**Problem Statement**

I own a multi-storey parking lot that can hold up to 'n' cars at any given point in time. Each slot is given a number starting at 1 increasing with increasing distance from the entry point in steps of one. I want to create an automated ticketing system that allows my customers to use my parking lot without human intervention.

When a car enters my parking lot, I want to have a ticket issued to the driver. The ticket issuing process includes us documenting the registration number (number plate), the colour of the car, the make & model of the car and allocating an available parking slot to the car. The customer should be allocated a parking slot which is nearest to the entry.

The system follows the following guidelines.

● There will be 2 different type of users

○ Admin Users - Can create admin user, disable admin user (root admin user cannot be disabled), See state of all parking spaces in a grid, see list of subscribers etc.

○ Customers - Can Register, Login, Park / Unpark Cars, See list of cars parked by them.

● Due to the government regulations an admin must be able to find the following

○ Registration numbers of all cars of a particular colour.

○ Slot number in which a car with a given registration number is parked.

○ Slot numbers of all slots where a car of a particular colour is parked.

● A subscriber will issue a ticket to himself. He/She can park multiple cars

●For setup, when there is no parking space info available (i.e. While initialization) admin can set any number as parking space capacity of the parking lot. Post initialization admins can only increase this number any time.

**Implementation**

**User Management**

Types of Users:

* root
* admin
* customer

To create any user, we can use /register API (<http://localhost:3000/auth/register>)

API Details:

* URL: <http://localhost:3000/auth/register>
* Method: POST
* Body:

{

"userName": "Tony Mathew",

"userEmail": "tonymathew17@gmail.com",

"password": "password",

"userType": "root",

"isDisabled": false

}

* Password is hashed before saving using “bcrpytjs” node module

To access any APIs, the user needs to login.

Login API details:

* URL: <http://localhost:3000/auth/login>
* Method: POST
* Body:

{

"userEmail":"ronakp@gmail.com",

"password":"password"

}

1. If the credentials are correct, an access token will be returned back in response header. This access token needs to be persisted in client side session storage to make subsequent API calls.

**API Validation**

To validate each API request, JWT(JSON Web Token) is used. The token will be having a payload which is enough to identify the user making the request. Payload contains userEmail, userType and isDisabled fields, with which a decision is made weather to route the request to API or not.

**Parking Lot Requirement Implementation**

To implement the parking lot requirements listed in the problem statement, following APIs have been defined:

1. createParkingLot
2. This API will connect to mongoDB using mongoose client, and create the following entry in db:

{

"\_id" : ObjectId("5f62723e03dafb3054c72c3c"),

"slots" : [

null,

null,

null,

null,

null,

null,

null,

null,

null,

null

],

"parkingLotID" : "c0ac32ba-1851-4409-b994-d8e9e059648b",

"totalSlots" : 10,

"availableSlots" : 10,

"\_\_v" : 0

}

1. There will be a separate record created in db for each parking lot created
2. parkCar

This API is used by a customer to park the car. Car Data and parkingLot ID should be provided. This API will return a unique carID and the slot.

1. leaveCar

This API is used by a customer to unpark. Parking Lot ID and CarID should be provided

1. getParkingStatus

This API can only be used by root/admin users to get parking status of a particular parking lot.

1. getCarNosWithSameColor

This API can only be used by root/admin users to get the registration numbers of cars with same color.

1. getSlotsWithSameColorCar

This APi can only be used by admin/root users to get the slots of all cars with same color.

1. getSlotByCarNumber

This API can be accessed only by root/admin users to get the slot of car by its registration number.

1. updateParkingLot

This API can be accessed only by root/admin users to increase the capacity of a parking lot

1. disableAdminUser

This API can be accessed only by root/admin users to disable an admin user. Root user cannot be disabled.

1. getListOfCarsParked

This API can be used only by a customer to get the details of all cars he has parked.

**Installation**

1. Prerequisites:

* Node.Js
* MongoDB

1. npm install
2. npm start