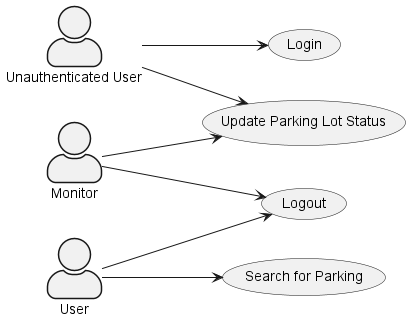
# Analysis and Design

## Introduction

Perfect Parking is a web application that will allow users to find parking in a city such as Limerick. The application will allow users to search for parking near a specific location, and will show the user the nearest parking to their location.

## Use Case Diagram



Alt text

## Actors

* Administrator: The administrator is responsible for managing the application. The administrator can add new parking locations to the database, and can also remove parking locations from the database.
* User: The user is the person who will be using the application. The user can search for parking near a specific location.
* Guest: The guest is a person who is not logged in to the application. The guest can only search for parking near a specific location.

## Use Case Descriptions

### Use Case: Find Parking

#### Description

A user searches for parking near a specific location.

#### Actors

* User

#### Trigger Event

* A user wants to find parking near a specific location.

#### Preconditions

* The user is logged in to the application.
* The website has permission to access the user’s GPS location.

#### Post conditions

* The user is shown a list of parking locations near the location they searched for.

#### Main Flow

1. The user details the location they want to find parking near by:
   * by searching for a specific address in the search bar.
   * by clicking on a location on the map.
   * by clicking on a location on the list of parking locations.
   * using the current location of the user.
2. The application shows the user a list of parking locations near the location they searched for.

#### Alternative Flows

* If the user does not have permission to access their GPS location, the user can search for a specific address in the search bar.

### Use Case: Register User

#### Description

A user registers for an account on the application.

#### Actors

* Guest user

#### Trigger Event

* A guest user wants to register for an account on the application.

#### Preconditions

* The guest user is not logged in to the application.
* The guest user has not registered for an account on the application.
* The guest has a valid email address.

#### Post conditions

* A user account is created for the guest user.

#### Main Flow

1. The guest user clicks on the “Register” button.
2. The guest user enters their details into the registration form.
3. The guest user clicks on the “Register” button.
4. The application creates a user account for the guest user.
5. The guest logs in to the application.

#### Alternative Flows

* If the guest user enters an email address that is already registered to an account, the application will display an error message.

### Use Case: Login User

#### Description

A user logs in to the application.

#### Actors

* User

#### Trigger Event

* A user wants to log in to the application.

#### Preconditions

* The user is not logged in to the application.

#### Post conditions

* The user is logged in to the application.

#### Main Flow

1. The user clicks on the “Login” button.
2. The user enters their details into the login form.
3. The user clicks on the “Login” button.
4. The application logs the user in to the application.

#### Alternative Flows

* If the user enters an incorrect username and password, the application will display an error message.
* If the user enters an username that is not registered to an account, the application will display an error message.
* If the user account is disabled, the application will display an error message.

### Use Case: Update Parking Lot Status

#### Description

A monitor bot automatically updates the status of a parking lot.

#### Actors

* Monitor

#### Trigger Event

* A monitor updates the status of a parking lot.

#### Preconditions

* The website application is running.
* The monitor is connected to the internet.

#### Post conditions

* The status of the parking lot is updated.

#### Main Flow

1. The monitor sends a PUT request to the application REST API.
2. The application updates the status of the parking lot.

#### Alternative Flows

* If the monitor is not connected to the internet, the monitor will not be able to update the status of the parking lot.
* If the monitor API access token is invalid or has expired, the monitor will not be able to update the status of the parking lot.
* If the monitor sends an invalid request to the application REST API, the application will not update the status of the parking lot.
* If the parking lot does not exist in the database, the application will not update the status of the parking lot.