# CS440 Maynooth Parking App

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

# Introduction

Parking at Maynooth University is a common challenge, with limited spots filling up quickly due to the large number of students and staff driving to campus. A potential solution to this issue is the development of a web or mobile app that manages parking availability in real time, improving the efficiency of space allocation and reducing time spent searching for parking.

# Project Objectives

The primary objective of this project is to design and implement a system that:

* Allows students to check in and out of parking spaces.
* Provides real-time information on available spaces.
* Reduces congestion and time spent looking for parking.
* Ensures ease of use with minimal user interaction (e.g., automated space availability updates).

# Brain storm

* Feature that asks user how long they will be there for ? (tbc)
* Ask user how busy is parking approx. how many spaces left ? (implemented)
* Web app rather than mobile app (implemented)
* Possibility of making mobile app that runs webapp (tbc)
* Collect data of each car park to determine when it is busiest and use data to direct users at specific times (tbc)

# Approaches

Check-In/Check-Out System

How it works: Users manually “check-in” when they occupy a parking spot, and “check-out” when leaving. The system then makes the spot available for other users to see.

Advantages:

Simple to implement.

Challenges:

Manual check-ins might discourage usage.

Automatic Location-Based System

How it works: The app uses GPS to track when a user enters or leaves a parking spot. When a spot becomes available, the system automatically updates and informs nearby users in need of parking.

Advantages:

Minimizes user input by automating the check-in/out process.

Provides real-time updates, increasing accuracy.

Challenges:

Privacy concerns related to location tracking.

More complex development involving GPS and possibly geofencing.

Requires constant location tracking, which can drain phone battery.

Hybrid Approach

How it works: Combines check-in/out functionality with automatic detection of available spaces. Users can choose between manually checking out or allowing location data to automatically check them out.

Advantages:

Flexibility for users who prefer manual or automated options.

Reduces inaccuracies from forgotten manual check-outs.

Challenges:

Complexity in development, requiring both manual and automatic systems to work seamlessly.

# Potential Features

* **Push notifications** to alert users when nearby parking becomes available.
* **Map view** showing real-time availability of parking spots across campus.
* **User history** allowing users to see past check-ins and estimate their parking behaviour.
* **Feedback system** for users to report issues or improvements.
* Show busy times for parking areas
* Collect data from check in for status of parking

# Technologies

**Frontend**: React Native or Flutter for mobile app; HTML5, CSS, and JavaScript for web app.

**Backend**: NodeJS with Express or Django to manage user data and parking space information.

**Database**: MongoDB for storing parking data and user interactions.

**GPS & Geofencing**: leaflet

# Layout Approaches

To ensure usability and clarity, I have considered several layout options for the web app:

**Option 1: Simple Dashboard**

* **Layout Overview:** A minimalistic interface with a map displaying parking spaces and a list of nearby lots. A user can check in or out from this page with a simple button.
* **Advantages:** Clean and simple, ensuring ease of navigation and minimal confusion for users.
* **Challenges:** Limited space to present other data, such as congestion trends or user feedback.

**Option 2: Integrated Data View**

* **Layout Overview:** This design includes a map and real-time parking data presented side by side. Under the map, a bar chart would display the busiest times, with times on the x-axis and the number of cars on the y-axis. Beneath this, there will be check-in and check-out options.
* **Advantages:**
  + Allows users to see both real-time availability and historical congestion patterns.
  + Supports a feature where users can plan their parking based on the busiest times.
* **Challenges:** Balancing the display of real-time and historical data without overwhelming the user interface.

# Main Tasks

## **1.**Requirements Gathering

* **Description:**
  + Collect and document all functional and non-functional requirements for the Parking App by consulting stakeholders, conducting surveys, and analysing existing parking systems.
* **Deliverable:**
  + Comprehensive Requirements Specification Document.
* **Start Date:**
  + 2024-09-30
* **End Date:**
  + 2024-10-10
* **Dependencies:**
  + Initial project approval.
* **Key Risks & Contingency:**
  + **Risk:** Difficulty in obtaining comprehensive requirements from stakeholders.
  + **Contingency:** Schedule additional meetings and use alternative methods like online surveys or interviews to gather necessary information.

## 2. System Design

* **Description:**
  + Design the overall architecture of the app, including frontend and backend components, database schema, and integration points using wireframes and UML diagrams.
* **Deliverable:**
  + System Architecture Diagrams and Wireframes.
* **Start Date:**
  + 2024-10-10
* **End Date:**
  + 2024-10-15
* **Dependencies:**
  + Completion of Requirements Gathering.
* **Key Risks & Contingency:**
  + **Risk:** Design complexity leading to delays.
  + **Contingency:** Break down the design into smaller, manageable modules and seek feedback iteratively.

## **3. Frontend Development**

* **Description:**
  + Develop the user interface of the web app using HTML5, CSS, JavaScript, and React (or your chosen framework), ensuring responsiveness and usability.
* **Deliverable:**
  + Functional Frontend Interface.
* **Start Date:**
  + 2024-10-15
* **End Date:**
  + 2024-12-15
* **Dependencies:**
  + Completion of System Design.
* **Key Risks & Contingency:**
  + **Risk:** Frontend framework learning curve.
  + **Contingency:** Allocate additional time for learning and utilise online resources or seek mentorship if needed.

## 4. Backend Development

* **Description:**
  + Implement the server-side logic using NodeJS with Express (or Django), including APIs for user management, parking space data, and real-time updates.
* **Deliverable:**
  + Functional Backend Server with APIs.
* **Start Date:**
  + 2024-10-15
* **End Date:**
  + 2024-12-15
* **Dependencies:**
  + Completion of System Design.
* **Key Risks & Contingency:**
  + **Risk:** Integration issues between frontend and backend.
  + **Contingency:** Regular integration testing and clear API documentation to ensure seamless communication.

## 5. Database Setup

* **Description:**
  + Design and set up the database using Firebase or MongoDB to store user data, parking space information, and interaction logs.
* **Deliverable:**
  + Configured and Populated Database.
* **Start Date:**
  + 2024-10-15
* **End Date:**
  + 2024-10-20
* **Dependencies:**
  + Completion of Backend Development.
* **Key Risks & Contingency:**
  + **Risk:** Data schema design flaws.
  + **Contingency:** Perform thorough schema reviews and iterate based on feedback.

## 7. Feature Implementation

* **Description:**
  + Develop core features such as Check-In/Check-Out, Real-Time Availability, Push Notifications, Map View, User History, and Feedback System.
* **Deliverable:**
  + Fully Functional Features Integrated into the App.
* **Start Date:**
  + 2024-10-15
* **End Date:**
  + 2025-02-01
* **Dependencies:**
  + Completion of Backend and Frontend Development.
* **Key Risks & Contingency:**
  + **Risk:** Feature complexity leading to delays.
  + **Contingency:** Prioritise essential features first and implement additional features in iterations.

## 8. Testing

* **Description:**
  + Conduct unit testing, integration testing, and user acceptance testing to ensure all components function correctly and meet requirements.
* **Deliverable:**
  + Test Reports and Bug Fixes.
* **Start Date:**
  + 2025-02-01
* **End Date:**
  + 2025-03-01
* **Dependencies:**
  + Completion of Feature Implementation.
* **Key Risks & Contingency:**
  + **Risk:** High number of bugs discovered late in the process.
  + **Contingency:** Allocate sufficient time for testing and prioritise bug fixes based on severity.

## 9. Deployment

* **Description:**
  + Deploy the web app to a live server or hosting platform, ensuring scalability and reliability.
* **Deliverable:**
  + Live Deployed Application Accessible to Users.
* **Start Date:**
  + 2025-03-01
* **End Date:**
  + 2024-03-28
* **Dependencies:**
  + Successful Testing Phase.
* **Key Risks & Contingency:**
  + **Risk:** Deployment issues such as server downtime or misconfigurations.
  + **Contingency:** Perform a staged deployment with rollback plans and conduct thorough pre-deployment checks.

## 10. Documentation

* **Description:**
  + Prepare comprehensive documentation, including user manuals, technical documentation, and project reports.
* **Deliverable:**
  + Complete Documentation Package.
* **Start Date:**
  + 2024-09-30
* **End Date:**
  + 2025-03-28
* **Dependencies:**
  + Throughout the project, especially post-development.
* **Key Risks & Contingency:**
  + **Risk:** Incomplete or unclear documentation.
  + **Contingency:** Start documentation early and review regularly to ensure completeness and clarity.

## 11. Final Presentation & Submission

* **Description:**
  + Prepare and deliver the final project presentation, and submit all required materials before the deadline.
* **Deliverable:**
  + Presentation Slides, Demonstration of the App, and Submission of Reports.
* **Start Date:**
  + 2025-03-01
* **End Date:**
  + 2024-04-08
* **Dependencies:**
  + Completion of all previous tasks.
* **Key Risks & Contingency:**
  + **Risk:** Incomplete preparation for the presentation.
  + **Contingency:** Allocate dedicated time for rehearsal and ensure all materials are ready well in advance.

# Gantt Chart

A screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generated

# Example layout

Wireframe 1

A screenshot of a web page

Description automatically generated

A screenshot of a computer

Description automatically generated

**Free**

**0mins**

**Busy**

**20mins**



# Major Iterations

## 11th October

A screenshot of a parking map

Description automatically generated

## 25th October

A screen shot of a parking management

Description automatically generated

A map of a parking spot

Description automatically generated

## 6th November

A screenshot of a parking lot

Description automatically generated

A screenshot of a map

Description automatically generated

# Use Of AI

- troubleshooting, error detection

- npm

- git

- front js

- backend setup

# Sources Used

<https://medium.com/@ibrahimhz/creating-your-first-backend-with-node-js-step-by-step-guide-892769af4cb0>

<https://leafletjs.com>

<https://wireframe.cc>

<https://www.w3schools.com/bootstrap5/>

<https://product.hubspot.com/blog/git-and-github-tutorial-for-beginners>