

# Autonomous Vehicle Rental



CMPE-281 –Cloud Technologies  
Dr. Jerry Gao

**Group 6**

**Team Members:**

Annapurna Ananya Annadatha

Guruvardhan Reddy Rajanala

Indhu Priya Reddem

Monica Lakshmi Mandapati

Vineeth Reddy Govind



# Agenda

- Introduction
- Purpose of the System
- Advantages and Disadvantages of AV
- Functional Services
- Architecture Diagram and Cloud System Design
- Auto Scaling and System Load Balance Design
- Team and their work
- Individual component implementation

# Introduction

An autonomous vehicle is one that can drive itself from a starting point to a predetermined destination in autopilot mode using various in-vehicle technologies and sensors.

It is a vehicle, capable of sensing its environment and moving safely with little or no human input.

Autonomous driving technologies have become one of the promising applications because of their potentials to enhance transportation efficiency, such as, automated guided vehicles (AGV) for goods delivery in warehouses, and self-driving cars sharing the loads for drivers.



# Purpose of the system

---

- The purpose of our system is:
  - To provide an autonomous vehicle booking system for customers using real time data from CARLA Simulator.
  - To design an efficient cloud-based car booking system to ensure that end users can seamlessly access the application

# Advantages of Autonomous Vehicles

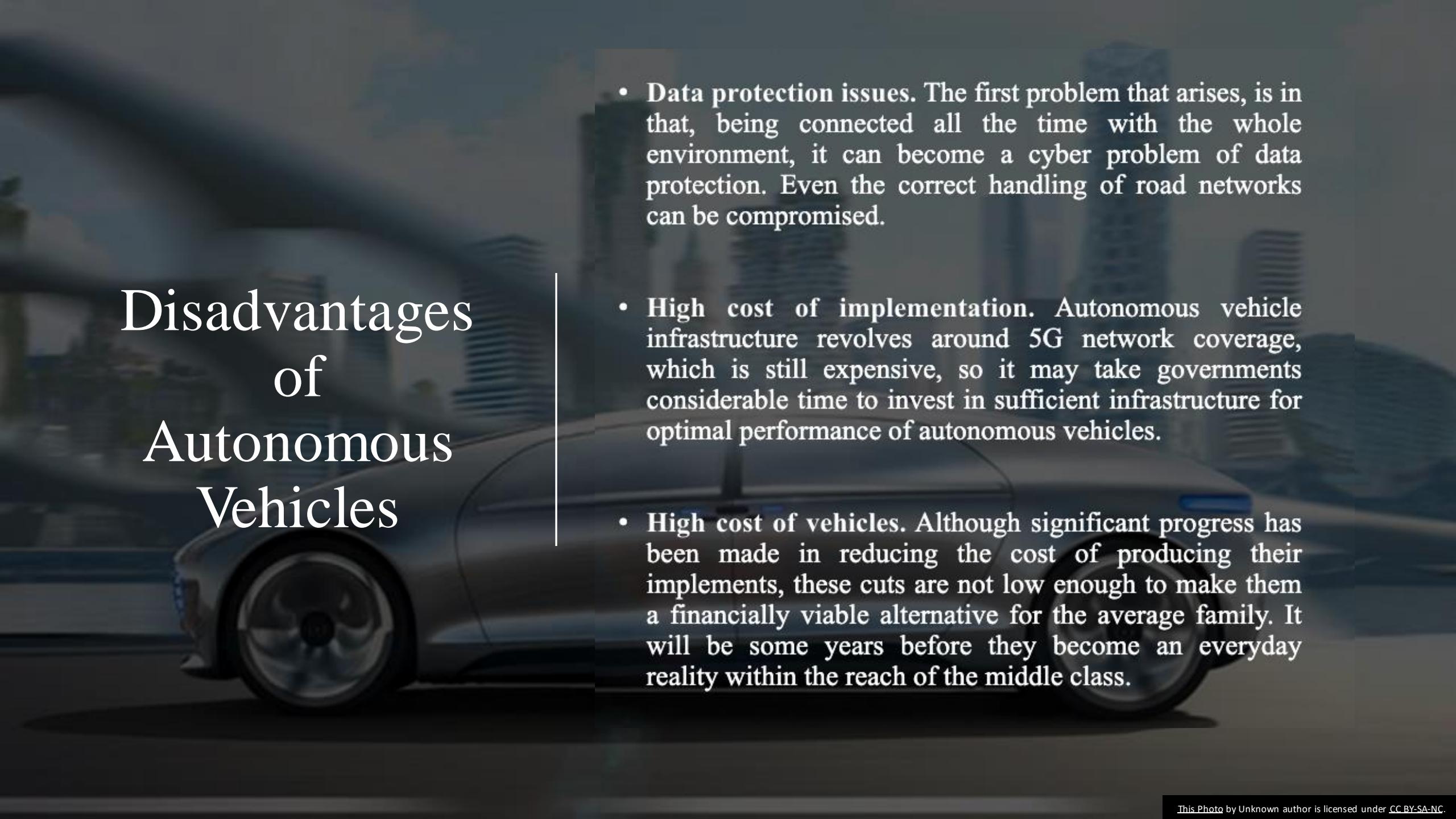
- **360° vision:** Autonomous vehicles possess the ability to view the environment in a 360° range, twice as much as humans, who have a viewing angle of only 180° horizontally.
- **Reduced accidents:** Thanks to 360° vision and vehicles being interconnected with each other and in constant communication, accidents will be significantly reduced. Although (at least initially) accidents will not be reduced to zero, they will be much less than accidents caused by human driving.



# Advantages of Autonomous Vehicles



- **Higher traffic efficiency:** Although it is estimated that their speed in big cities will be lower, their traffic efficiency will be higher.
- **Access to the disabled and people with reduced mobility:** Thanks to the fact that the automobile will be autonomous and will require practically no human interaction for its operation, even people with visual or hearing disabilities will be able to have one, i.e., they will become inclusive.
- **Sustainable vehicles:** It is expected that these vehicles will operate based on clean energy, so carbon and greenhouse gas emissions will be practically zero.



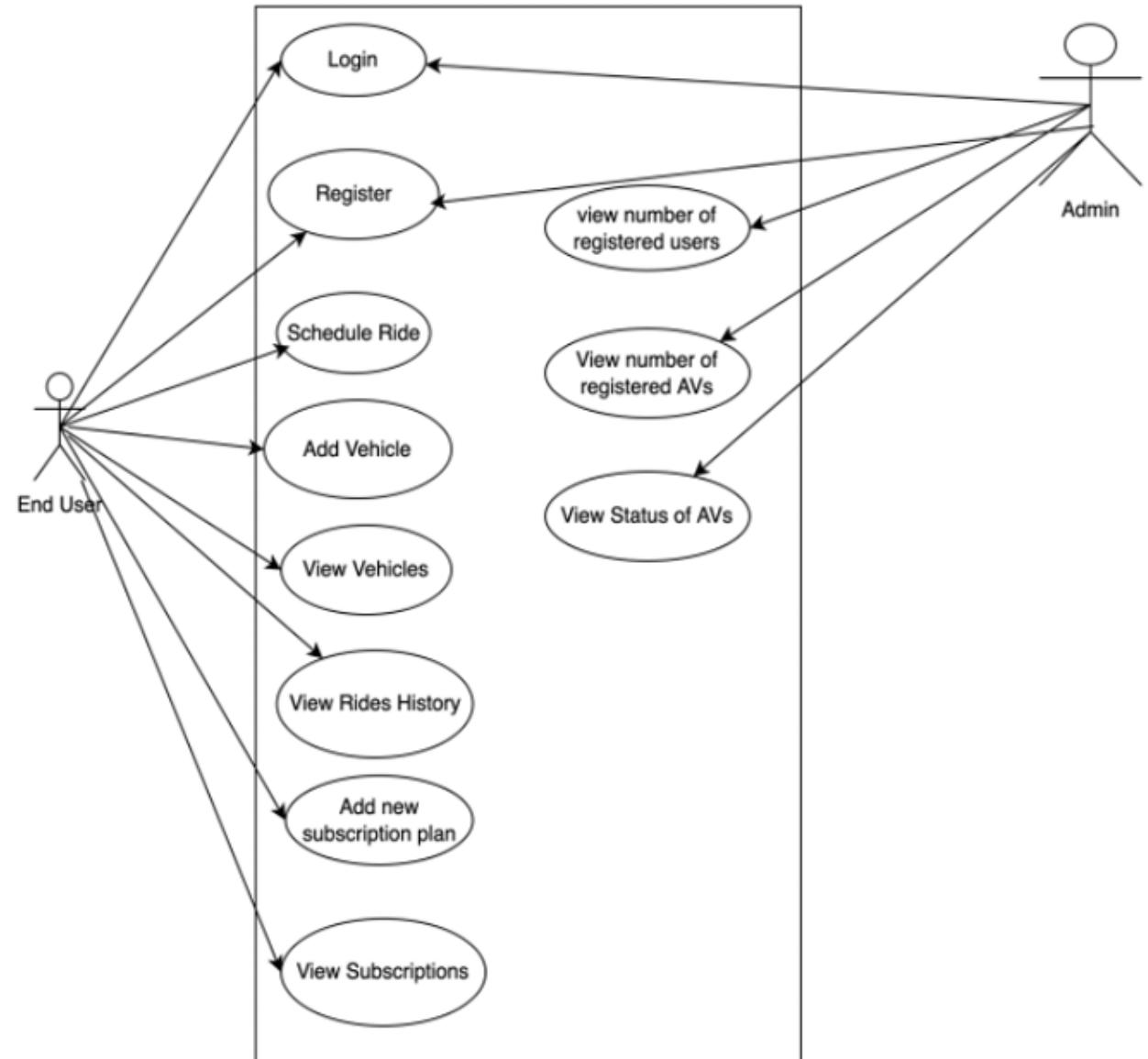
# Disadvantages of Autonomous Vehicles

- **Data protection issues.** The first problem that arises, is in that, being connected all the time with the whole environment, it can become a cyber problem of data protection. Even the correct handling of road networks can be compromised.
- **High cost of implementation.** Autonomous vehicle infrastructure revolves around 5G network coverage, which is still expensive, so it may take governments considerable time to invest in sufficient infrastructure for optimal performance of autonomous vehicles.
- **High cost of vehicles.** Although significant progress has been made in reducing the cost of producing their implements, these cuts are not low enough to make them a financially viable alternative for the average family. It will be some years before they become an everyday reality within the reach of the middle class.

# Functional Services

---

- Our System comprises of two types actors - Admin and End Users.
- Use case diagram:



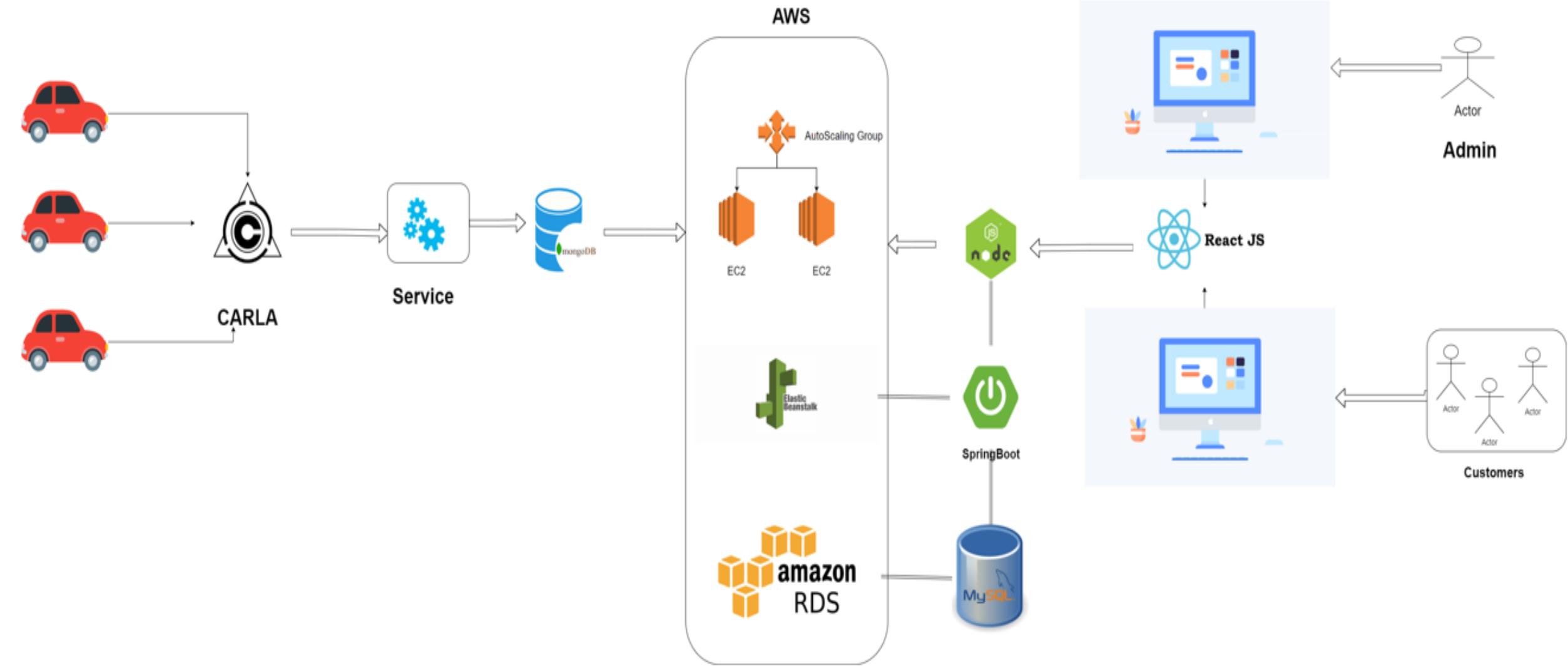
# Admin User Responsibilities

Function	Description
Register	Register to the application by giving Name,Username,Password
Login	Login to the application by giving username,password
Admin Dashboard	To view status of AVs, To view number of registered users, To view number of registered vehicles

# End User Responsibilities

Function	Description
Register	Register to the application by giving Name,username,password
Login	Login to the application by giving Username,Password
Add Vehicle	End user can add vehicle by giving vehicleid, vehicle color, vehicle model, vehicle mileage, passenger space
My Vehilces	End user can able to view all the vehciles added by him
Schedule a ride	End user can schedule a ride by entering pickup and drop off locations
Rides history	End user can view the rides history
Add plan	End user can add subscription plan

# Architecture Diagram



# Components of System Architecture

---

- **CARLA Simulator:**

It is an open-source simulator for Autonomous vehicle driving research. It provides different API's that allows us to retrieve data from the Autonomous Vehicle surroundings.

- **User Interface:**

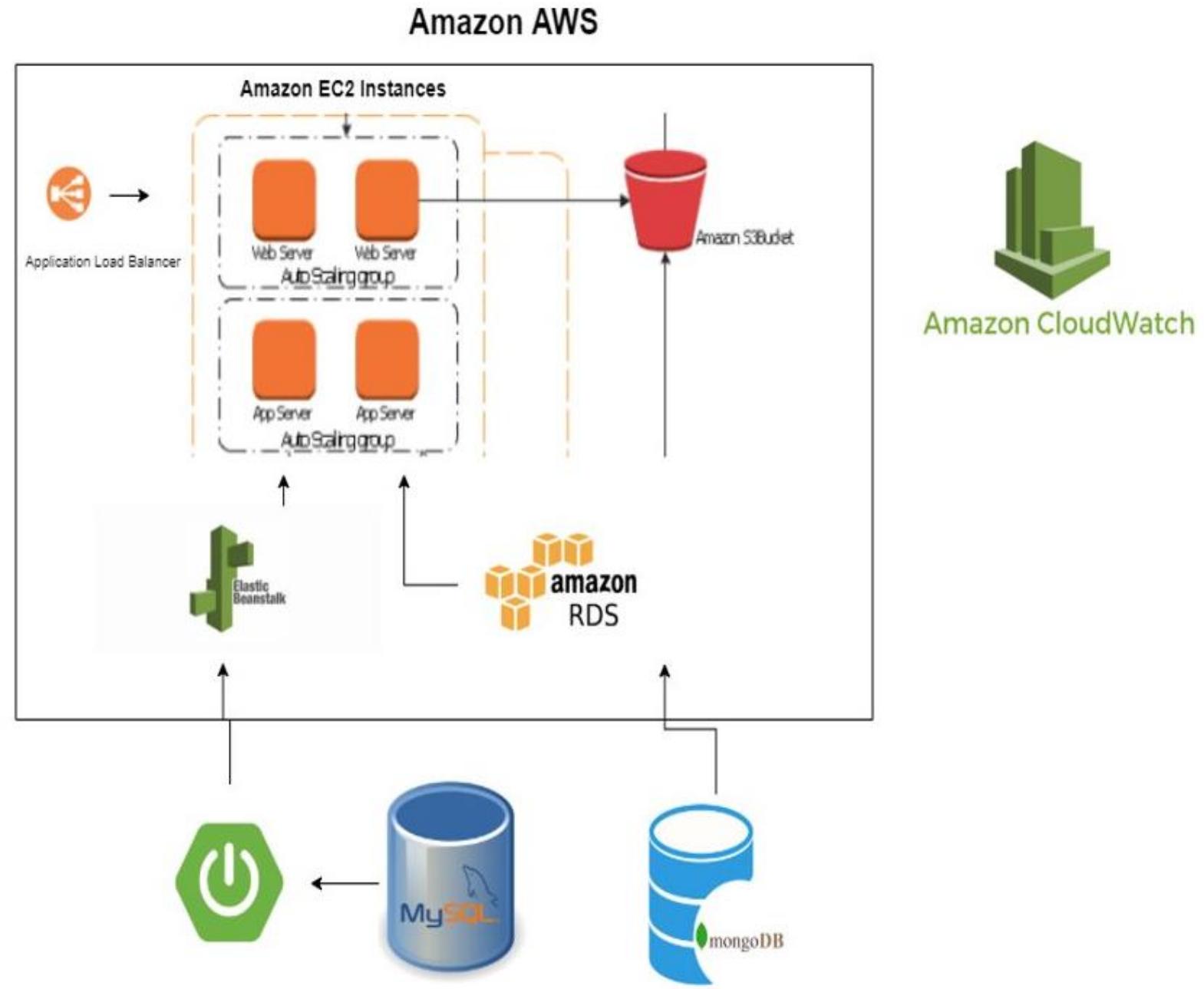
We are using **ReactJS** for the frontend of the application and **NodeJS** for the backend of the application. Customers can register, login, add membership plan, add vehicles, book a ride and track the car from the User Interface and Admins can monitor the Registered Av vehicles data and information.

- **Databases:**

**MongoDB:** The CARLA Simulator sends massive amounts of data, so we are using a NoSQL database to handle all the sensor data that is sent by the simulator.

**MySQL:** We are using MySQL- a Relational database to store the user's data, rides history and the membership plan information.

# Cloud System Design



# Components of Cloud System Design

- **Amazon Remote Database Service:** To access the database from anywhere, we have deployed the database on AWS-RDS which is web service running on cloud designed to simplify the setup and operation. To connect the node application to MySQL database, MySQL middleware is used.
- **Amazon Elastic BeanStalk:** We have used SpringBoot application to configure restful api's to fetch data from MySQL database. To access the java application remotely we have deployed the jar file onto Elastic BeanStalk.
- **Amazon EC2:** We have used Amazon EC2 instance to deploy NodeJS and react application. We cloned the application and installed the dependencies on EC2 instance by running npm install command. Once the application cloned, we start both frontend and backend servers. The application will be deployed on the generated publicIP and can be accessed using the same.
- **Amazon Cloud Watch:** CloudWatch collects monitoring and operational data in the form of logs, metrics, and events, and visualizes it using automated dashboards so you can get a unified view of your AWS resources, applications, and services that run on AWS and on premises. It helps us to monitor the AWS applications that are deployed on cloud.



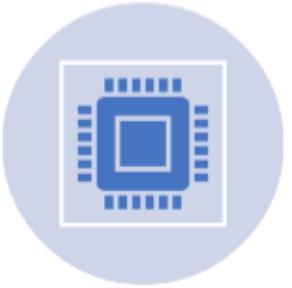
# Auto Scaling



Auto Scaling monitors applications and automatically adjusts capacity to maintain steady, predictable performance at the lowest possible cost.



Using AWS Auto Scaling, it's easy to setup application scaling for multiple resources across multiple services in minutes.

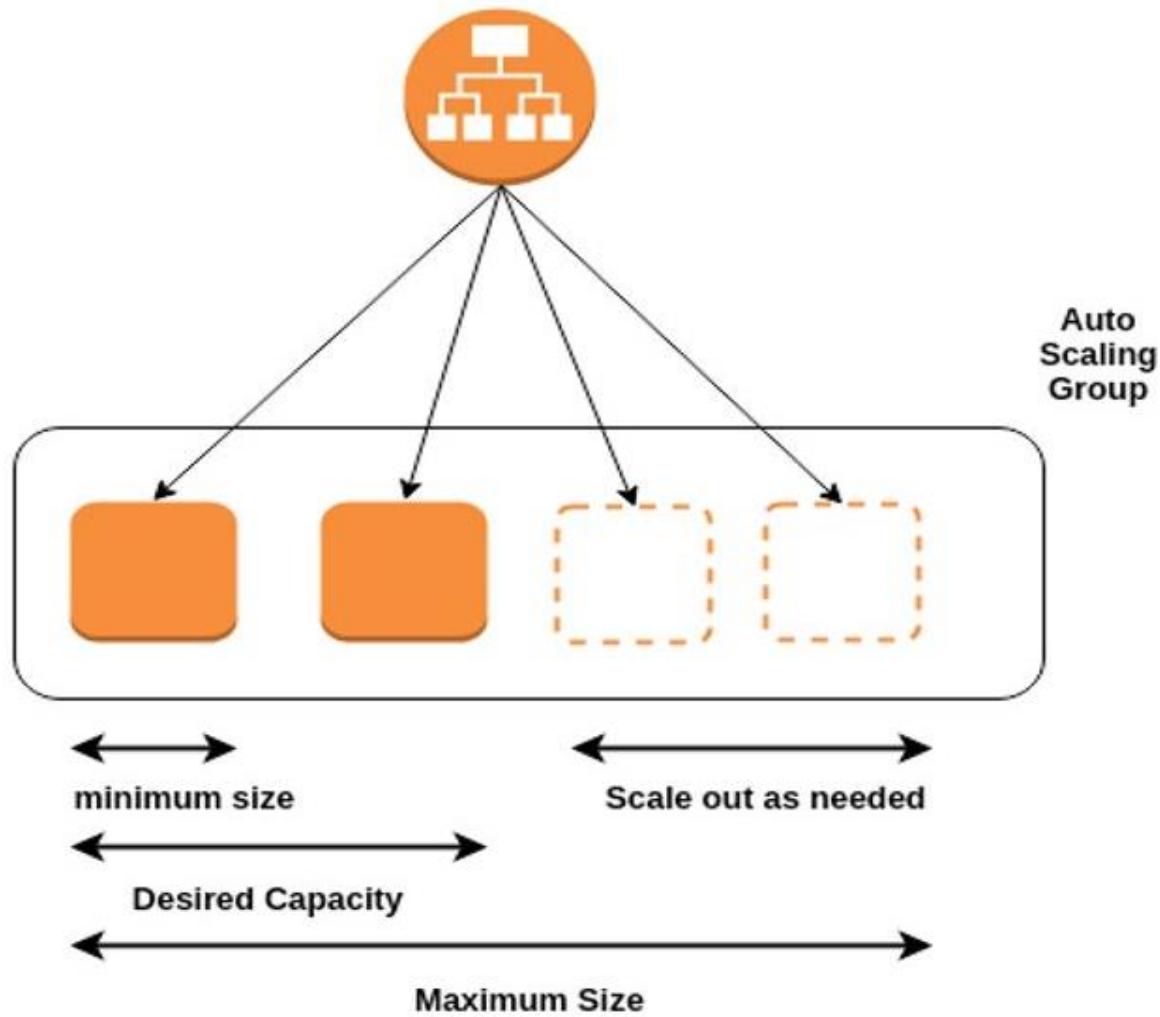


AWS Auto Scaling makes scaling simple with recommendations that allows us to optimize performance, costs, or balance between them.



With AWS Auto Scaling, applications always have the right resources at the right time.

# Amazon Auto Scaling





# System Load Balance Design

We have used Application Load balancer (ALB) service provided by AWS in front of our web and application servers. This is responsible for maintaining load among different availability zones our servers are set in. Whenever a request comes in at Application Load Balancer, it analyzes the rules provided by the listener and directs it accordingly to a target group. The targets can be registered by IP address of the EC2 instance auto scaling groups.



## Application Load Balancer



EC1      EC2

Group 1 Picture

Us-east -1a

10.0.1.0/24

Target group 1 Picture



EC3      EC4

Group 2 Video

Us-east -1b

10.0.3.0/24

Target group 2 Video

# **Team and their Work**

---

**Guruvardhan Reddy Rajanala:**  
Component #0 – Edge-based (or simulated) autonomous vehicles (Carla-based simulator)

---

**Vineeth Reddy Govind :** Component #1 – AV-based database management component(s)

---

**Monica Lakshmi Mandapati :** Component #2 – Remote online AV tracking and controlling

---

**Annapurna Ananya Annadatha :** Component #3 – AV service connectivity supporting rental AVs on a cloud

---

**Indhu Priya Reddem:** Component #4 – User service management

Component #0 : Edge based(Or Simulated) Autonomous Vehicles (CARLA-based Simulator)

## Functions Of CARLA

Our Edge-based simple autonomous vehicles will use the following components:

Data from sensors is pushed into MongoDB, supporting the following:

- In the dashboard, show the user statistics about the AV.
- Display AV Sensor Data

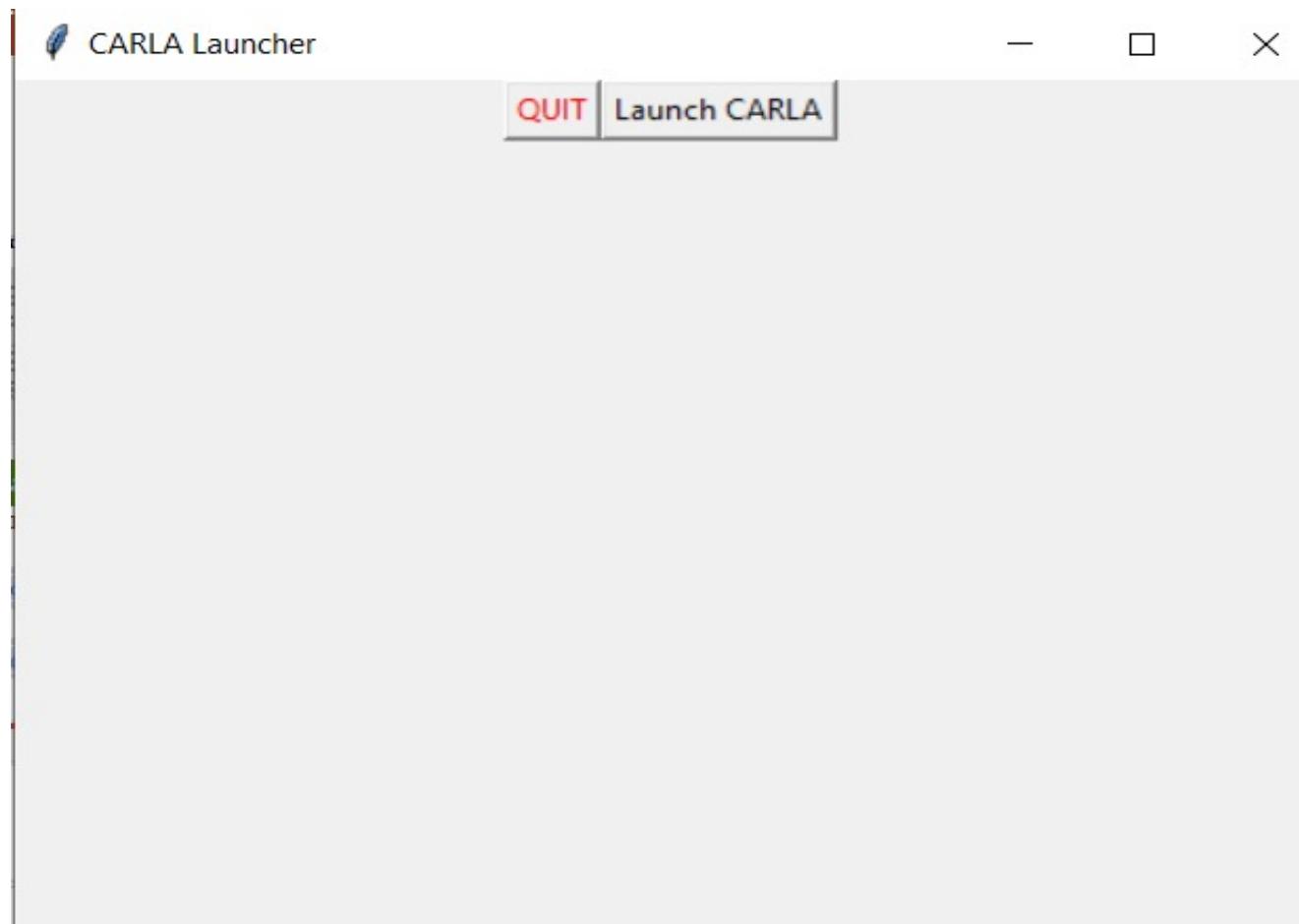


CARLA Simulator Interface



# Autonomous Vehicle Spawn

Button to interact from UI to  
CARLA



# **Component #1 – AV-based database management component(s)**

- Database 1: RDMS SQL database
- RDBMS SQL database is being used to store the user information, ride information, payment information and booking information.
- Following are the tables:
  - 1. user\_data,
  - 2. rides\_data,
  - 3. payment\_info,
  - 4. booking\_info,
  - 5. vehicle\_info

# MySQL Database

bandicam 2021-11-14 20-09-54-075.mp4

File Edit View Query Database Server Tools Scripting Help

Navigator

SCHEMAS

group6

Tables

plandetails

hibernate\_sequence

userdetails

vehicledetails

vehiclide

Views

Stored Procedures

Functions

SYS

Administration Schemas

Information

Table: plandetails

Columns:

plandid	bigint PK
amount	int
email	varchar(255)
enddate	varchar(255)
paymenttype	varchar(255)
startdate	varchar(255)
tag	varchar(255)

Query 1 plandetails vehiclide plandetails plandetails

1 \* SELECT \* FROM group6.plandetails;

Result Grid | Filter Rows | Edit | Export/Import | Wrap Cell Contents | Result Grid | Form Editor

plandid	amount	email	enddate	paymenttype	startdate	tag
2	20	ananya@gmail.com	12 Dec 2021	Credit Card	14 Nov 2021	current
5	20	indhu@gmail.com	12 Dec 2021	PayPal	14 Nov 2021	current
12	20	m@gmail.com	13 Dec 2021	Credit Card	15 Nov 2021	current
13	20	m@gmail.com	13 Dec 2021	Credit Card	15 Nov 2021	current
14	20	monu@gmail.com	13 Dec 2021	Credit Card	15 Nov 2021	current
16	20	test@gmail.com	13 Dec 2021	Debit Card	15 Nov 2021	current
18	20	user1@gmail.com	13 Dec 2021	Credit Card	15 Nov 2021	current

plandetails 1 ×

Action Output

Time	Action	Message	Duration / Fetch
1 15:54:07	select * from group6.vehicledetails LIMIT 0, 1000	8 row(s) returned	0.094 sec / 0.000 sec
2 15:54:25	select * from group6.userdetails LIMIT 0, 1000	8 row(s) returned	0.094 sec / 0.000 sec
3 19:50:34	SELECT * FROM group6.plandetails LIMIT 0, 1000	9 row(s) returned	0.140 sec / 0.000 sec
4 19:53:17	SELECT * FROM group6.vehiclide LIMIT 0, 1000	9 row(s) returned	0.094 sec / 0.000 sec
5 20:00:53	SELECT * FROM group6.plandetails LIMIT 0, 1000	9 row(s) returned	0.500 sec / 0.000 sec
6 20:00:58	SELECT * FROM group6.plandetails LIMIT 0, 1000	9 row(s) returned	0.464 sec / 0.000 sec

Object Info Session

Automatic context help is disabled.  
Use the toolbar to manually get help for the current caret position or to toggle automatic help.

# MongoDB

MongoDB is a NoSQL database i.e information is stored in the form of documents.

This database is used to store all the information from vehicle sensors.

Following is the information that is stored in this database

1. Frame data
2. Timestamp
3. Vehicle speed
4. Vehicle direction
5. Vehicle name
6. Ride destination
7. Vehicle location
8. Vehicle GNSS data
9. Vehicle state

+ Create Database

Namespaces

sensor\_frame\_data

frame

## sensor\_frame\_data.frame

COLLECTION SIZE: 2.31MB TOTAL DOCUMENTS: 5138 INDEXES TOTAL SIZE: 216

Find

Indexes

Schema Anti-Patterns

Aggregation

FILTER { field: 'value' }

### QUERY RESULTS 1-20 OF MANY

```
_id: ObjectId("6191b9d409193fae512214a2")
time: 1636940244762996700
vehicle: Array
  0: "Server"          0 FPS"
  1: "Client"          0 FPS"
  2: ""
  3: "Vehicle: Diamondback Century"
  4: "Map: Town10HD_Opt"
  5: "Simulation time: 0:03:13"
  6: ""
  7: "Speed: 10 km/h"
  8: "Heading: -180° 5"
  9: "Location: (-15.4, 133.7)"
  10: "GNSS: (-0.001281, -0.000147)"
  11: "Height: 0 m"
  12: ""
```

# Component #4- User Service Management

- The graphical user interface (GUI) is utilized by the user or AV owner to operate the service, track status, charge, and perform basic taxi services. The data from the sensors is extracted with ROS, saved in a database, and then shown on the dashboard for users.
- This component provides GUI for the Users to Login, Register, Manage their profile, Books a ride, View Payment Details, Manage AV Sensor Data.
- Components also include User Accounts, AV Sensor Accounts. User Accounts for Manage and provide access to the user information. AV Sensor Accounts for manage and provide AV information to the admin

# Register

React App Not secure | 18.225.6.194:3000/register Incognito Google one.SJSU AVRental Login Register

Register

Name  
user89

Username  
user89@gmail.com

Password  
.....

Register

Already have an account? Please login here [Login](#)



# Login

React App

Not secure | 18.225.6.194:3000/login

Google one.SJSU Incognito Reading list

AVRental Login Register

Login

Username

Password

Not a user already? Click here [Register](#)



# Add Vehicle

React App Not secure | 18.225.6.194:3000/myVehicles/addVehicle

Google one.SJSU Incognito Reading list

AVRental Schedule a Ride View Rides History My Plan Add a Vehicle My Vehicles Dashboard Welcome, user12345 Log Out

Add your Vehicle

Vehicle ID: vehicle789

Vehicle Color: red

Vehicle Make: tata

Vehicle Model: xuv

Vehicle Mileage: 45

Vehicle Passengers Space: 3

Vehicle Location (City): san jose

Submit



# My Vehicles

React App

Not secure | 18.225.6.194:3000/myVehicles

Google one.SJSU Incognito Reading list

AVRental Schedule a Ride View Rides History My Plan Add a Vehicle My Vehicles Dashboard Welcome, user12345 Log Out

## My Vehicles

Vehicle Id	Vehicle Color	Vehicle Make	Vehicle Model	Vehicle Mileage	Passenger Space	Service Status	Current Status	Current Location	Road Service
vehicle789	red	tesla	suv	13	3	Inactive	Idle	san jose	No Service

Delete Vehicle

# My Plan

React App

Not secure | 18.225.6.194:3000/myplan

Google one.SJSU

AVRental Schedule a Ride View Rides History My Plan Add a Vehicle My Vehicles Dashboard

Welcome, user12345 Log Out

My Plan

Schedule a Ride View Rides History My Plan Add a Vehicle My Vehicles Dashboard

Welcome, user12345 Log Out

## Current Plan

No active current plan at the moment

Add Plan

# Add Plan

React App

Not secure | 18.225.6.194:3000/myPlan/addPlan

Google one.SJSU Incognito Reading list

AVRental Schedule a Ride View Rides History My Plan Add a Vehicle My Vehicles Dashboard Welcome, user12345 Log Out

## Add your Plan

Billing Details

**Cycle:** 15 Nov 2021 - 13 Dec 2021

**Amount:** \$40

Credit Card  
 Debit Card  
 Amazon Pay

**Pay**

# Schedule Ride

React App    Not secure | 18.225.6.194:3000/mySchedule    Incognito    Google one.SJSU    Reading list

Schedule a Ride    View Rides History    My Plan    Add a Vehicle    My Vehicles    Dashboard    Welcome, user12345    Log Out

Schedule your ride

VID  
vehicle789

Pick-up Location  
san jose

Number of passengers  
3

Drop-off Location  
santa clara

**Book Ride**

# Ride History

React App

Not secure | 18.225.6.194:3000/myRides

Google one.SJSU

Schedule a Ride View Rides History My Plan Add a Vehicle My Vehicles Dashboard

Welcome, user12345 Log Out

## User12345's Ride History

Vehicle License Plate #	Origin Location	Vehicle Passenger Space	Destintaion Location	Date/Time
vehicle789	san jose	3	santa clara	2021-11-14 20:52:6

# **Component #2 – Remote online AV tracking and controlling**

---

This component provides the Graphical User interface for tracking and controlling of the Autonomous Vehicle.

The purpose of this component is to track and monitor the moving status, service states, location information, road service records and sensor states of the Autonomous Vehicle. The sensor data which is generated by the CARLA is stored in a NoSQL database – MongoDB and is transmitted onto the cloud. The user can view the moving status, service status and road service status of the booked vehicle on the User Dashboard. The user can also view the Additional sensor data of his vehicle.

The admin user will have the access to view the details of the AV vehicles and also can log into to MongoDB to view the detailed sensor information.

# User Dashboard

- The User Dashboard consists of information about the active and registered cars along with the registered user's information.
- Moving state of the car – idle/Moving/Moved
- Service state of the car – active, Inactive
- Road Service – In Service/ No Service
- Location details – Latitude and Longitude
- GNSS details,
- Heading Towards
- Speed/Velocity
- Simulated Time of the AV

## My Dashboard

Vehicle License Plate

vehicle789

Current State

Moved

Service Status

Active

Road Service

Active

- The different sensors that are used to fetch the sensor data are
- **Camera sensors:** which fetches the heading towards parameter details.
- **GNSS sensors:** which fetches the geolocation, latitude and longitude of the vehicle.
- **RADAR Sensors:** which fetches velocity, acceleration of the vehicle.
- IMU Sensors:** which fetches the Simulation time of the AV.

AV/Rental      Schedule a Ride      View Rides History      My Plan      Add a Vehicle      My Vehicles      Dashboard      Welcome, Indhureddem      Log Out

Additional Sensor Information

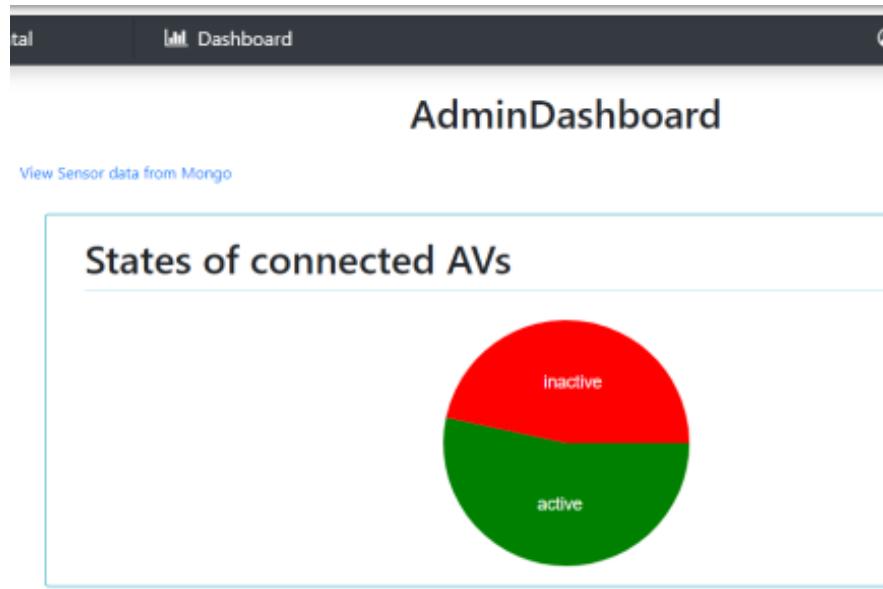
Simulation time: 1:18:27      Speed: 26 km/h

Heading: 165° SE      Location: ( 82.9, 13.4)

GNSS: (-0.000123, 0.000737)

# Admin Dashboard

- The Admin Dashboard consists of information about the active and registered cars along with the registered user's information. It comprises of the following data about the Autonomous vehicle:
- Chart which represent Active and Inactive Vehicles.
- The Admin Dashboard also has a button to navigate to MongoDB



AVRental

Dashboard

Welcome, ad

## List of connected AVs

AV Number	AV Owner	AV Make	AV Model	AV Mileage	Road Service	AV State
AR2123	testuser@gmail.com	Audi	A3	1000	Active	Moved
CALI	test@gmail.com	Ford	Ford	5000	Active	Moved
MON12	m@gmail.com	honda	H4	200	In Service	Moved
MRNG123	mrg@gmail.com	Tesla	Tesla	200	Active	Active
us123	indhu@gmail.com	Audi	A3	500	No Service	Moved

Dashboard

vehicle123	user78@gmail.com	Suzuki	xuv	12	Active
VEHICLE2	ananya@gmail.com	Tesla	A3	2000	No Service
Vehicle4	user1@gmail.com	Audi	S3	1000	Active
Vehicle5	user2@gmail.com	Audi	S3	1000	Active
vehicle789	user12345@gmail.com	tesla	suv	13	Active

Registered AV Users

Connected AV Vehicles

22	15
----	----



# Component #3: AV service connectivity supporting rental AVs on a cloud

- The purpose of the component#3 AV service connectivity supporting rental AVs on a cloud is to provide connectivity between different components of the rental AVs on the cloud - CARLA simulator, cloud DB, and GUI of the autonomous vehicle cloud.
- This component aims to achieve the following types of communication between the components.
  1. AV cloud management connectivity protocol.
  2. AV service management connectivity protocol.

# Implementation of the component

- 1) MongoDB implementation
- 2) MySQL and Remote Database Service
- 3) Amazon Elastic Bean Stalk
- 4) Amazon EC2
- 5) Amazon CloudWatch

+ Create Database

Namespaces

sensor\_frame\_data

frame

sensor\_frame\_data.frame

COLLECTION SIZE: 2.31MB TOTAL DOCUMENTS: 5138 INDEXES TOTAL SIZE: 216

Find Indexes Schema Anti-Patterns Aggregation

FILTER { field: "value" }

QUERY RESULTS 1-20 OF MANY

```
_id: ObjectId("6191b9d409193fae512214a2")
time: 1636940244762996700
vehicle: Array
  0: "Server"          0 FPS"
  1: "Client"          0 FPS"
  2: ""
  3: "Vehicle: Diamondback Century"
  4: "Map: Town10HD_Opt"
  5: "Simulation time: 0:03:13"
  6: ""
  7: "Speed: 10 km/h"
  8: "Heading: -180° S"
  9: "Location: (-15.4, 133.7)"
  10: "GNSS: (-0.001281, -0.000147)"
  11: "Height: 0 m"
  12: ""
```

## MongoDB sensor data

## MongoDB connection from NodeJS

```
const mongoose = require("mongoose");

mongoose
  .connect("mongodb+srv://root: mongoPassword@cluster0.a5yk3.mongodb.net/sensor_frame_data?retryWrites=true&w=majority" , {
    // retry to connect
    reconnectTries: 1,
    // wait 5 seconds before retryMon
    reconnectInterval: 5000,
  })
  .then(() => console.log("Connected to MongoDB"))
  .catch((err) => console.log("Failed to connect to Database"));
```

# AWS Remote Database Services

Amazon RDS

X

cmpe281group6

Modify Actions ▾

Dashboard

Databases

Query Editor

Performance Insights

Snapshots

Automated backups

Reserved instances

Proxies

Subnet groups

Parameter groups

Option groups

Custom engine versions

Events

Event subscriptions

Recommendations 3

Summary

DB identifier	CPU	Status	Class
cmpe281group6	5.17%	Available	db.t2.micro
Role	Current activity	Engine	Region & AZ
Instance	10 Connections	MySQL Community	us-east-2a

Connectivity & security Monitoring Logs & events Configuration Maintenance & backups Tags

Connectivity & security

Endpoint & port	Networking	Security
Endpoint	Availability Zone	VPC security groups
cmpe281group6.clqyaakkwpfa.us-east-2.rds.amazonaws.com	us-east-2a	default (sg-d8eb3b95)
Port	VPC	Active
		Publicly accessible

Connection to RDS from NodeJS app

```
const pool = mysql.createConnection({
  host: 'cmpe281group6.clqyaakkwpfa.us-east-2.rds.amazonaws.com',
  user: 'root',
  port: 3306,
  password: 'Password12345',
  database: 'group6'
});

pool.connect((err) => {
  if(err){
    console.log("failed to connect"+err);
  }
  if(!err){
    console.log("connected to mysql database");
  }
});

if (!config.get("jwtPrivateKey")) {
  console.log("FATAL ERROR: jwtPrivateKey is not defined");
  process.exit(1);
}
```

## Elastic Beanstalk



Environments

Applications

Change history

avrental.group6

Application versions

Saved configurations

Avrentalgroup6-env

Go to environment

Configuration

the action, add permission to your instance profile according to Enhanced health authorization in the *Amazon Elastic Beanstalk Developer Guide*.

### Avrentalgroup6-env

Avrentalgroup6-env.eba-jyjhezut.us-east-2.elasticbeanstalk.com (e-pa9ghp4xuh)

Application name: avrental.group6

Refresh

Actions ▾

#### Health



Ok

Causes

#### Running version

avrental.group6-source

Upload and deploy

#### Platform



Corretto 11 running on 64bit

Amazon Linux 2/3.2.7

Change

# AWS Elastic Bean Stalk interface

# AWS EC2 Connection Details

EC2 > Instances > i-058d7a9643791106b > Connect to instance

### Connect to instance Info

Connect to your instance i-058d7a9643791106b (CmpeAvrental) using any of these options

[EC2 Instance Connect](#) [Session Manager](#) [SSH client](#) [EC2 Serial Console](#)

**Instance ID**

[i-058d7a9643791106b \(CmpeAvrental\)](#)

1. Open an SSH client.
2. Locate your private key file. The key used to launch this instance is cmpe281groupproject.pem
3. Run this command, if necessary, to ensure your key is not publicly viewable.  
`chmod 400 cmpe281groupproject.pem`
4. Connect to your instance using its Public DNS:  
`ec2-18-225-6-194.us-east-2.compute.amazonaws.com`

**Example:**

`ssh -i "cmpe281groupproject.pem" ec2-user@ec2-18-225-6-194.us-east-2.compute.amazonaws.com`

aws Services ▾ cloud9 indupriya.reddem Ohio Support ▾

New EC2 Experience Tell us what you think

EC2 Dashboard EC2 Global View Events Tags Limits

Instances (1/2) [Info](#)

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone
AmazonLinux	i-02be36b9f29c70a0c	Stopped	t2.micro	-	No alarms	us-east-2a
<b>CmpeAvrental</b>	<b>i-058d7a9643791106b</b>	<b>Running</b>	<b>t2.micro</b>	<b>2/2 checks passed</b>	<b>No alarms</b>	<b>us-east-2c</b>

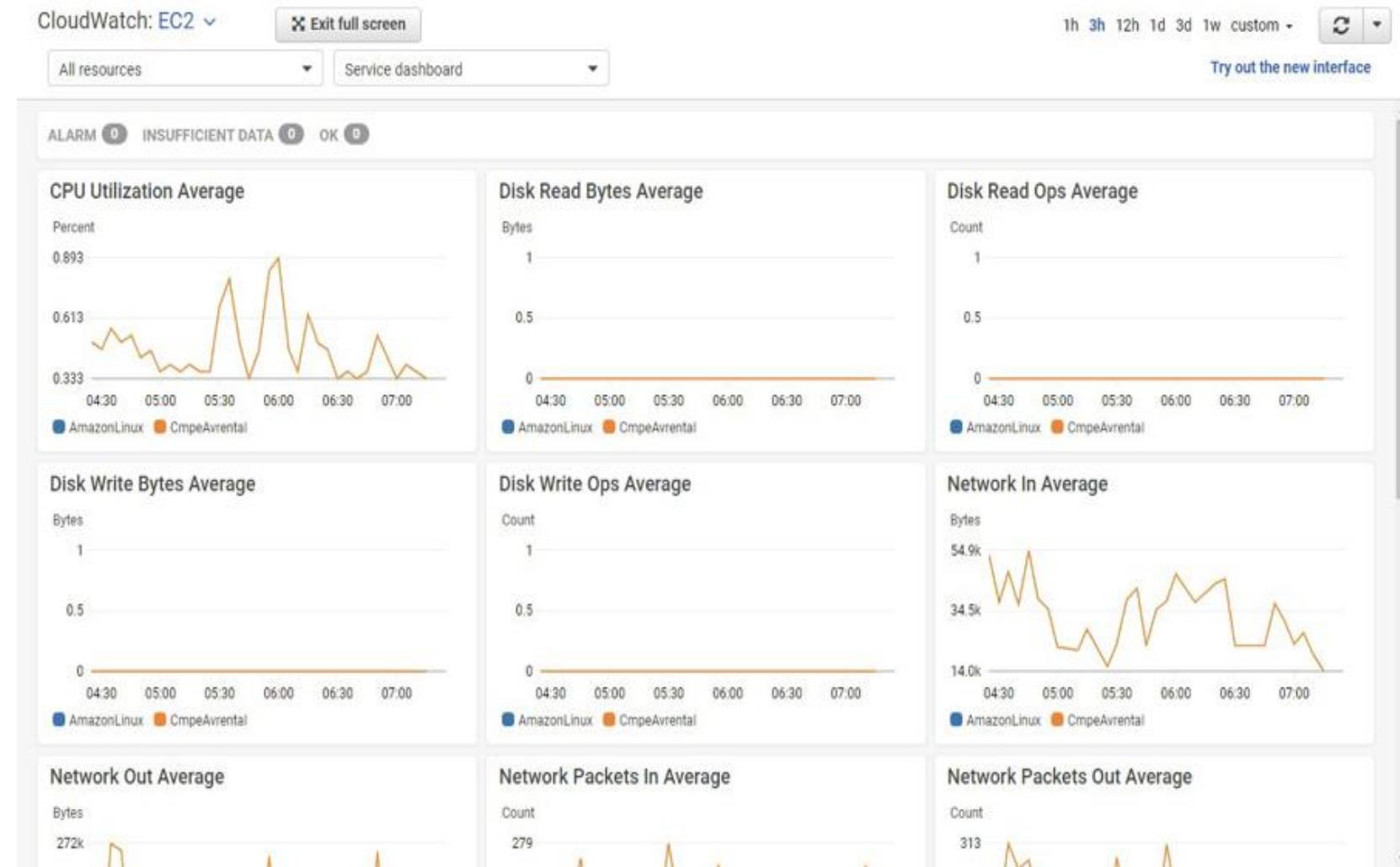
**Instance: i-058d7a9643791106b (CmpeAvrental)**

**Instance summary** Info

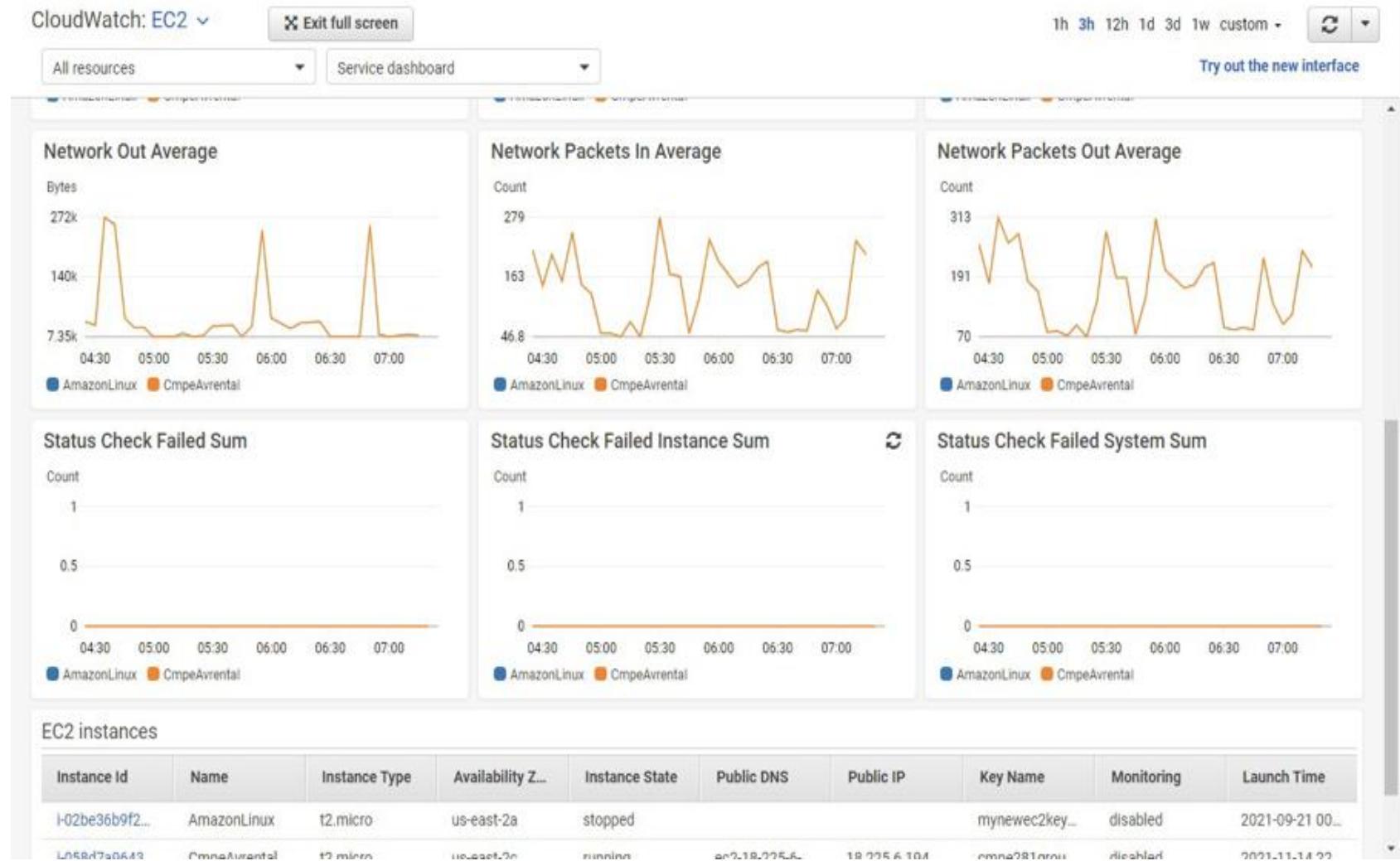
Instance ID	Public IPv4 address	Private IPv4 addresses
i-058d7a9643791106b (CmpeAvrental)	18.225.6.194   open address	172.31.35.250
IPv6 address	Instance state	Public IPv4 DNS
-	Running	ec2-18-225-6-194.us-east-2.compute.amazonaws.com   open address
Private IPv4 DNS	Instance type	Elastic IP addresses
ec2-18-225-6-194.us-east-2.compute.amazonaws.com	t2.micro	-

Feedback English (US) ▾ © 2008 - 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved. Privacy Policy Terms of Use Cookie preferences

- AWS Cloud Watch Graphs



- AWS Cloud Watch Graphs



# Extra Features

- Added an interaction from UI (Button) to simulate CARLA.
- Fetching additional data like Simulation time, Speed, Location and GNSS

THANK YOU

