

Nithish Kumar Saravanan

Tejaswi Linge Gowda

Programming for Internet of Things

05 December 2023

### Smart Park – Parking Management with Intelligent License Plate Recognition

The main objective of the project is to deliver a web-based application for real time tracking of user specific parking status for faster and efficient parking in urban areas. The number plate of the vehicle will be detected using cascade classifier and EasyOCR is used for recognizing the plate number. The python code used for detecting and recognizing the number plates is also embedded with certain logic for parking space allotment and for updating the MongoDB Atlas database. A Raspberry Pi microprocessor is used for deploying the python code which is connected to a web camera. Node.js is used on the server side and will be hosted in AWS for providing access to multiple users. When the vehicle exits the parking lot the total payable amount is calculated and displayed on the web interface.

### NUMBER PLATE DETECTION, RECOGNITION AND PARKING SPACE ALLOTMENT

This section describes the process of implementing the number plate detection and recognition algorithm. It also includes a brief explanation of the logic in handling vehicle entries and exits.

*Number Plate Detection.* There are a lot of methods available for number plate detection. One such method is using computer vision techniques and the second method is to use classifier models or machine learning models.

1. Computer vision techniques have better accuracy only when the background is simple. If fails to detection when the background has more features.
2. I have used the cascade classifier for number plate detection. In specific, I have used Haar cascade classifier model already trained for detecting number plate in a camera image frame.

*Number Plate Recognition.* Once the ROI is cropped out using the detection module, it can be passed into an Optical Character Recognition algorithm for recognizing plate number. OCR algorithms can be used only when the number plate is simple enough. For complicated number plates machine learning has to be used for extracting the number plate details from the rest of the characters. I have used EasyOCR for number plate recognition as the license plates I chose are simple.

*Parking Space Allotment.*

1. Once a vehicle number is recognized, the user collection in the database is checked for user details. If the user has already signed up and the credentials are correct the user will be logged in or else the user will be asked to sign up and come back later.
2. Once the user signs up, a parking space will be allotted for the user randomly based on the available parking lots.
3. When the vehicle exits, the total payable amount is calculated based on the entry and the exit time.
4. Once the user signs up and logs in the home page will be rendered displaying the parking information of the specific user and all the allotted parking lots.

NODE SERVER AND WEB INTERFACE

This section describes the functions of node server and the features of the user interface.

*Node Server.* Node.js is used in the server-side scripting. The server node contains several handlers for get and post request used for extracting and displaying information on html page and for uploading data into the database respectively.

*User Interface.* The user interface is split up into three .ejs files – login.ejs, signup.ejs, home.ejs. I am using ejs as the rendering engine. The config.js file contains the schema of the collections and connects to the MongoDB Atlas database.

*login.ejs.* Login page

*signup.ejs.* Signup page

*home.ejs.* GUI for Parking space information and payment information during exit.

## SYSTEM COMPONENTS

*License Plate Recognition (LPR) System.* Identifying and recognizing the alphanumeric characters on the plates.

*Camera System.* To capture images of vehicles and their license plates as they enter the parking area.

*Raspberry Pi.* To deploy the LPR system and parking space management system.

*Database.* Stores information about registered users, their license plate data, parking space assignments, and payment details.

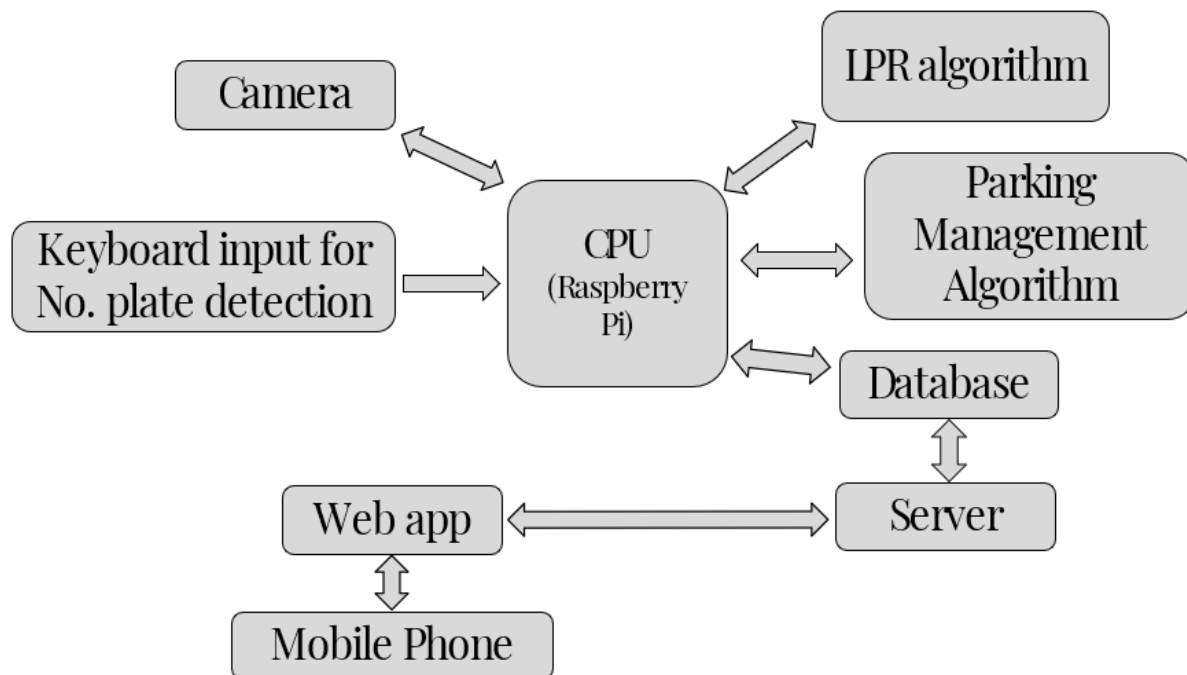
*User Interface.* Web application allows users to register their vehicles, view parking availability, make reservations.

*Server.* Hosts the web application

*Parking Space Management.* Optimizes parking space allocation.

*Central Processing Unit.* Processes the data received from cameras, manages the recognition of license plates, allocates parking spaces.

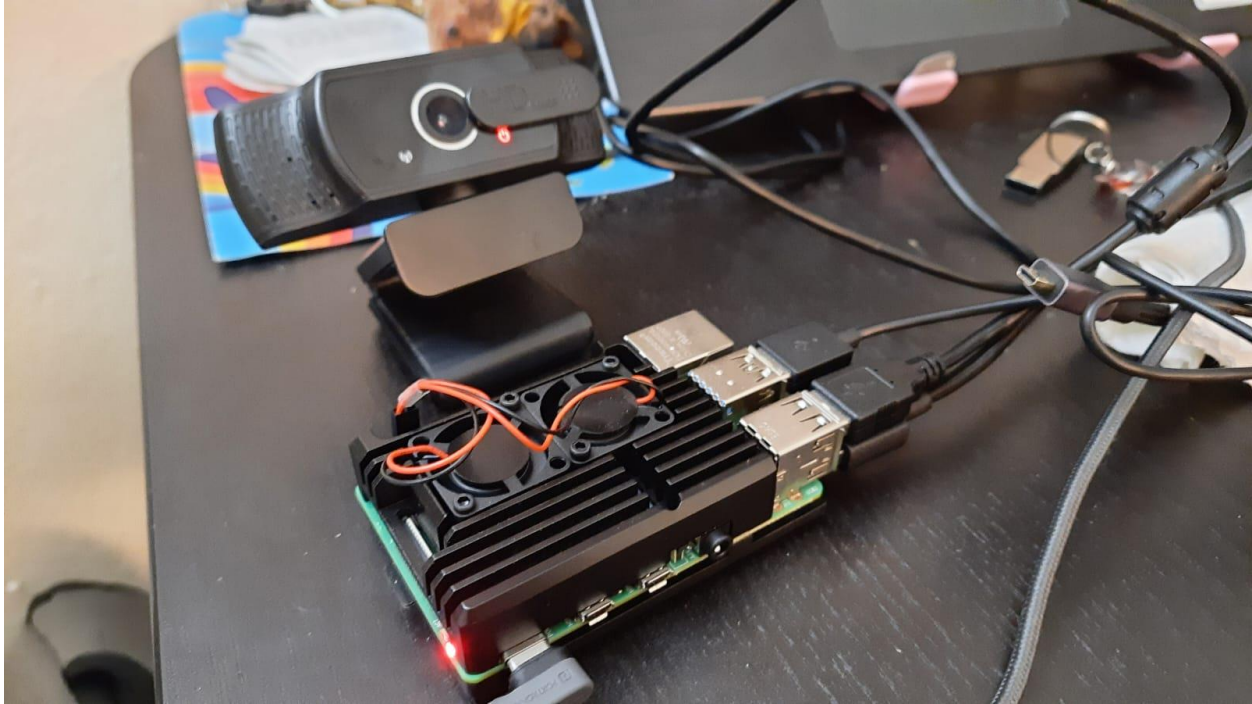
## INTERACTION BETWEEN COMPONENTS



*Fig. 1* Flow diagram of interaction between components

## DEPLOYMENT

This section provides a detailed walk through of the working of the entire system.



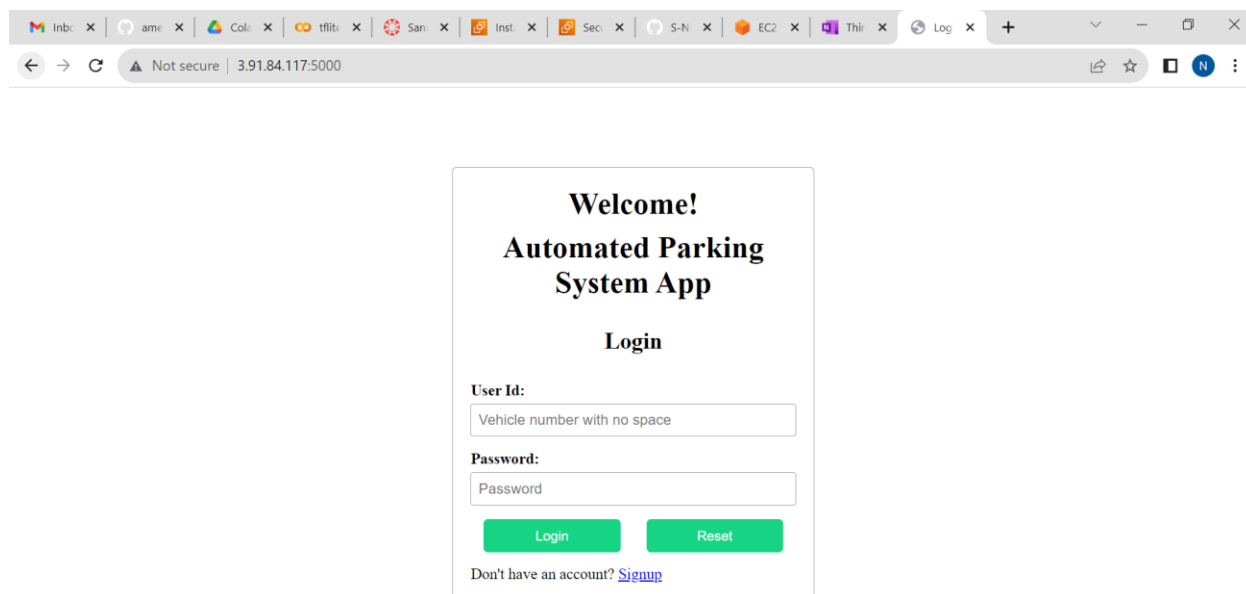
*Fig 2. Raspberry Pi setup*

```

aws
Services
Search
[Alt+S]
N. Virginia
voclabs/user2749945=nsarava3@asu.edu @ 2941-3088-1337
npm help
ubuntu@ip-172-31-21-106:~/IoT_based_automated_vehicle_parking_with_number_plate_recognition/src$ npm uninstall bcrypt
removed 52 packages, and audited 130 packages in 1s
17 packages are looking for funding
  run `npm fund` for details
found 0 vulnerabilities
ubuntu@ip-172-31-21-106:~/IoT_based_automated_vehicle_parking_with_number_plate_recognition/src$ npm cache clean --force
npm WARN using --force Recommended protections disabled.
ubuntu@ip-172-31-21-106:~/IoT_based_automated_vehicle_parking_with_number_plate_recognition/src$ npm install bcrypt
added 52 packages, and audited 182 packages in 4s
20 packages are looking for funding
  run `npm fund` for details
found 0 vulnerabilities
ubuntu@ip-172-31-21-106:~/IoT_based_automated_vehicle_parking_with_number_plate_recognition/src$ node server.js
(node:19822) [DEP0040] DeprecationWarning: The 'punycode' module is deprecated. Please use a userland alternative instead.
(Use `node --trace-deprecation ...` to show where the warning was created)
Server running on Port: 5000
Database connected successfully
i-Oe65eddbb0e21188d (nithish_web_server)
PublicIPs: 3.91.84.117 PrivateIPs: 172.31.21.106
CloudShell Feedback
© 2023, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

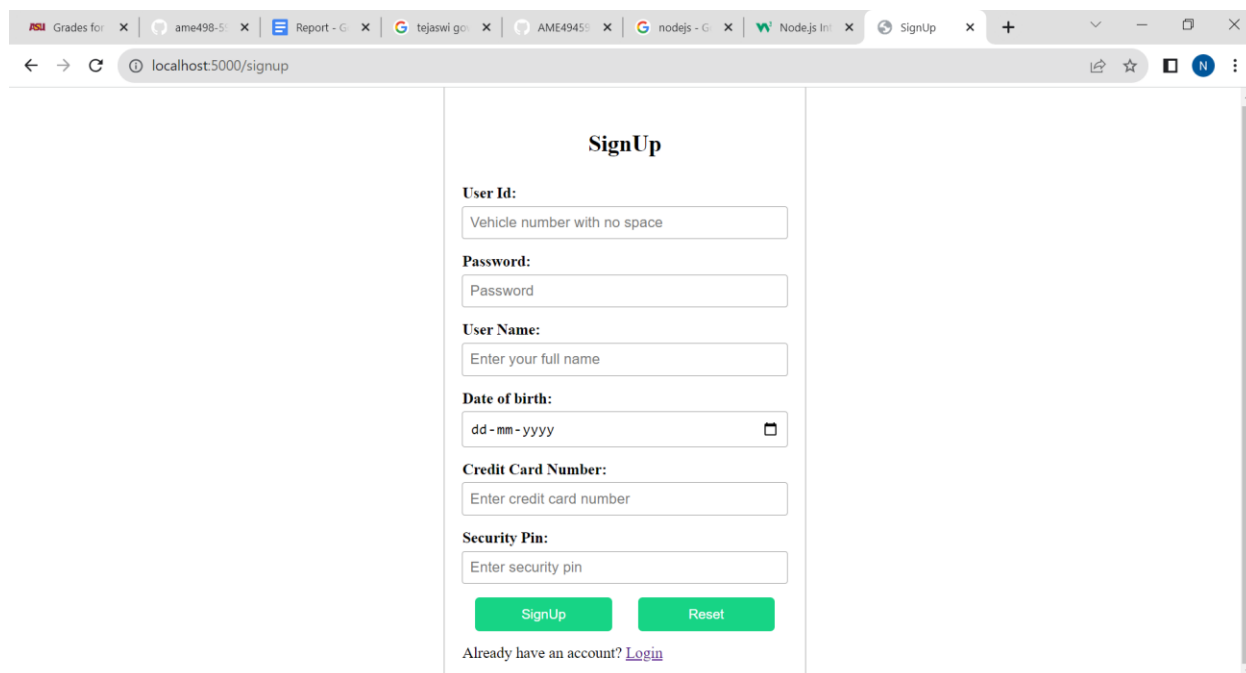
```

*Fig 3. Hosting node server in AWS*



The screenshot shows a web browser window with multiple tabs. The active tab is titled 'Log' and shows a login page for the 'Automated Parking System App'. The page has a white background with a black border. At the top, it says 'Welcome!' in bold. Below that is the title 'Automated Parking System App' in bold. Underneath is the heading 'Login' in bold. There are two input fields: 'User Id:' with a placeholder 'Vehicle number with no space' and 'Password:' with a placeholder 'Password'. Below these fields are two green buttons: 'Login' and 'Reset'. At the bottom, it says 'Don't have an account? [Signup](#)'.

*Fig 4. Login page*



The screenshot shows a web browser window with multiple tabs. The active tab is titled 'Signup' and shows a signup page for the 'Automated Parking System App'. The page has a white background with a black border. At the top, it says 'SignUp' in bold. Below that are several input fields: 'User Id:' with a placeholder 'Vehicle number with no space', 'Password:' with a placeholder 'Password', 'User Name:' with a placeholder 'Enter your full name', 'Date of birth:' with a placeholder 'dd - mm - yyyy' and a calendar icon, 'Credit Card Number:' with a placeholder 'Enter credit card number', and 'Security Pin:' with a placeholder 'Enter security pin'. Below these fields are two green buttons: 'SignUp' and 'Reset'. At the bottom, it says 'Already have an account? [Login](#)'.

*Fig 5. Signup page*

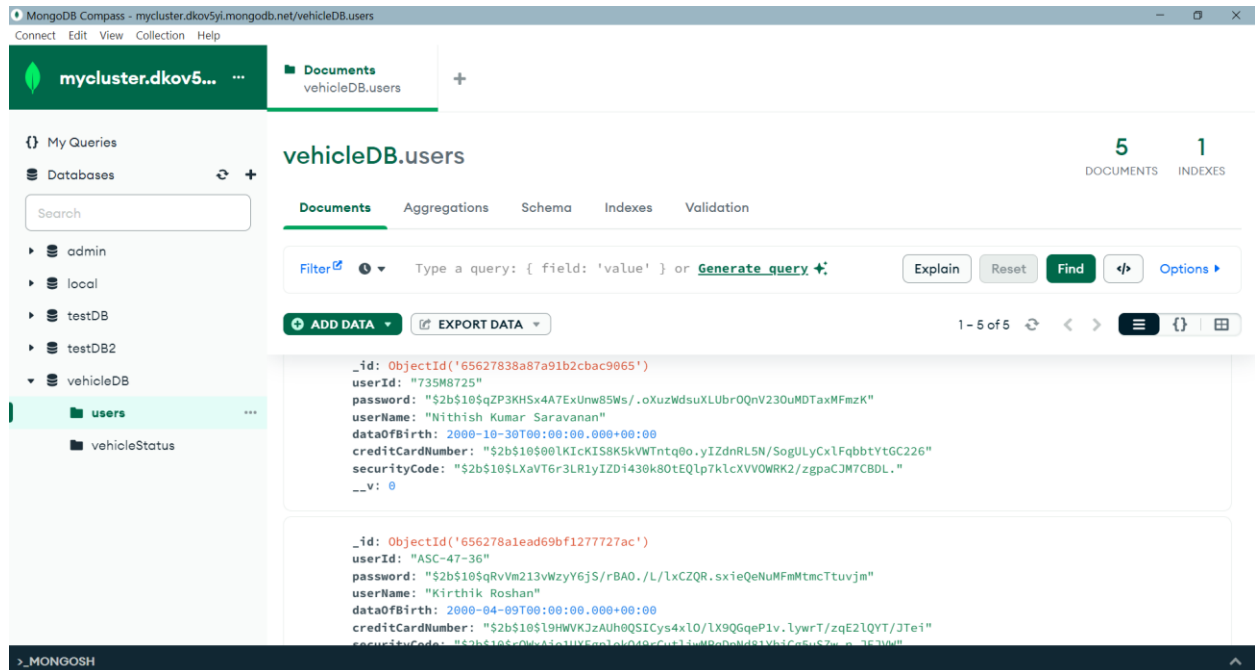


Fig 6. Users Collection – Signed up by users already

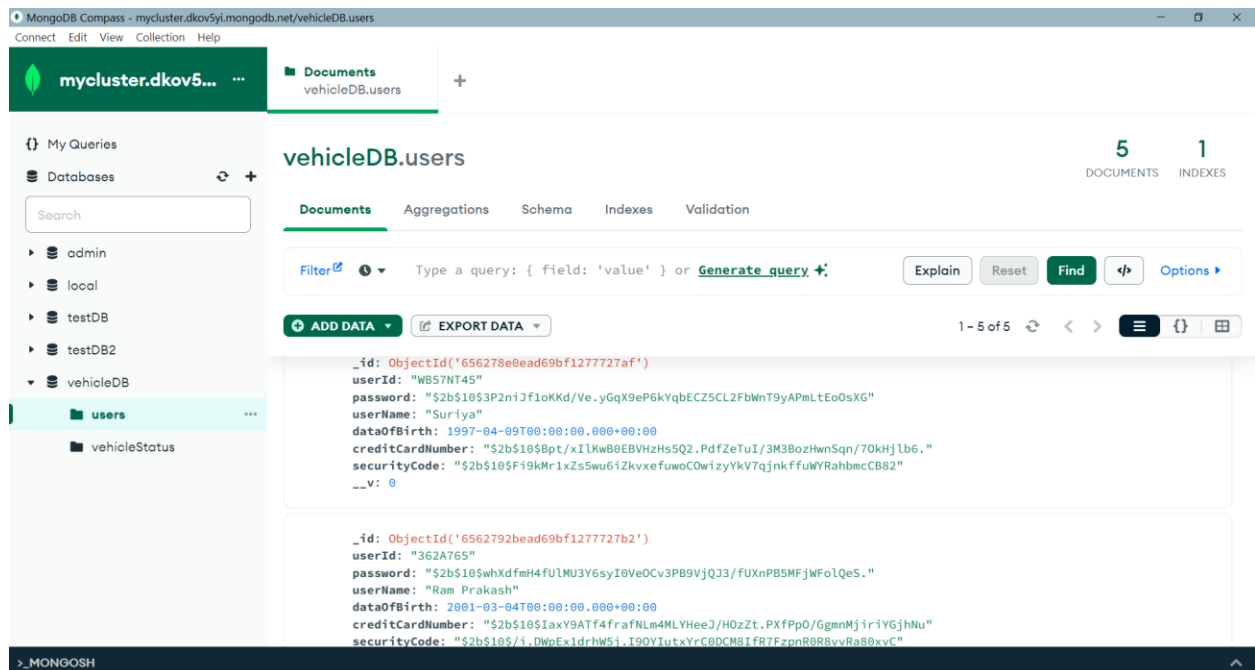
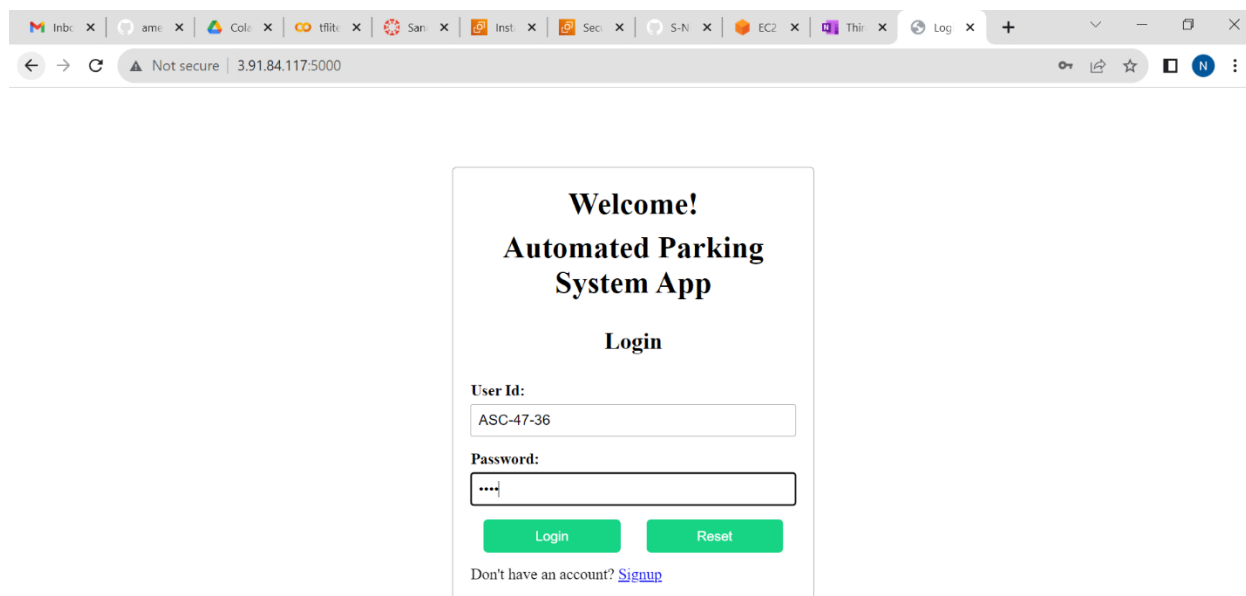


Fig 7. Users Collection – Signed up by users already



Browser tabs: Inbc, ame, Col, tflite, San, Inst, Sec, S-N, EC2, Thir, Log.

Address bar: Not secure | 3.91.84.117:5000

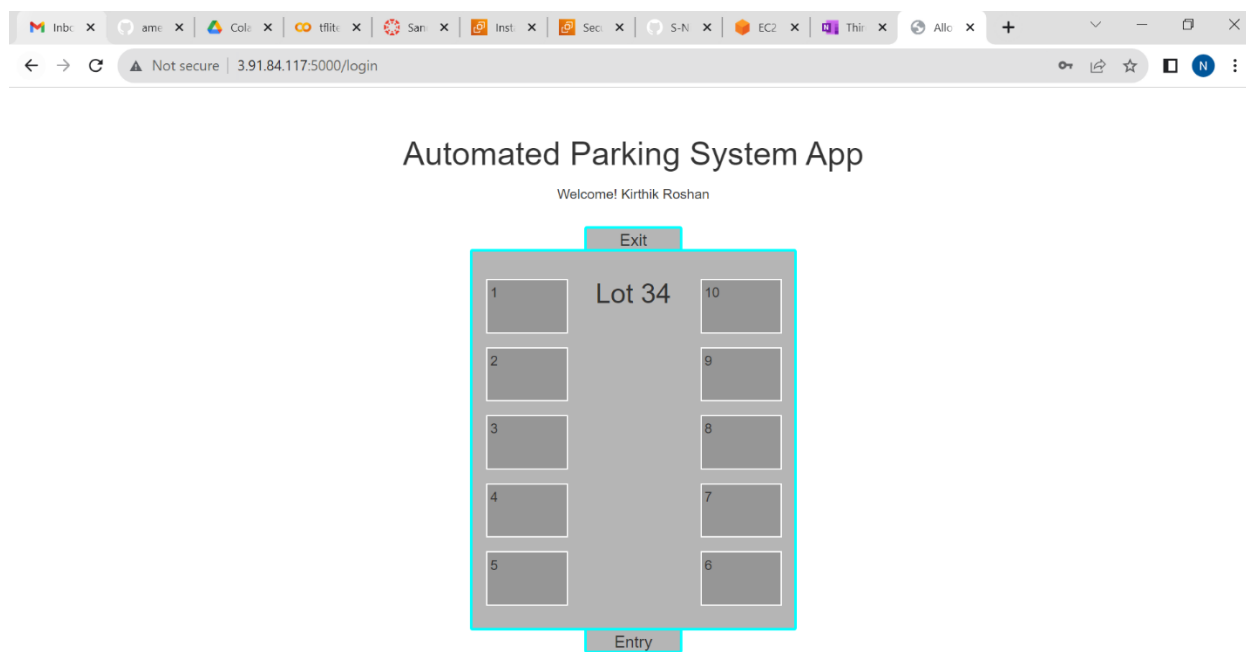
**Welcome!**  
**Automated Parking System App**  
**Login**

User Id:

Password:

Don't have an account? [Signup](#)

Fig 8. Logging in – already signed up user



Browser tabs: Inbc, ame, Col, tflite, San, Inst, Sec, S-N, EC2, Thir, Allc.

Address bar: Not secure | 3.91.84.117:5000/login

**Automated Parking System App**  
Welcome! Kirthik Roshan

Exit

Lot 34

1 2 3 4 5 6 7 8 9 10

Entry

Fig 9. Logging in – already signed up user



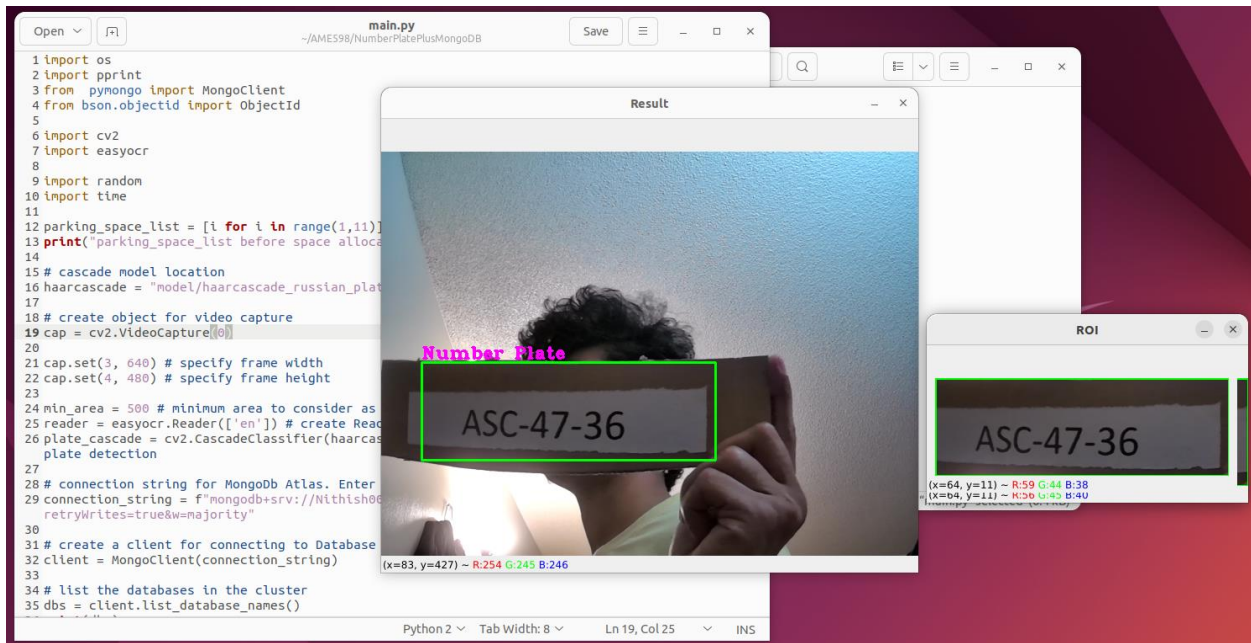


Fig 10. Number plate detection and recognition

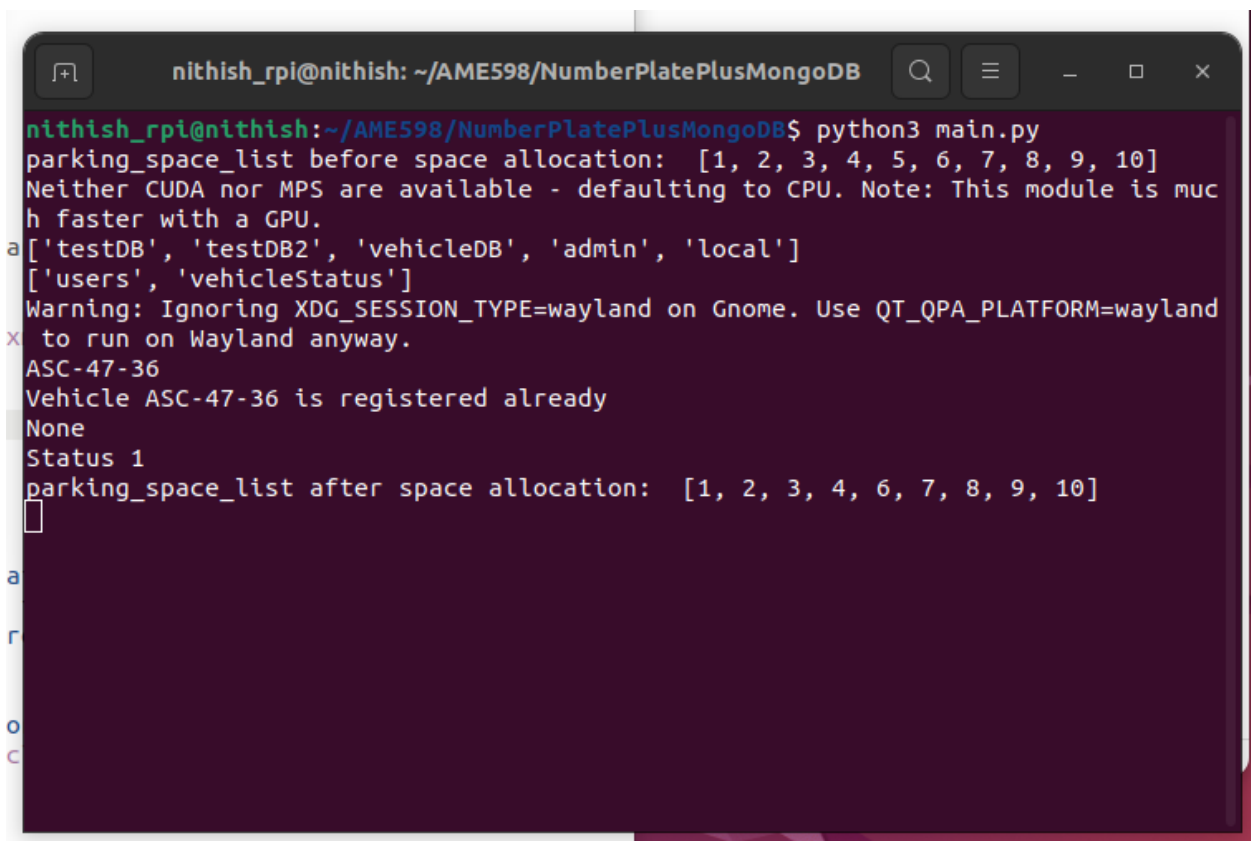


Fig 11. Space allotted after number plate recognition and verifying database

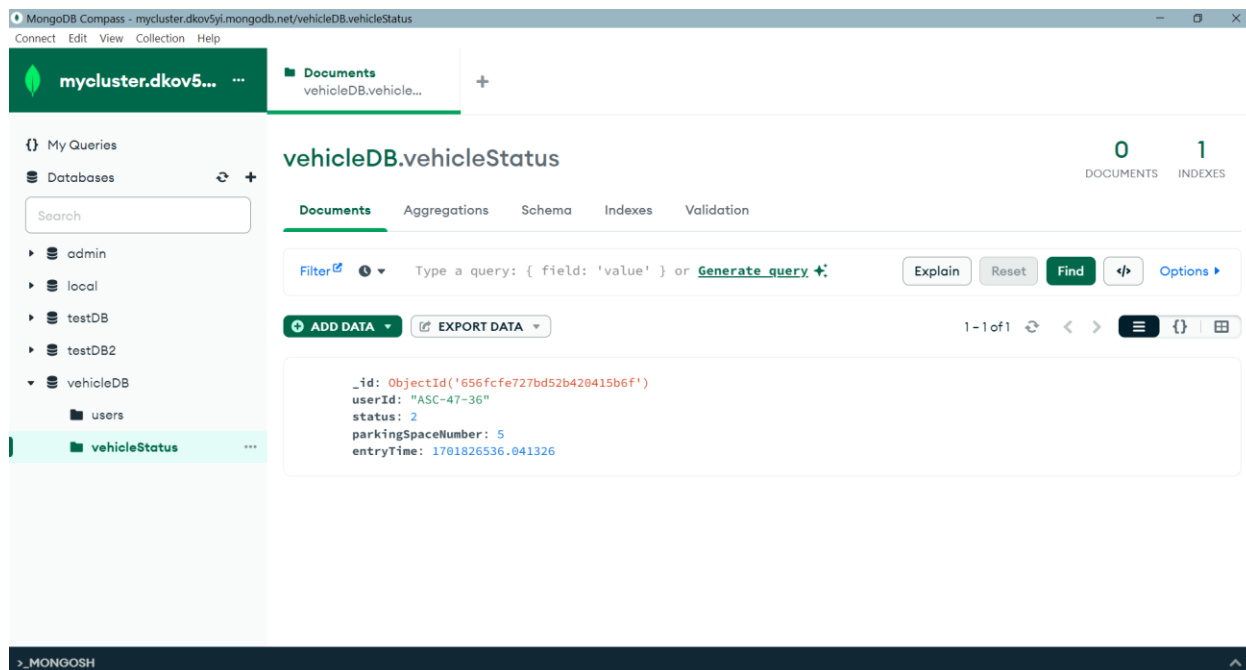


Fig 12. vehicleStatus collection after allotting parking space to the vehicle

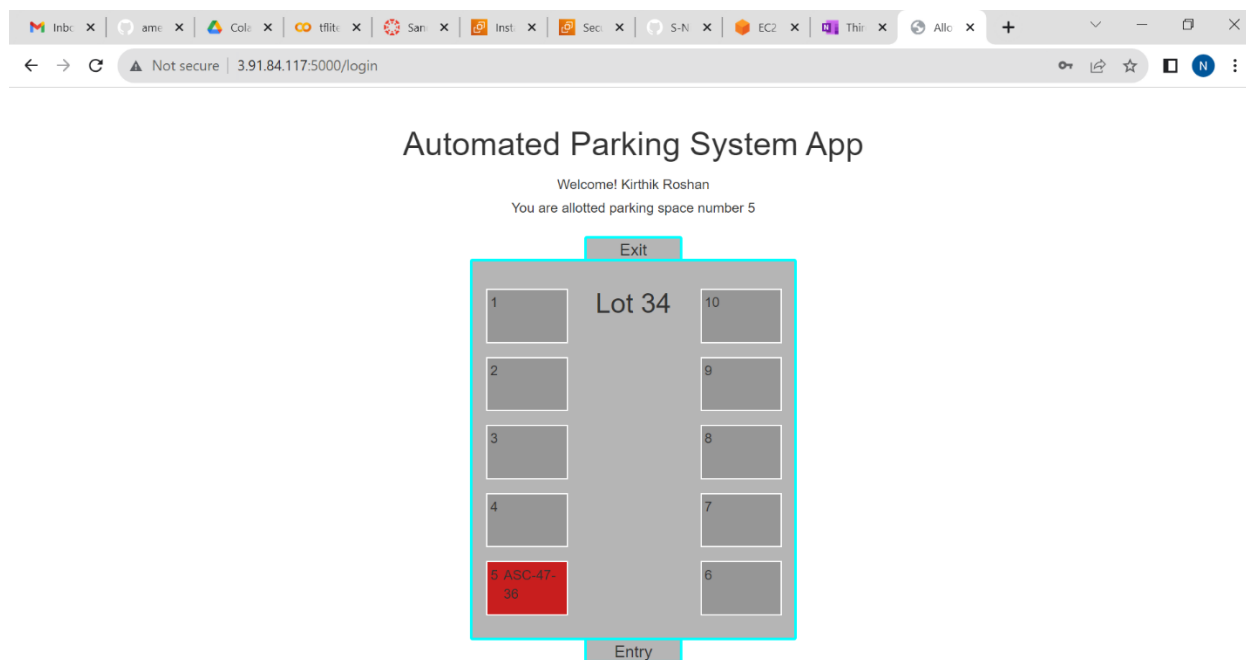


Fig 13. Updated UI after parking space allotment

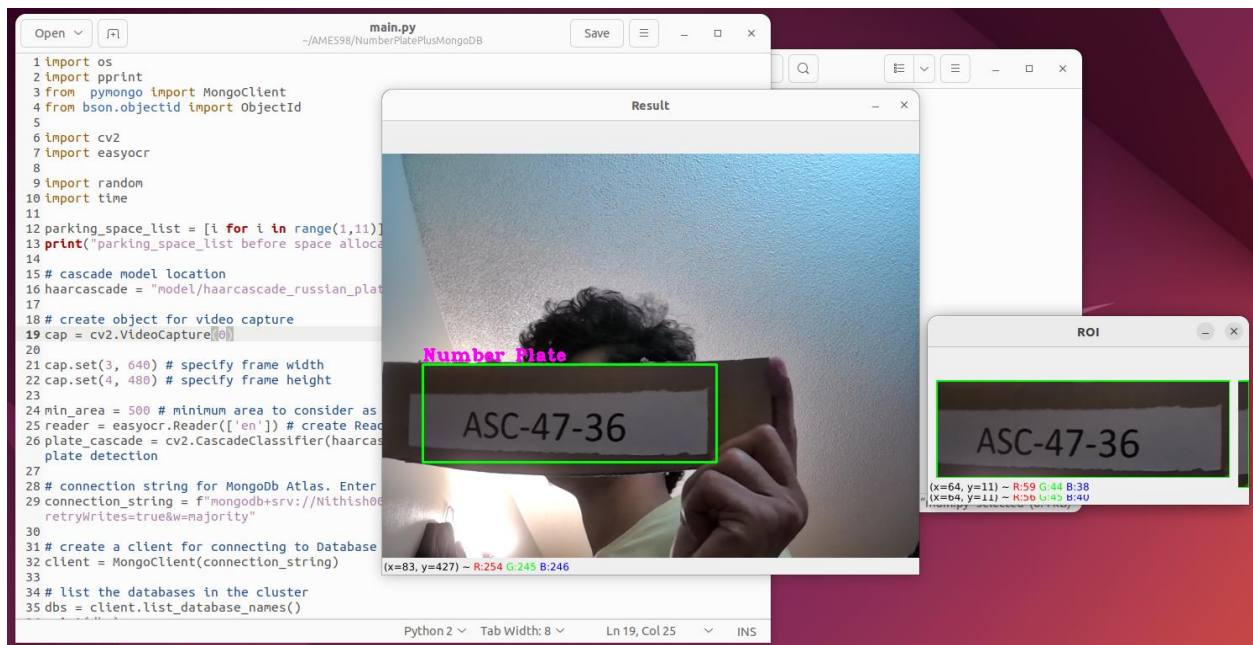


Fig 14. Number plate detection and recognition at the exit

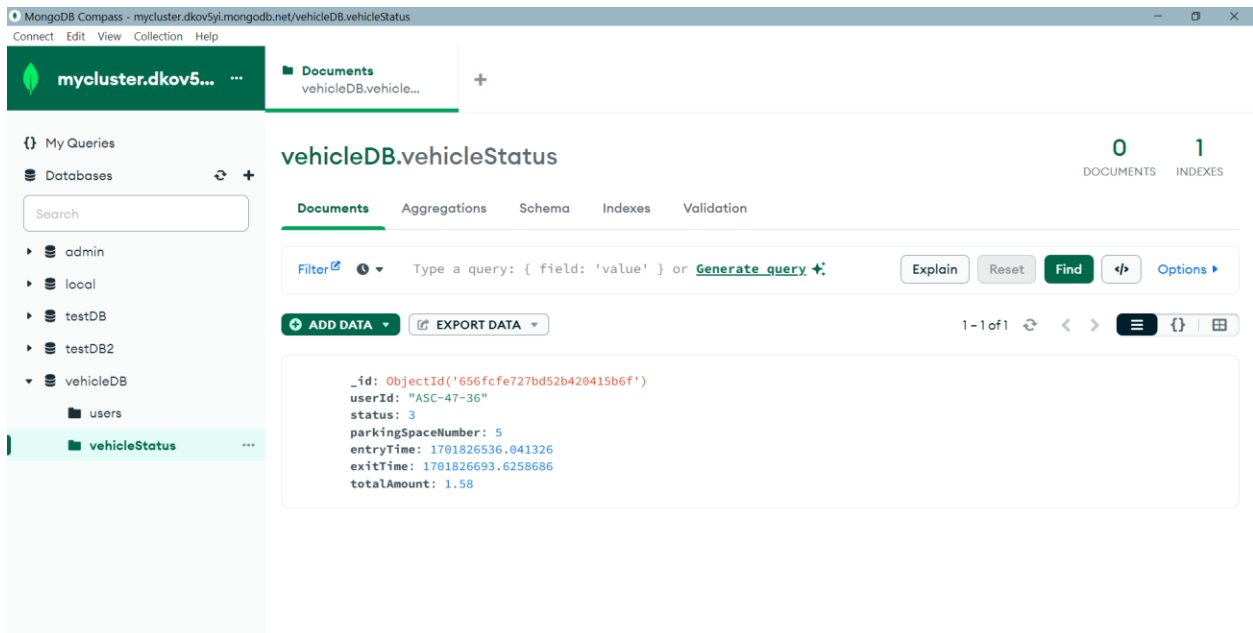


Fig 15. vehicleStatus collection after the vehicle exits

```

nithish_rpi@nithish: ~/AME598/NumberPlatePlusMongoDB
nithish_rpi@nithish:~/AME598/NumberPlatePlusMongoDB$ python3 main.py
parking_space_list before space allocation: [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
Neither CUDA nor MPS are available - defaulting to CPU. Note: This module is much faster with a GPU.
a ['testDB', 'testDB2', 'vehicleDB', 'admin', 'local']
['users', 'vehicleStatus']
Warning: Ignoring XDG_SESSION_TYPE=wayland on Gnome. Use QT_QPA_PLATFORM=wayland to run on Wayland anyway.
x ASC-47-36
Vehicle ASC-47-36 is registered already
None
Status 1
parking_space_list after space allocation: [1, 2, 3, 4, 6, 7, 8, 9, 10]
ASC-47-36
Vehicle ASC-47-36 is registered already
a {'_id': ObjectId('656fcfe727bd52b420415b6f'), 'userId': 'ASC-47-36', 'status': 2, 'parkingSpaceNumber': 5, 'entryTime': 1701826536.041326}
r Inside vehicle park status
payment amount: $ 1.58
parking_space_list after removing vehicle: [1, 2, 3, 4, 6, 7, 8, 9, 10, 5]
o
c

```

Fig 16. Update the available parking space list

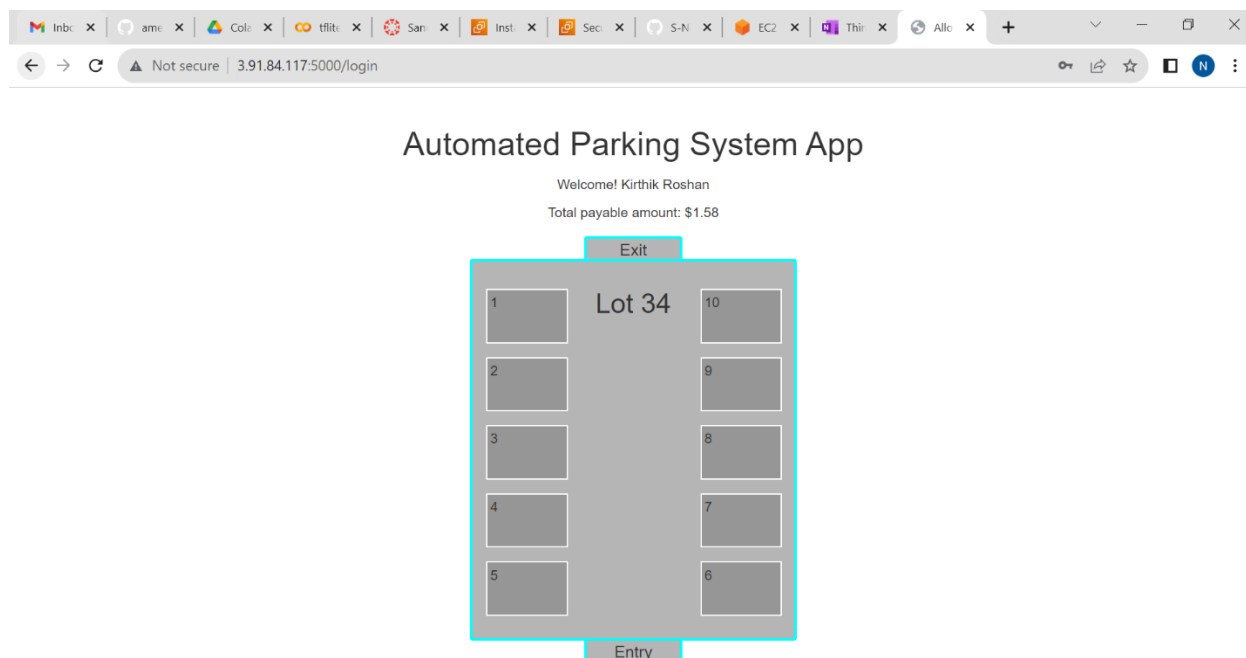
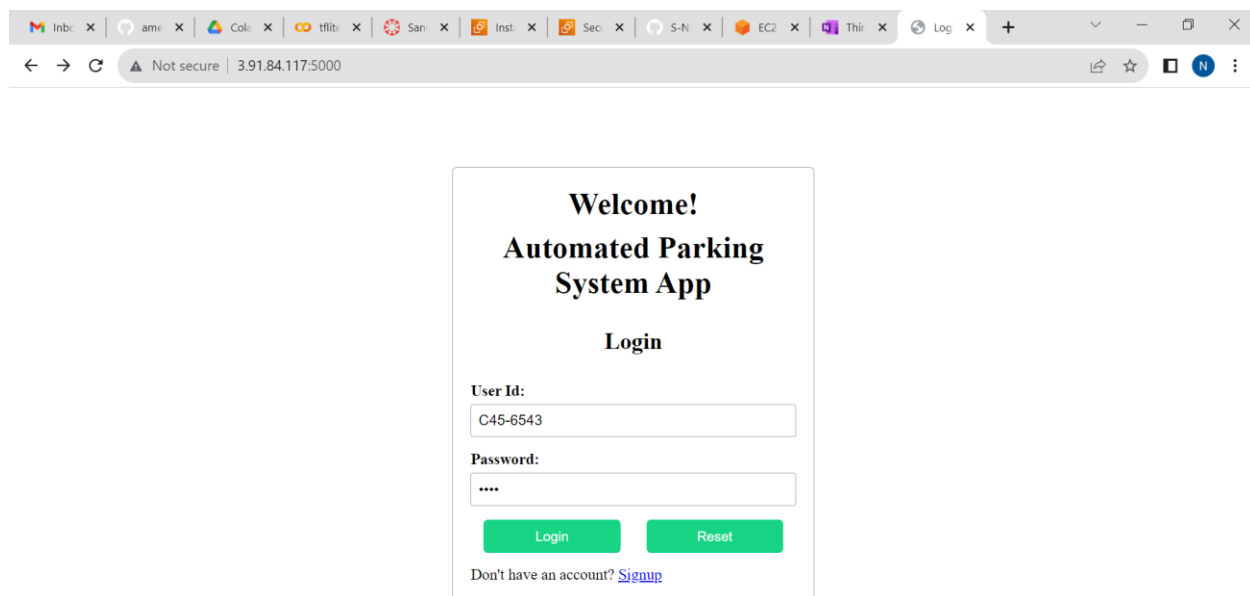
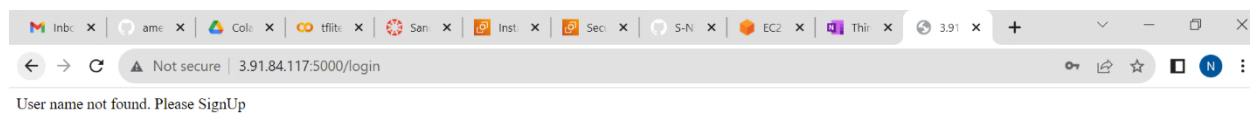


Fig 17. Updated UI during vehicle exit



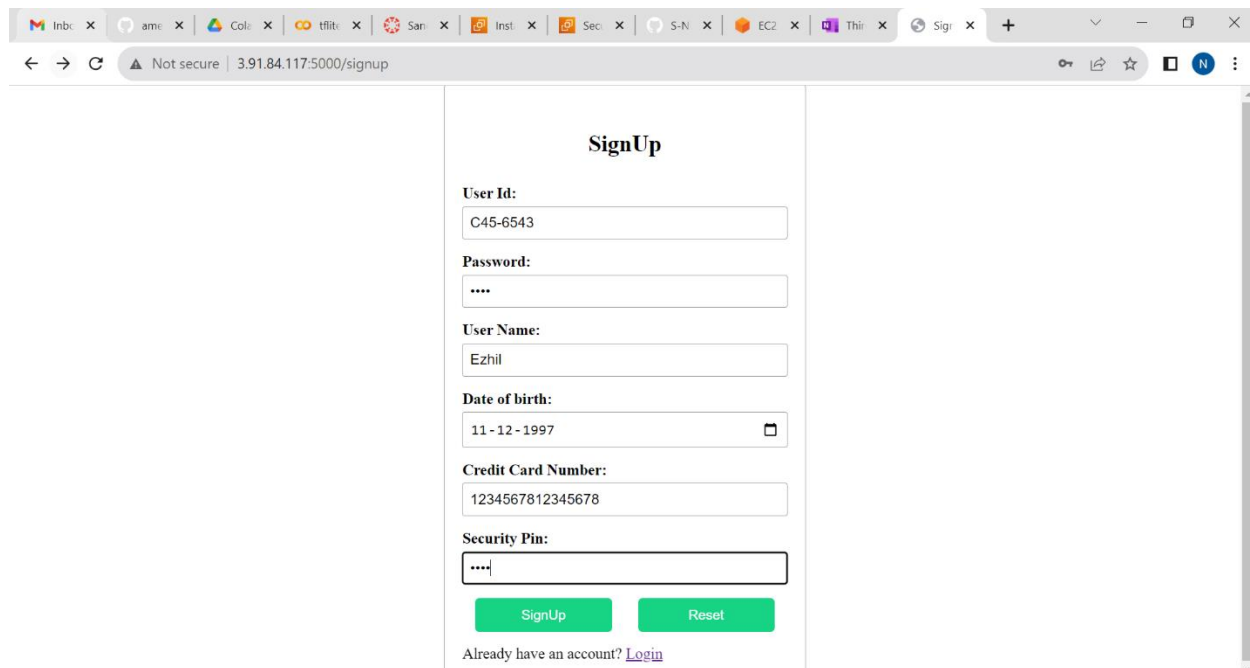
The screenshot shows a web browser window with multiple tabs open. The active tab is titled 'Log' and shows a login page for the 'Automated Parking System App'. The page has a white background with a light gray border. At the top, it says 'Welcome!' in bold. Below that, 'Automated Parking System App' is written in a larger, bold font. Underneath, 'Login' is centered. There are two input fields: 'User Id:' with the value 'C45-6543' and 'Password:' with four dots. Below the password field are two green buttons: 'Login' and 'Reset'. At the bottom, it says 'Don't have an account? [Signup](#)'.

*Fig 18.* Login with user data not already signed up



The screenshot shows the same web browser window as Fig 18, but the URL in the address bar is '3.91.84.117:5000/login'. The page content is mostly blank, with a message at the top that says 'User name not found. Please SignUp'.

*Fig 19.* Prompts for signup

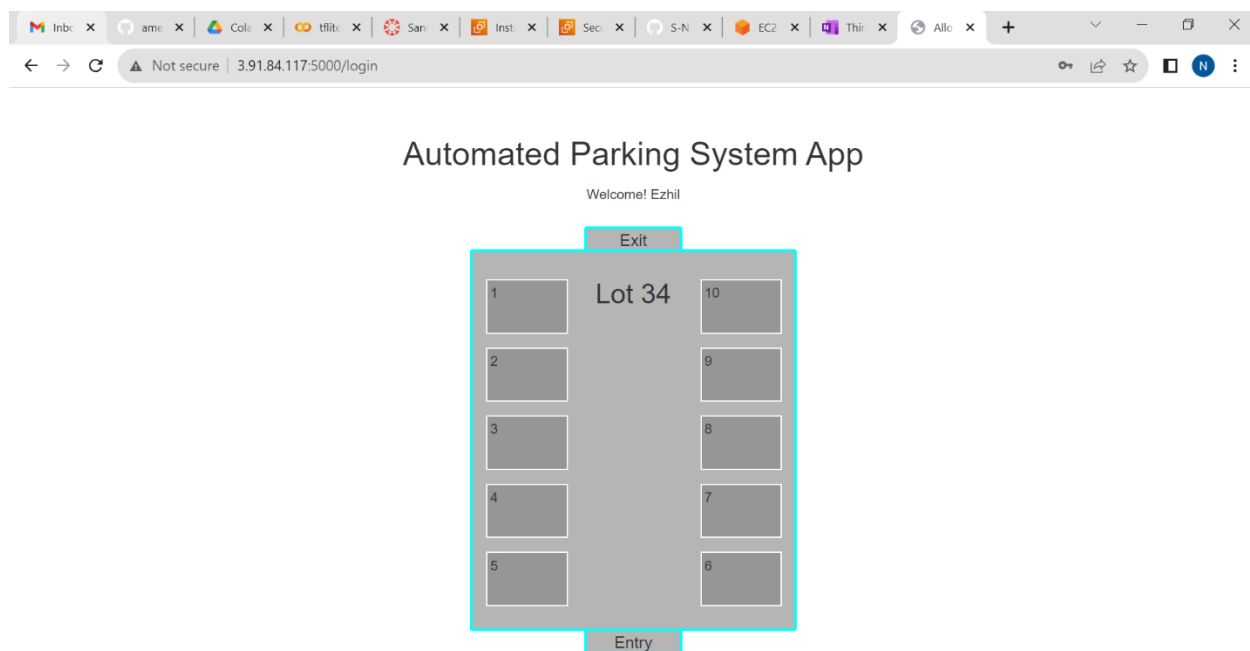


The screenshot shows a web browser window with the address bar displaying "Not secure | 3.91.84.117:5000/signup". The page title is "SignUp". The form contains the following fields and values:

- User Id: C45-6543
- Password: \*\*\*\*
- User Name: Ezhil
- Date of birth: 11 - 12 - 1997
- Credit Card Number: 1234567812345678
- Security Pin: \*\*\*\*

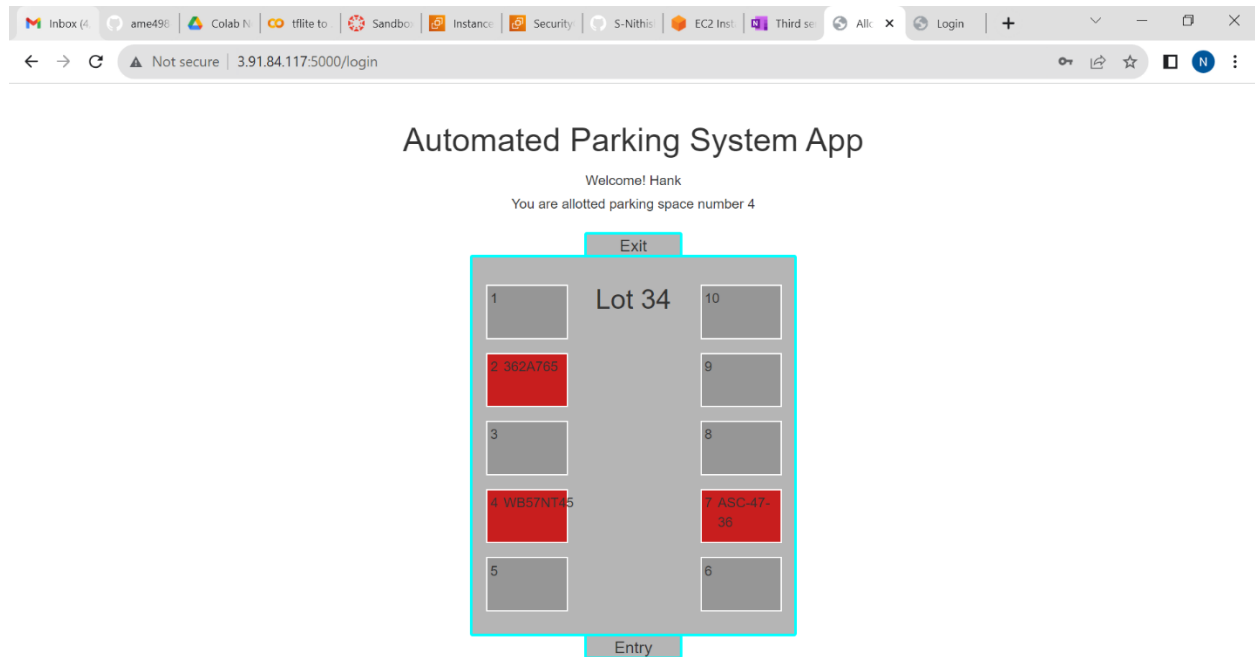
At the bottom of the form, there are two green buttons: "SignUp" and "Reset". Below the buttons, it says "Already have an account? [Login](#)".

Fig 20. Signup page



The screenshot shows a web browser window with the address bar displaying "Not secure | 3.91.84.117:5000/login". The page title is "Automated Parking System App". Below the title, it says "Welcome! Ezhil". The main content is a parking lot diagram labeled "Lot 34". The diagram shows a grid of 10 parking spaces, numbered 1 to 10. Spaces 1-5 are on the left, and spaces 6-10 are on the right. The diagram is enclosed in a cyan border. Above the diagram is an "Exit" button, and below it is an "Entry" button.

Fig 21. Logging in after signup



*Fig 22. Example of multiple vehicles parked*

## DESIGN GAP

1. The User Interface is not responsive for different devices.
2. Right now, I am using time module in python for finding the timestamp for entry and exit. This is not accurate as the DateTime module especially when run time of the program goes beyond one hour.
3. I planned to use an ultrasonic sensor to detect the presence of vehicle. But due to the unavailability, I am just using the key press of the button 'r' on the keyboard to start the detection and recognition algorithm.
4. I was planning to use a Servo motor to mimic the opening and closing of gate at entry and exit but due to limited time I have skipped that part.

## ACHIEVED GOALS

1. Deployment in Raspberry Pi
2. Implementation of detection and recognition of simple number plates.

3. Accessing Database using two different drivers – python, node.js
4. Deployment of node server in AWS.
5. Development of UI for displaying parking lot information.

#### FUTURE WORKS

1. Make the UI responsive.
2. Incorporate money transfer API.

#### Works Cited

Dolor, L.I. *Lorem ipsum dolor sit amet, consectetur adipiscing elit*, 1998. Print.

Dolor, L.I. *Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed diam nonummy nibh.*

*New York*: Columbia UP, 1998. Print.

Doe, R. John. *Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed diam nonummy nibh*, 1998. Print.