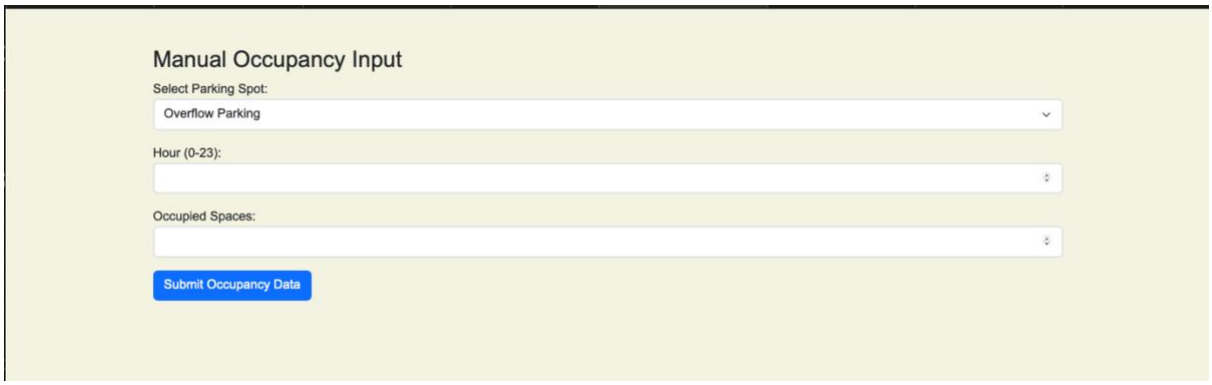
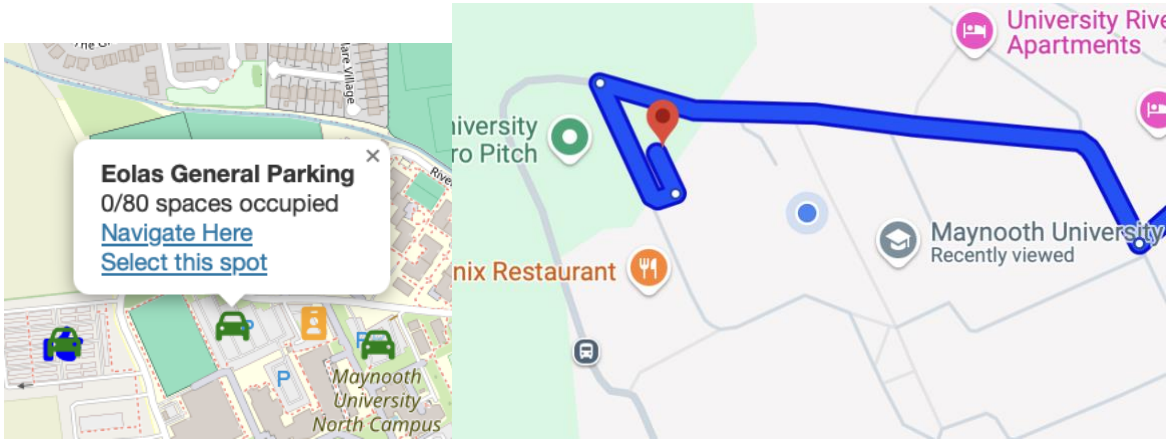
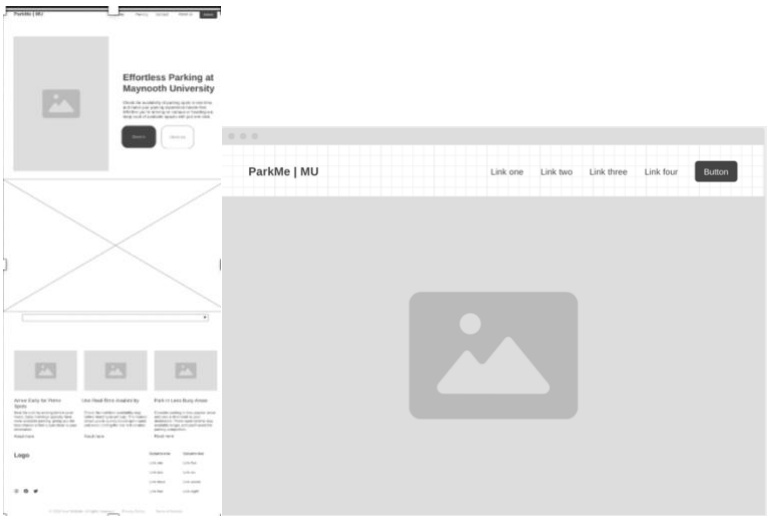


Figure 5: *Google maps integration with different icons*



Appendix B: Wireframe

Figure 6: *Initial Wireframes*



Appendix C: Backend

Figure 7: *MongoDB Dashboard*

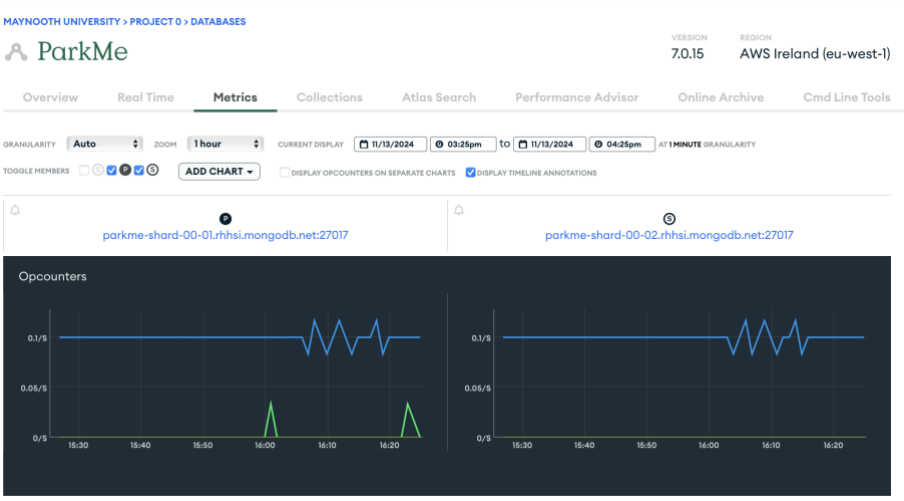


Figure 8: *MongoDB Collection Sample*

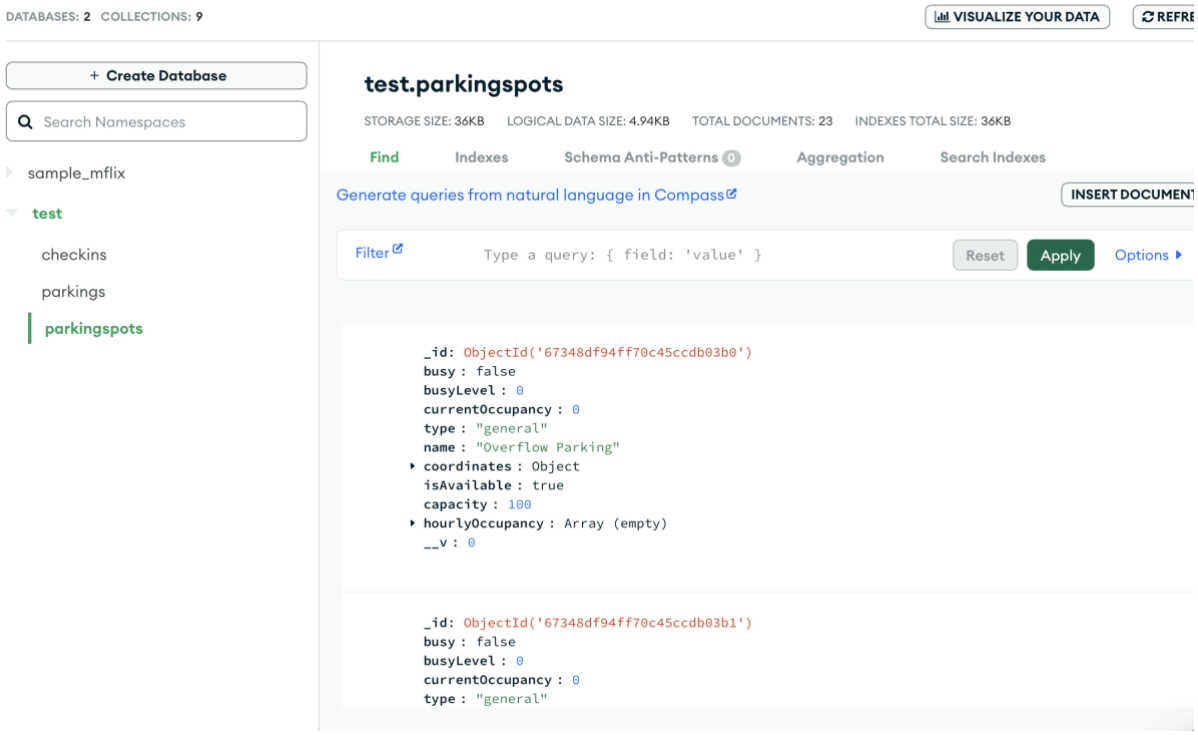
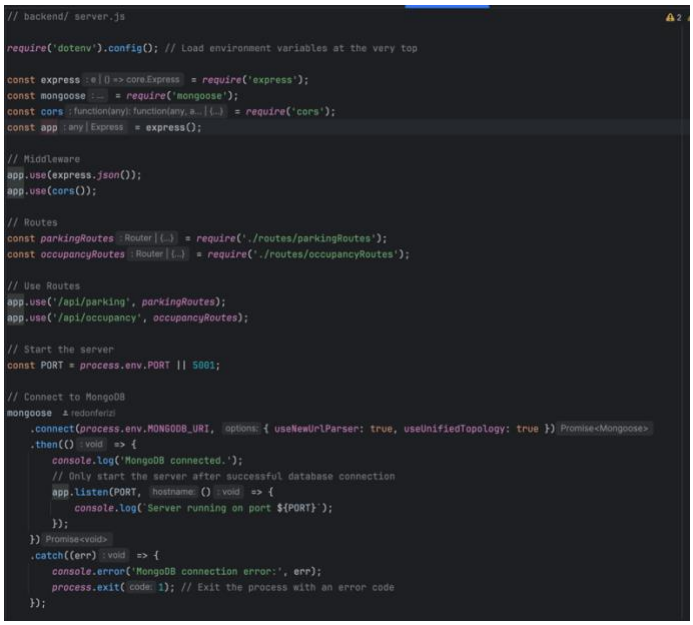
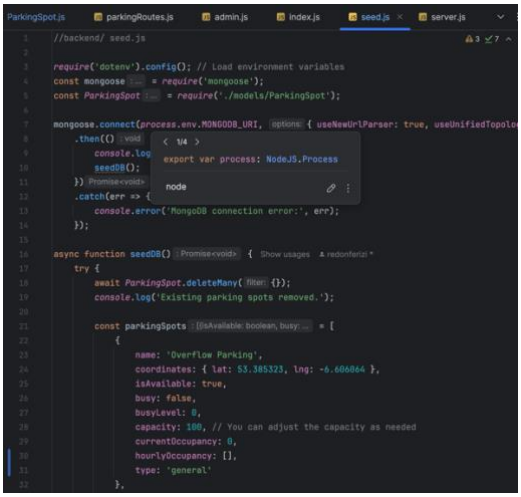


Figure 9: *Server.js Config*



```
// 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*

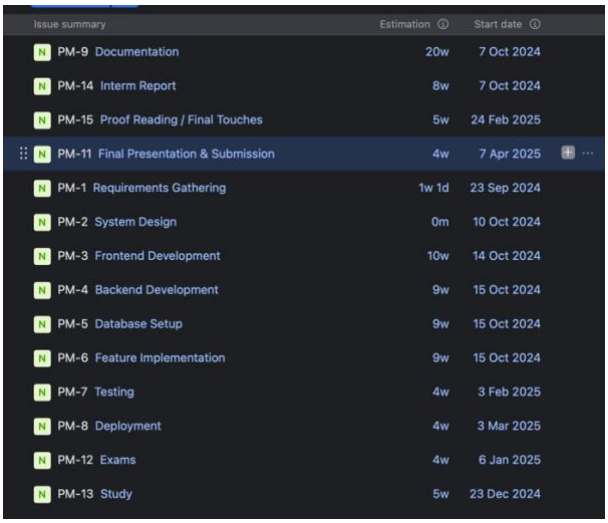


```
ParkingSpot.js  parkingRoutes.js  admin.js  index.js  seed.js  server.js

//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: -6.606864 },
                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

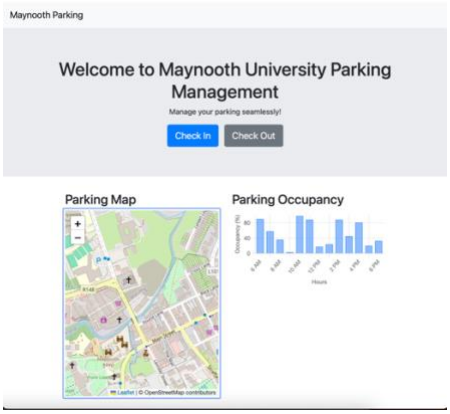
Figure 11: *YouTrack Gantt Chart*

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

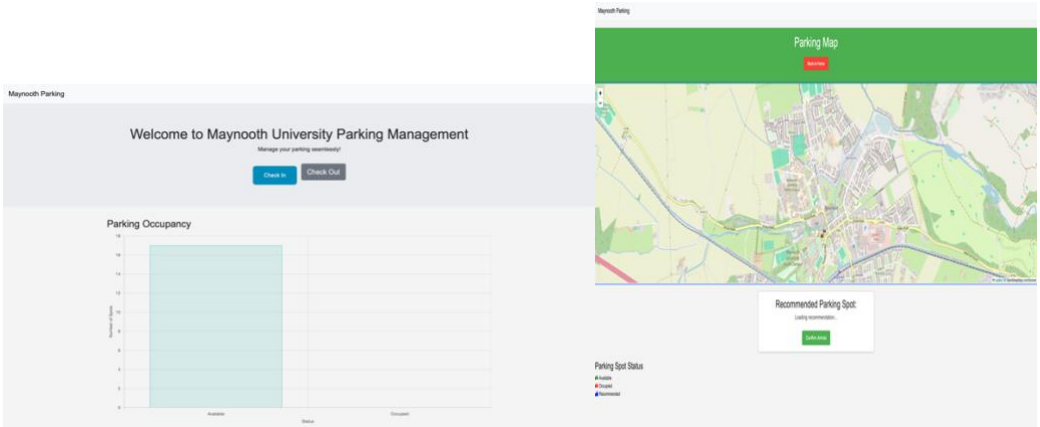


Appendix E: UI Iterations

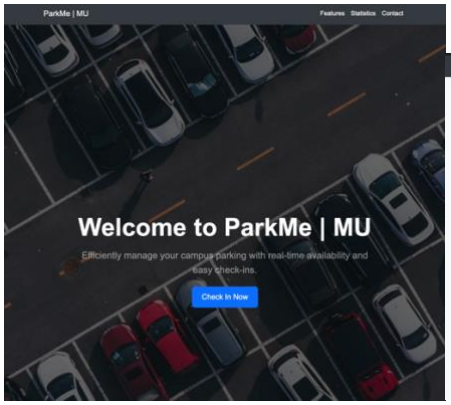
11th October

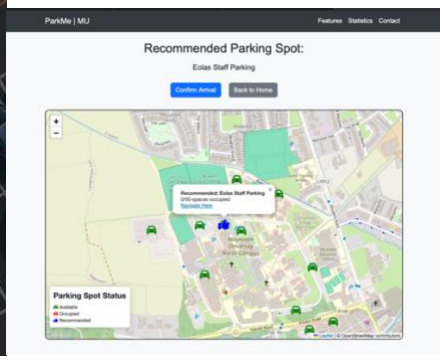


25th October



6th November





13th November