



## **DATABASE & CLOUD SYSTEM (BERR2243)**

### **WEEK 4 TASK**

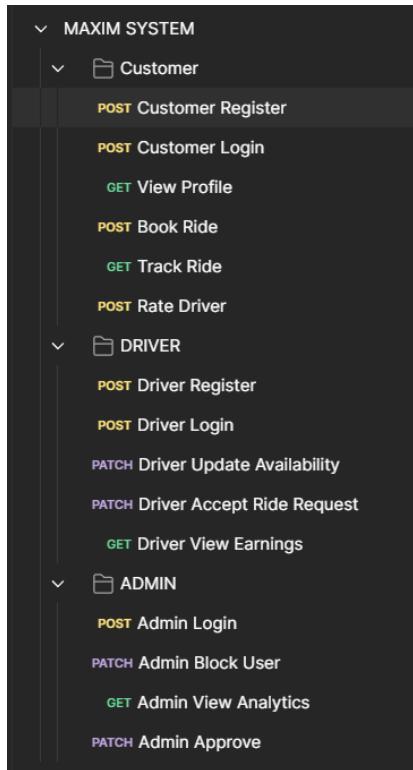
### **MAXIM SYSTEM**

#### **Group H**

|                       |   |
|-----------------------|---|
| <b>GROUP MEMBERS</b>  | 1. RIGNES A/P TAMIL VANAN (B122320069)<br>2. DAMIA NAZURAH BINTI ABDUL RIZAN (B122320020) |
| <b>COURSE</b>         | BERR 3 S5   |
| <b>LECTURE'S NAME</b> | DR SOO  |

## 1. Translate use cases into RESTful endpoints

I. First create a new collection and click add folder for customer, driver and admin. Then Name the folder as Customer, Driver and Admin and add request inside each Folders.



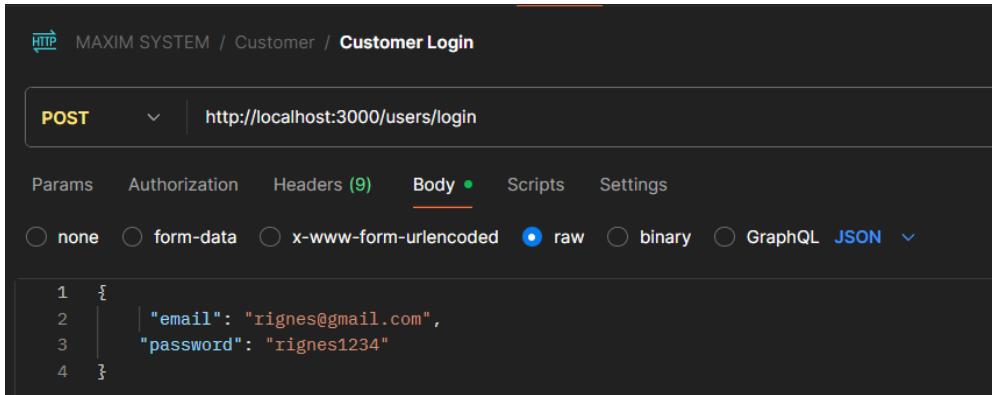
### Customer folder:

- Choose POST and rename the request as Register Customer. Then create body JSON

```
POST http://localhost:3000/users/register
```

| Params  | Authorization                   | Headers (8)                                 | Body                                 | Scripts                      | Settings                      |
|---|---------------------------------|---|--------------------------------------|------------------------------|-------------------------------|
| <input type="radio"/> none  | <input type="radio"/> form-data | <input type="radio"/> x-www-form-urlencoded | <input checked="" type="radio"/> raw | <input type="radio"/> binary | <input type="radio"/> GraphQL |
| <pre>1  { 2    "name": "Rignes", 3    "email": "rignes@gmail.com", 4    "password": "rignes1234", 5    "phone": "01116245676" 6 }</pre> |                                 |   |                                      |                              |                               |

- Choose POST and rename the request as Customer Login and create body JSON



HTTP MAXIM SYSTEM / Customer / **Customer Login**

**POST** http://localhost:3000/users/login

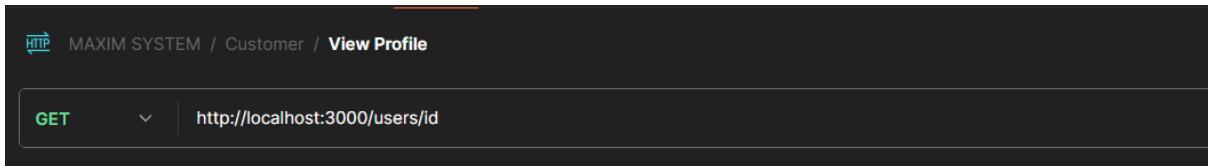
Params Authorization Headers (9) **Body** • Scripts Settings

none  form-data  x-www-form-urlencoded  raw  binary  GraphQL **JSON** ▾

```

1 {
2   "email": "rignes@gmail.com",
3   "password": "rignes1234"
4 }
```

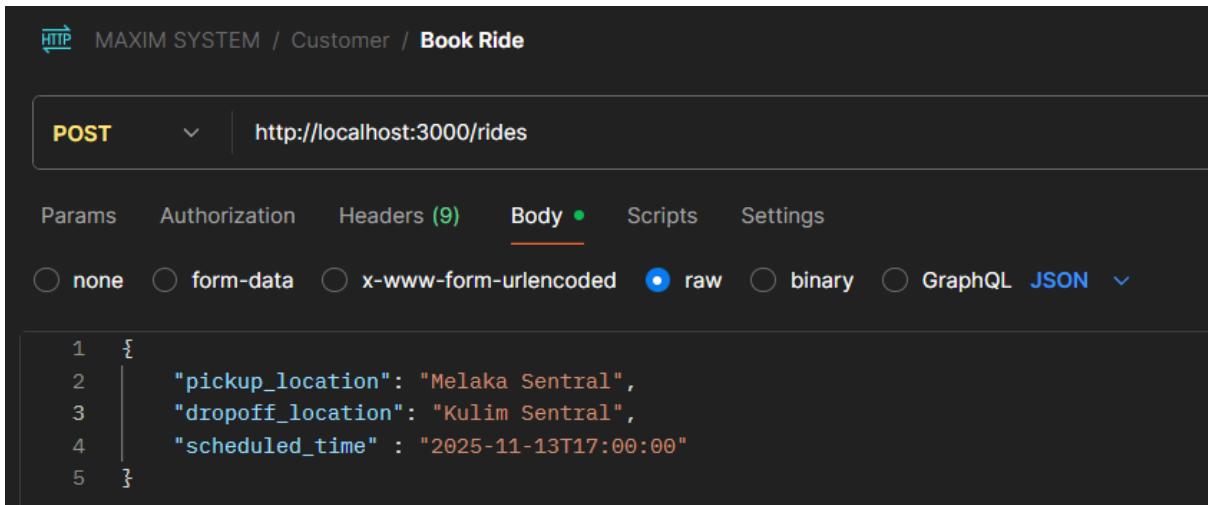
- Choose GET and rename the request as View Profile



HTTP MAXIM SYSTEM / Customer / **View Profile**

**GET** http://localhost:3000/users/id

- Create request Book Ride. Then create body JSON



HTTP MAXIM SYSTEM / Customer / **Book Ride**

**POST** http://localhost:3000/rides

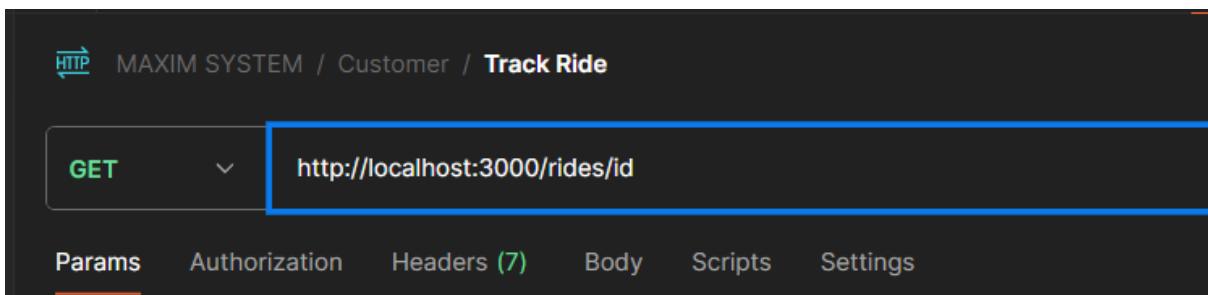
Params Authorization Headers (9) **Body** • Scripts Settings

none  form-data  x-www-form-urlencoded  raw  binary  GraphQL **JSON** ▾

```

1 {
2   "pickup_location": "Melaka Sentral",
3   "dropoff_location": "Kulim Sentral",
4   "scheduled_time" : "2025-11-13T17:00:00"
5 }
```

- Create request Track Ride

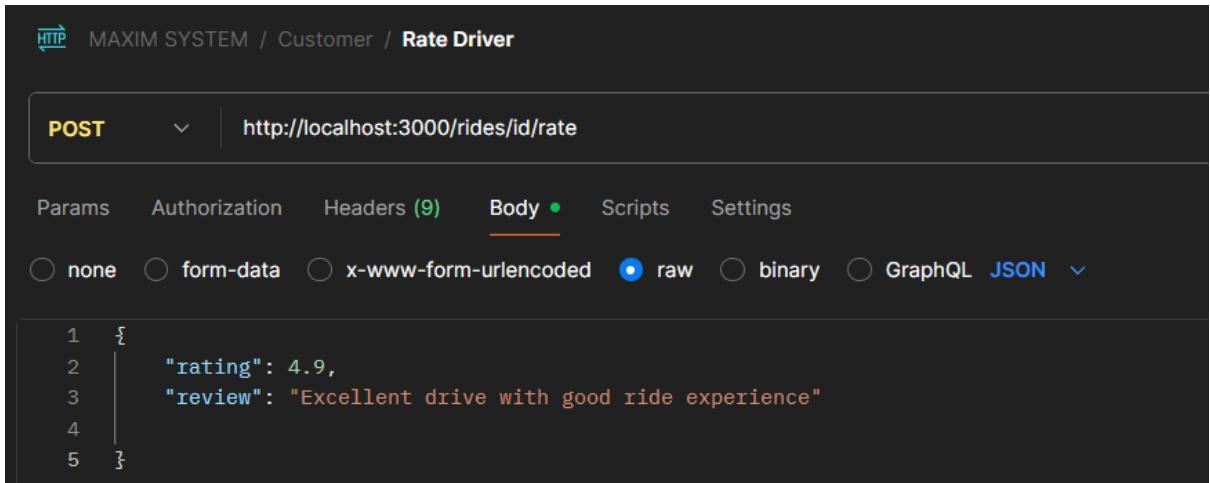


HTTP MAXIM SYSTEM / Customer / **Track Ride**

**GET** http://localhost:3000/rides/id

Params Authorization Headers (7) Body Scripts Settings

- Create request Rate Driver. Then create body JSON



MAXIM SYSTEM / Customer / Rate Driver

**POST** | <http://localhost:3000/rides/id/rate>

Params Authorization Headers (9) **Body** Scripts Settings

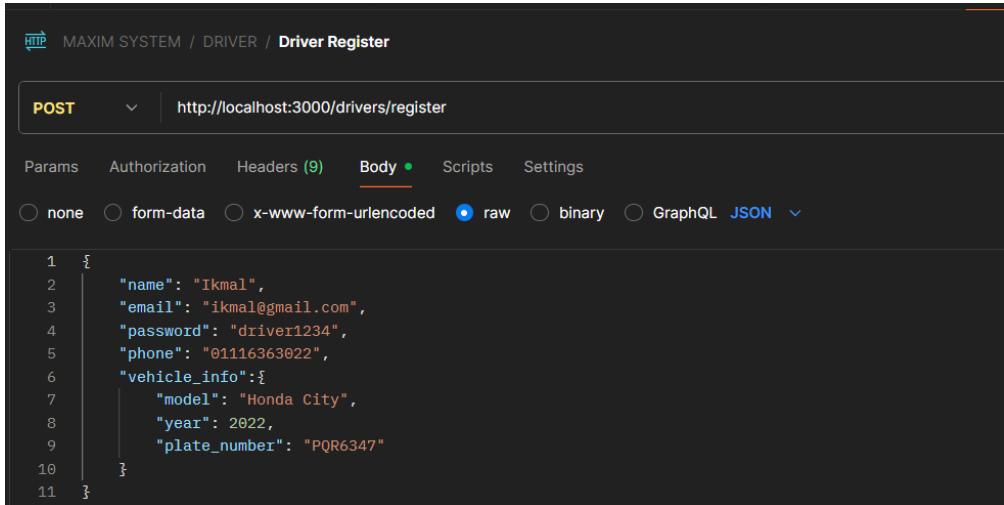
none  form-data  x-www-form-urlencoded  raw  binary  GraphQL **JSON**

```

1  {
2    "rating": 4.9,
3    "review": "Excellent drive with good ride experience"
4
5 }
```

### Driver Folder:

- Add request and rename as Driver Register. Create body JSON



MAXIM SYSTEM / DRIVER / Driver Register

**POST** | <http://localhost:3000/drivers/register>

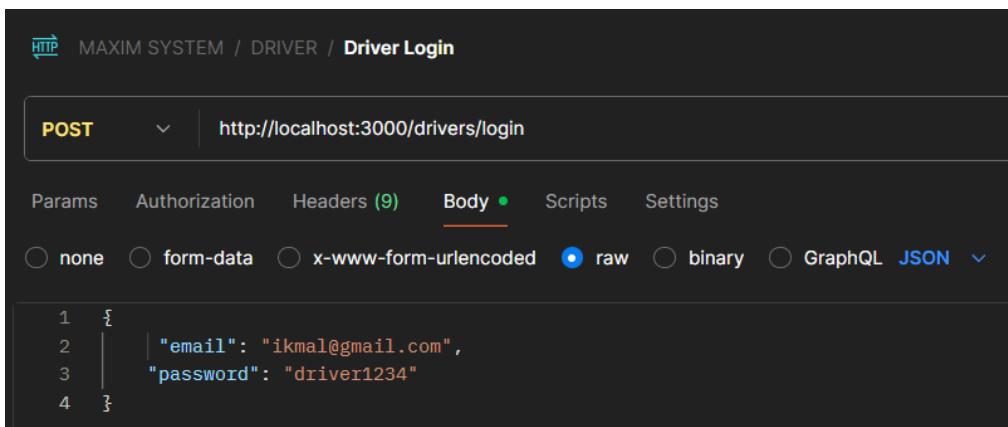
Params Authorization Headers (9) **Body** Scripts Settings

none  form-data  x-www-form-urlencoded  raw  binary  GraphQL **JSON**

```

1  {
2    "name": "Ikmal",
3    "email": "ikmal@gmail.com",
4    "password": "driver1234",
5    "phone": "01116363022",
6    "vehicle_info": {
7      "model": "Honda City",
8      "year": 2022,
9      "plate_number": "PQR6347"
10   }
11 }
```

- Add request and rename as Driver Login. Create body JSON



MAXIM SYSTEM / DRIVER / Driver Login

**POST** | <http://localhost:3000/drivers/login>

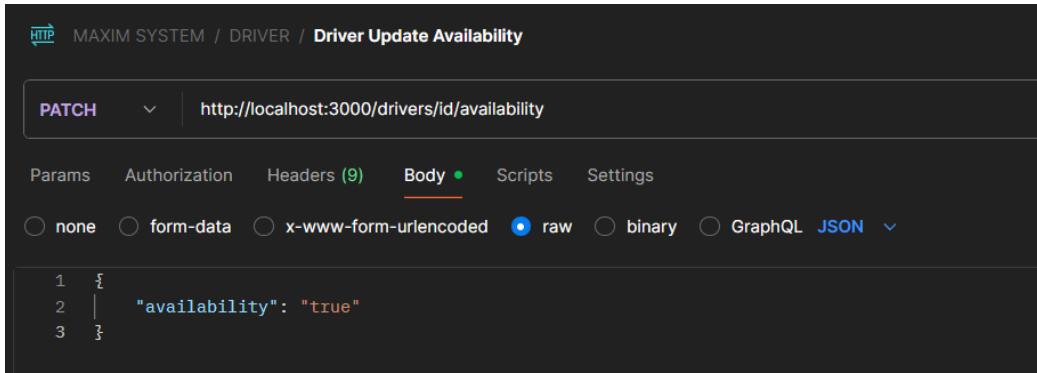
Params Authorization Headers (9) **Body** Scripts Settings

none  form-data  x-www-form-urlencoded  raw  binary  GraphQL **JSON**

```

1  {
2    "email": "ikmal@gmail.com",
3    "password": "driver1234"
4 }
```

- Add request and rename as Driver Update Availability. Then create body JSON



MAXIM SYSTEM / DRIVER / **Driver Update Availability**

**PATCH** | <http://localhost:3000/drivers/id/availability>

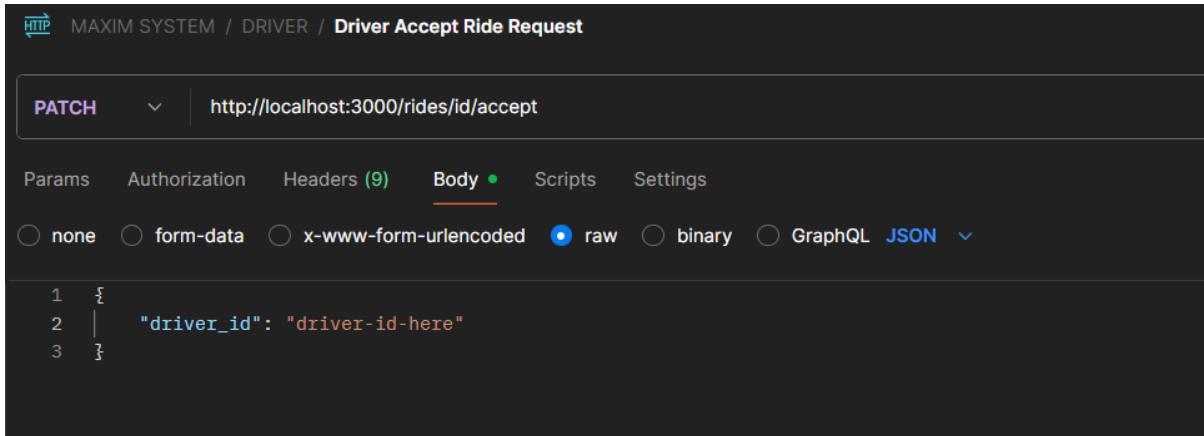
Params Authorization Headers (9) **Body** Scripts Settings

none  form-data  x-www-form-urlencoded  raw  binary  GraphQL **JSON**

```

1 {
2   "availability": "true"
3 }
```

- Add request and rename as Driver Accept Ride Request. Create body JSON



MAXIM SYSTEM / DRIVER / **Driver Accept Ride Request**

**PATCH** | <http://localhost:3000/rides/id/accept>

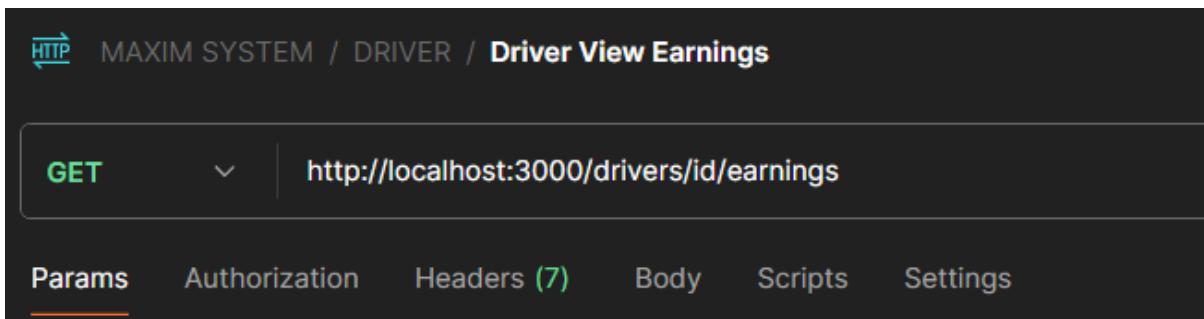
Params Authorization Headers (9) **Body** Scripts Settings

none  form-data  x-www-form-urlencoded  raw  binary  GraphQL **JSON**

```

1 {
2   "driver_id": "driver-id-here"
3 }
```

- Add request and rename as Driver View Earnings



MAXIM SYSTEM / DRIVER / **Driver View Earnings**

**GET** | <http://localhost:3000/drivers/id/earnings>

**Params** Authorization Headers (7) Body Scripts Settings

## Admin Folder:

- Add request and rename as Admin Login. Create body JSON

HTTP MAXIM SYSTEM / ADMIN / Admin Login

POST http://localhost:3000/admin/login

Params Authorization Headers (9) **Body** Scripts Settings

none  form-data  x-www-form-urlencoded  raw  binary  GraphQL **JSON**

```
1 {  
2   "username": "yuan@gmail.com",  
3   "password": "admin1234"  
4 }  
5
```

- Add request and rename as Admin Block User. Then create body JSON

HTTP MAXIM SYSTEM / ADMIN / Admin Block User

PATCH http://localhost:3000/admin/block/id

Params Authorization Headers (9) **Body** Scripts Settings

none  form-data  x-www-form-urlencoded  raw  binary  GraphQL **JSON**

```
1 {  
2   "reason": "Violation of terms and conditions"  
3 }
```

- Add request and rename as Admin View Analytics

HTTP MAXIM SYSTEM / ADMIN / Admin View Analytics

GET http://localhost:3000/admin/analytics

**Params** Authorization Headers (7) Body Scripts Settings

- Add request and rename as Admin Approve. Create body JSON

The screenshot shows the POSTMAN interface. The URL is `http://localhost:3000/admin/approve/driveid`. The method is set to `PATCH`. The `Body` tab is active, showing the following JSON payload:

```

1  {
2   |   "approved": "true"
3   }

```

Part 2: Implement the RESTful APIs

### **Task 1: Develop the API to the index.js**

1. Continue from the last week exercise, develop the API designed in previous Part

I. Edit WEEK4.js

THE CODE :

```

const express = require('express');
const { MongoClient, ObjectId } = require('mongodb');
const port = 3000;

```

```

const app = express();
app.use(express.json());

```

```

let db;

```

```

async function connectToMongoDB() {
  const uri = "mongodb://localhost:27017";
  const client = new MongoClient(uri);

```

```
try {
    await client.connect();
    console.log("Connected to MongoDB!");
    db = client.db("testDB");
} catch (err) {
    console.error("Error:", err);
}
connectToMongoDB();

app.listen(port, () => {
    console.log(`Server running on port ${port}`);
});

//Customer

// Customer Registration
app.post('/users/register', async (req, res) => {
    try {
        const result = await db.collection('users').insertOne(req.body);
        res.status(201).json({ id: result.insertedId });
    } catch (err) {
        res.status(400).json({ error: "Registration failed" });
    }
});

// Customer Login
```

```
app.post('/users/login', async (req, res) => {
  const { email, password } = req.body;
  try {
    const user = await db.collection('users').findOne({ email, password });
    if (!user) return res.status(401).json({ error: "Invalid credentials" });
    res.status(200).json(user);
  } catch (err) {
    res.status(500).json({ error: "Login failed" });
  }
});

// View Profile

app.get('/users/:id', async (req, res) => {
  try {
    const user = await db.collection('users').findOne({ _id: new ObjectId(req.params.id) });
    if (!user) return res.status(404).json({ error: "User not found" });
    res.status(200).json(user);
  } catch (err) {
    res.status(400).json({ error: "Invalid user ID" });
  }
});

// Ride

// Create Ride (Book Ride)
app.post('/rides', async (req, res) => {
  try {
```

```
const result = await db.collection('rides').insertOne(req.body);
res.status(201).json({ id: result.insertedId });
} catch (err) {
  res.status(400).json({ error: "Invalid ride data" });
}
});

// Track Ride
app.get('/rides/:id', async (req, res) => {
  try {
    const ride = await db.collection('rides').findOne({ _id: new ObjectId(req.params.id) });
    if (!ride) return res.status(404).json({ error: "Ride not found" });
    res.status(200).json(ride);
  } catch (err) {
    res.status(400).json({ error: "Invalid ride ID" });
  }
});

// Rate Driver
app.post('/rides/:id/rate', async (req, res) => {
  const { rating, comment } = req.body;
  try {
    const result = await db.collection('rides').updateOne(
      { _id: new ObjectId(req.params.id) },
      { $set: { rating, comment } }
    );
    if (result.modifiedCount === 0) return res.status(404).json({ error: "Ride not found or already rated" });
  }
});
```

```
    res.status(200).json({ updated: result.modifiedCount });

} catch (err) {
    res.status(400).json({ error: "Invalid data" });

}

});

// Driver

// Driver Registration

app.post('/drivers/register', async (req, res) => {

try {

const result = await db.collection('drivers').insertOne(req.body);

res.status(201).json({ id: result.insertedId });

} catch (err) {

res.status(400).json({ error: "Registration failed" });

}

});

// Driver Login

app.post('/drivers/login', async (req, res) => {

const { email, password } = req.body;

try {

const driver = await db.collection('drivers').findOne({ email, password });

if (!driver) return res.status(401).json({ error: "Invalid credentials" });

res.status(200).json(driver);

} catch (err) {

res.status(500).json({ error: "Login failed" });

}

});
```

```
});

// Update Driver Availability
app.patch('/drivers/:id/availability', async (req, res) => {
  try {
    const result = await db.collection('drivers').updateOne(
      {_id: new ObjectId(req.params.id)},
      { $set: { availability: req.body.availability } }
    );
    if (result.modifiedCount === 0) return res.status(404).json({ error: "Driver not found" });
    res.status(200).json({ updated: result.modifiedCount });
  } catch (err) {
    res.status(400).json({ error: "Invalid driver ID or data" });
  }
});

// Accept Ride Request
app.patch('/rides/:id/accept', async (req, res) => {
  try {
    const result = await db.collection('rides').updateOne(
      {_id: new ObjectId(req.params.id)},
      { $set: { driver_id: req.body.driver_id, status: "accepted" } }
    );
    if (result.modifiedCount === 0) return res.status(404).json({ error: "Ride not found" });
    res.status(200).json({ updated: result.modifiedCount });
  } catch (err) {
    res.status(400).json({ error: "Invalid ride ID or data" });
  }
});
```

```
    }

});

// View Earnings

app.get('/drivers/:id/earnings', async (req, res) => {

  try {

    const rides = await db.collection('rides').find({ driver_id: req.params.id }).toArray();

    const earnings = rides.reduce((sum, ride) => sum + (ride.fare || 0), 0);

    res.status(200).json({ total_earnings: earnings });

  } catch (err) {

    res.status(400).json({ error: "Failed to calculate earnings" });

  }

});

// Admin

// Admin Login

app.post('/admin/login', async (req, res) => {

  const { username, password } = req.body;

  try {

    const admin = await db.collection('admins').findOne({ username, password });

    if (!admin) return res.status(401).json({ error: "Invalid credentials" });

    res.status(200).json(admin);

  } catch (err) {

    res.status(500).json({ error: "Login failed" });

  }

});
```

```
// Block User (Customer or Driver)

app.patch('/admin/block/:id', async (req, res) => {

  try {

    const userResult = await db.collection('users').updateOne(
      {_id: new ObjectId(req.params.id)},
      {$set: { blocked: true }}
    );

    const driverResult = await db.collection('drivers').updateOne(
      {_id: new ObjectId(req.params.id)},
      {$set: { blocked: true }}
    );

    if (userResult.modifiedCount === 0 && driverResult.modifiedCount === 0) {
      return res.status(404).json({ error: "User or Driver not found" });
    }

    res.status(200).json({ message: "Blocked successfully" });

  } catch (err) {
    res.status(400).json({ error: "Invalid ID" });
  }
});
```

```
// Approve Driver Registration

app.patch('/admin/approve/:driverId', async (req, res) => {

  try {

    const result = await db.collection('drivers').updateOne(
      {_id: new ObjectId(req.params.driverId)},
      {$set: { approved: true }}
    );

  }
});
```

```

    if (result.modifiedCount === 0) return res.status(404).json({ error: "Driver not found" });
  });

  res.status(200).json({ updated: result.modifiedCount });

} catch (err) {
  res.status(400).json({ error: "Invalid driver ID" });

}

});

// View System Analytics

app.get('/admin/analytics', async (req, res) => {

  try {
    const usersCount = await db.collection('users').countDocuments();
    const driversCount = await db.collection('drivers').countDocuments();
    const ridesCount = await db.collection('rides').countDocuments();
    res.status(200).json({ users: usersCount, drivers: driversCount, rides: ridesCount });
  } catch (err) {
    res.status(500).json({ error: "Failed to fetch analytics" });
  }
}
);

```

#### THE TERMINAL RUN CODE WEEK 4 :

```

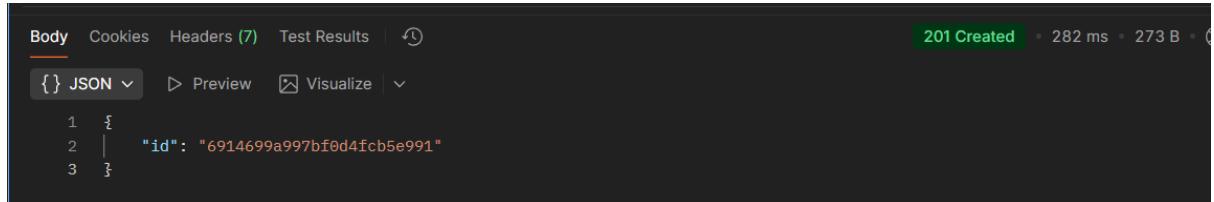
at node:internal/main/run_main_module:36:49 {
  code: 'MODULE_NOT_FOUND',
  requireStack: []
}

Node.js: v14.15.0
Focus folder in explorer (ctrl + click)
PS C:\Users\User\Desktop\mongodb-Lab> Node WEEK4.js
Server running on port 3000
Connected to MongoDB!

```

## 1. Result in the postman (Customer)

### I. Customer Register

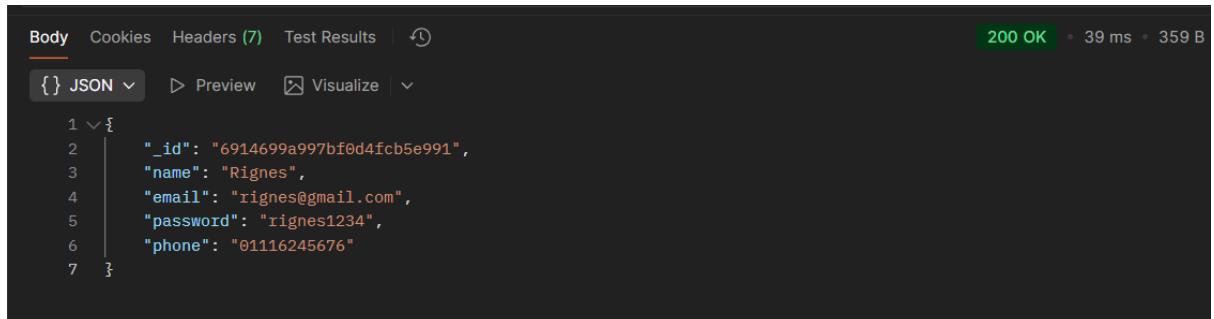


Body Cookies Headers (7) Test Results ⏱ 201 Created • 282 ms • 273 B ◉

{ } JSON ▾ Preview Visualize ▾

```
1 {  
2   "id": "6914699a997bf0d4fcb5e991"  
3 }
```

### II. Customer Login

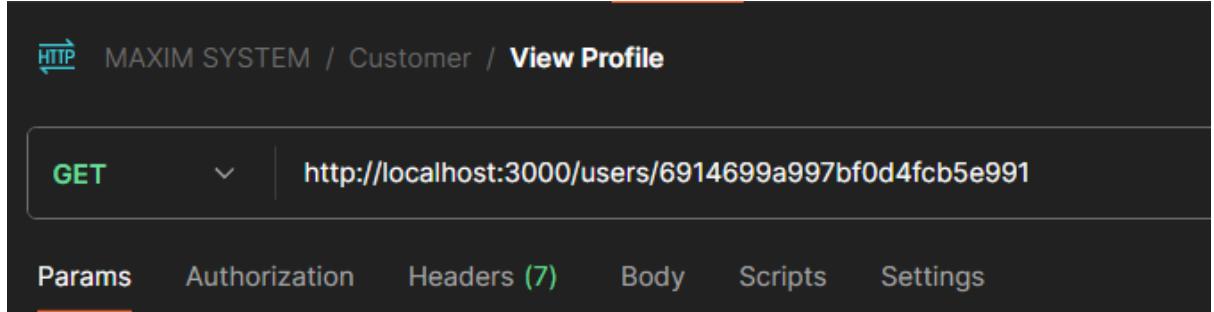


Body Cookies Headers (7) Test Results ⏱ 200 OK • 39 ms • 359 B

{ } JSON ▾ Preview Visualize ▾

```
1 {  
2   "_id": "6914699a997bf0d4fcb5e991",  
3   "name": "Rignes",  
4   "email": "rignes@gmail.com",  
5   "password": "rignes1234",  
6   "phone": "01116245676"  
7 }
```

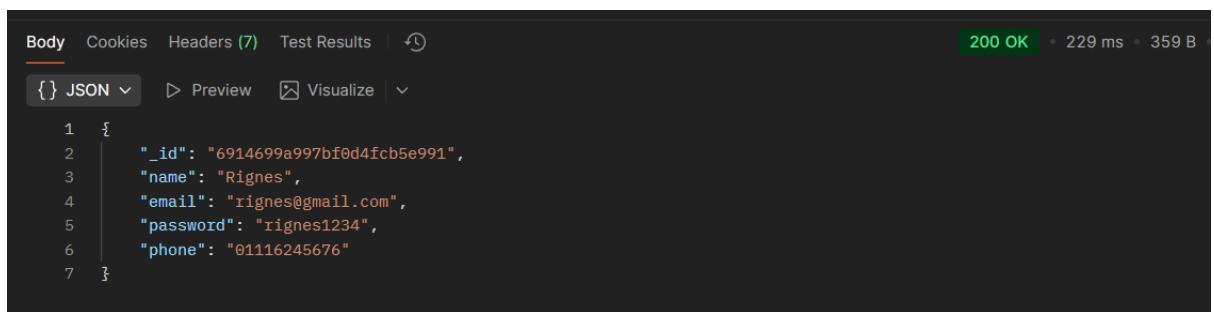
### III. View Profile



HTTP MAXIM SYSTEM / Customer / **View Profile**

GET http://localhost:3000/users/6914699a997bf0d4fcb5e991

Params Authorization Headers (7) Body Scripts Settings



Body Cookies Headers (7) Test Results ⏱ 200 OK • 229 ms • 359 B

{ } JSON ▾ Preview Visualize ▾

```
1 {  
2   "_id": "6914699a997bf0d4fcb5e991",  
3   "name": "Rignes",  
4   "email": "rignes@gmail.com",  
5   "password": "rignes1234",  
6   "phone": "01116245676"  
7 }
```

#### IV. Book Ride

Body Cookies Headers (7) Test Results ⏱

201 Created • 75 ms • 273 B

{ } JSON ▾ Preview Visualize ▾

```
1 {  
2   "id": "69146a4a997bf0d4fcb5e992"  
3 }
```

#### V. Track Ride

HTTP MAXIM SYSTEM / Customer / **Track Ride**

GET http://localhost:3000/rides/69146a4a997bf0d4fcb5e992

Params Authorization Headers (7) Body Scripts Settings

Body Cookies Headers (7) Test Results ⏱

200 OK • 256 ms • 379 B

{ } JSON ▾ Preview Visualize ▾

```
1 {  
2   "_id": "69146a4a997bf0d4fcb5e992",  
3   "pickup_location": "Melaka Sentral",  
4   "dropoff_location": "Kulim Sentral",  
5   "scheduled_time": "2025-11-13T17:00:00"  
6 }
```

#### VI. Rate Driver

POST http://localhost:3000/rides/69146a4a997bf0d4fcb5e992/rate

Params Authorization Headers (9) **Body** • Scripts Settings

none  form-data  x-www-form-urlencoded  raw  binary  GraphQL **JSON** ▾

```
1 {  
2   "rating": 4.9,  
3   "review": "Excellent drive with good ride experience"  
4 }  
5 
```

Body Cookies Headers (7) Test Results ⏱

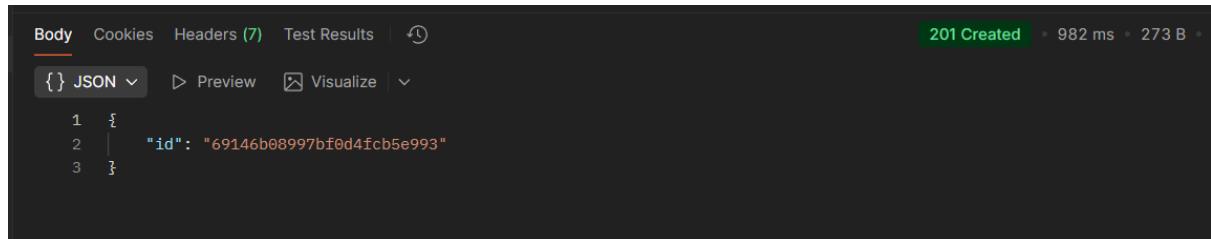
200 OK • 1.25 s • 247 B

{ } JSON ▾ Preview Visualize ▾

```
1 {  
2   "updated": 1  
3 }
```

## 2. Post in the postman (DRIVER)

### I. Driver Registration



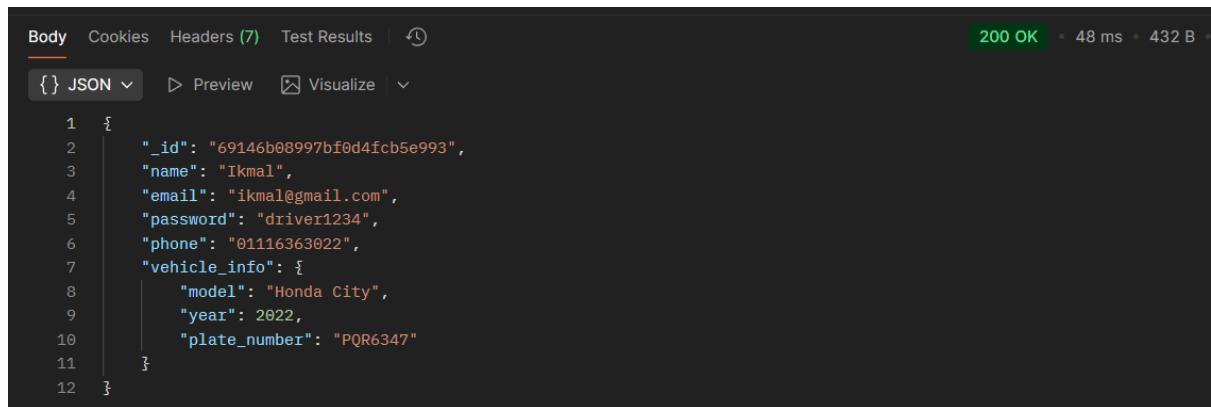
Body Cookies Headers (7) Test Results ⏱

201 Created 982 ms 273 B

{ } JSON ▾ Preview Visualize

```
1 {  
2   "id": "69146b08997bf0d4fcb5e993"  
3 }
```

### II. Driver Login



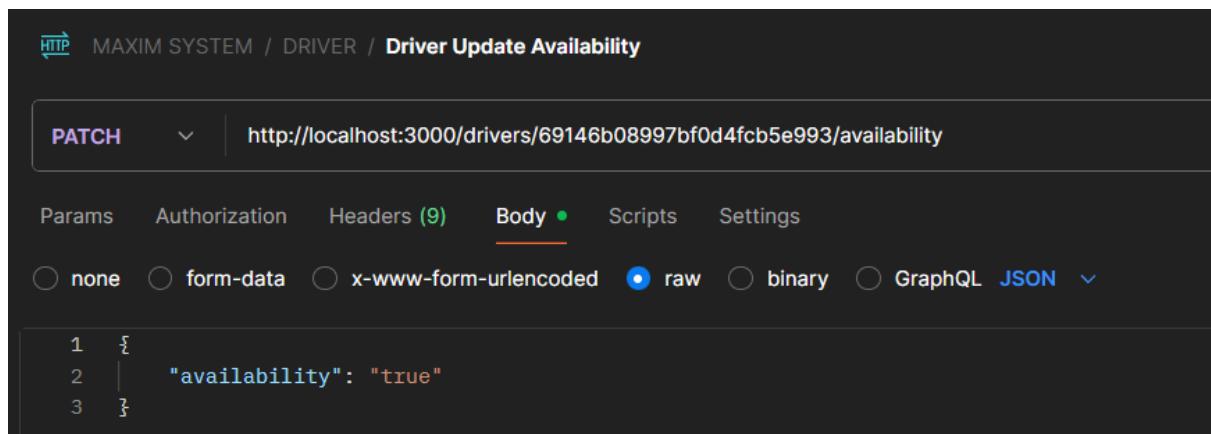
Body Cookies Headers (7) Test Results ⏱

200 OK 48 ms 432 B

{ } JSON ▾ Preview Visualize

```
1 {  
2   "_id": "69146b08997bf0d4fcb5e993",  
3   "name": "Ikmal",  
4   "email": "ikmal@gmail.com",  
5   "password": "driver1234",  
6   "phone": "01116363022",  
7   "vehicle_info": {  
8     "model": "Honda City",  
9     "year": 2022,  
10    "plate_number": "PQR6347"  
11  }  
12 }
```

### III. Driver update Availability



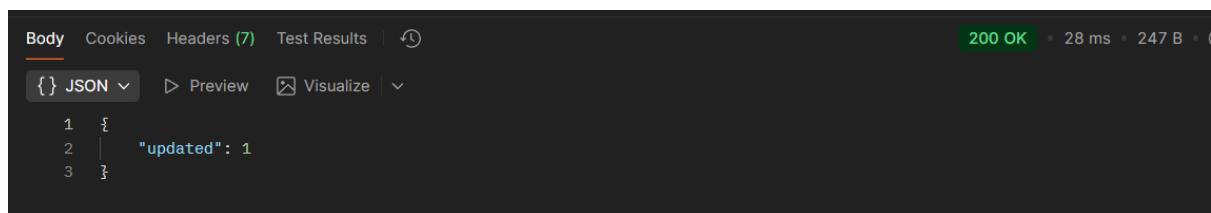
HTTP MAXIM SYSTEM / DRIVER / Driver Update Availability

PATCH http://localhost:3000/drivers/69146b08997bf0d4fcb5e993/availability

Params Authorization Headers (9) Body Scripts Settings

none form-data x-www-form-urlencoded raw binary GraphQL JSON

```
1 {  
2   "availability": "true"  
3 }
```



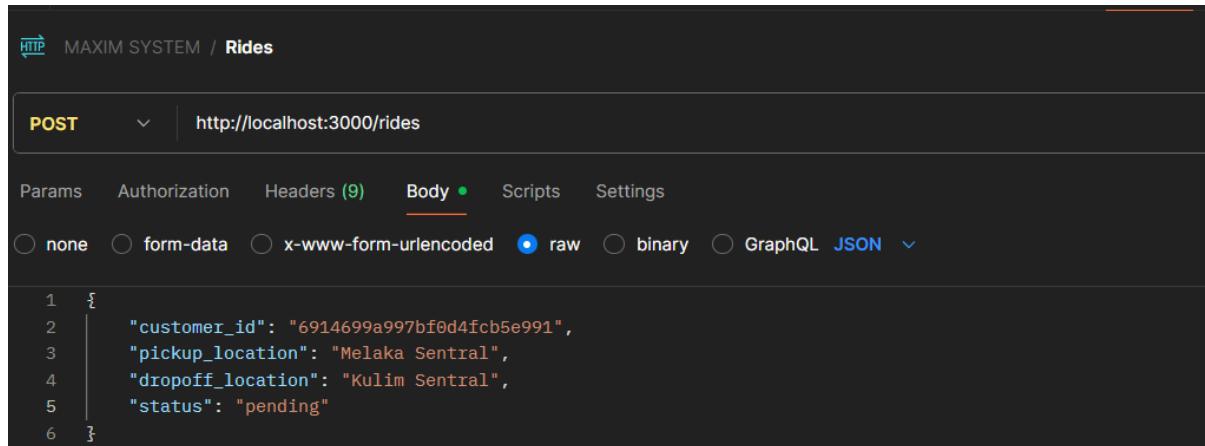
Body Cookies Headers (7) Test Results ⏱

200 OK 28 ms 247 B

{ } JSON ▾ Preview Visualize

```
1 {  
2   "updated": 1  
3 }
```

#### IV. Accept Ride Request



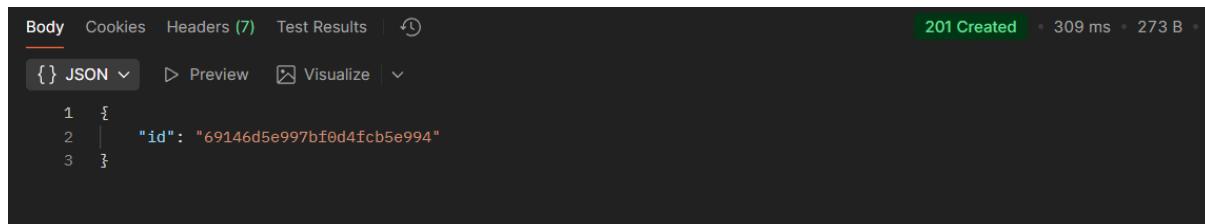
HTTP MAXIM SYSTEM / Rides

POST <http://localhost:3000/rides>

Params Authorization Headers (9) **Body** • Scripts Settings

none  form-data  x-www-form-urlencoded  raw  binary  GraphQL **JSON** ▾

```
1 {  
2   "customer_id": "6914699a997bf0d4fcb5e991",  
3   "pickup_location": "Melaka Sentral",  
4   "dropoff_location": "Kulim Sentral",  
5   "status": "pending"  
6 }
```

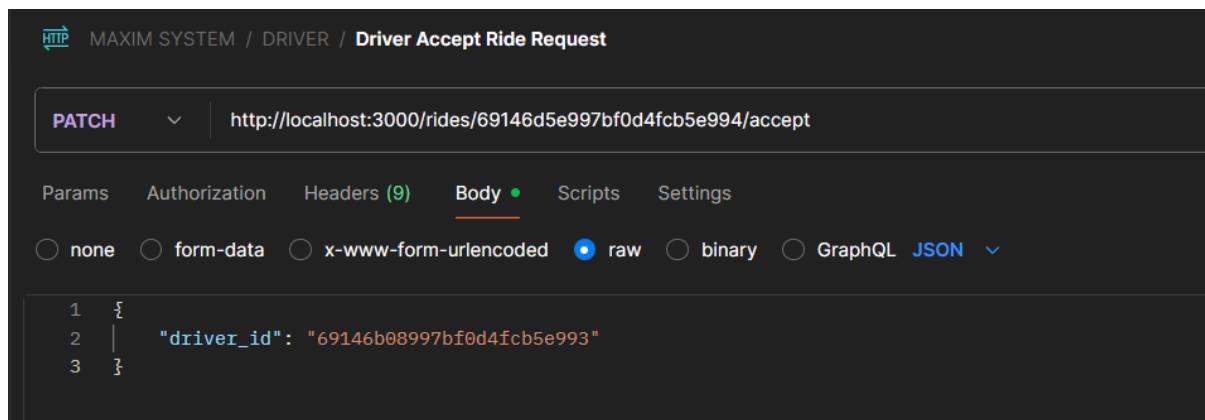


Body Cookies Headers (7) Test Results ⏱

201 Created • 309 ms • 273 B • ⏱

{ } JSON ▾ ▶ Preview 🖼 Visualize ▾

```
1 {  
2   "id": "69146d5e997bf0d4fcb5e994"  
3 }
```



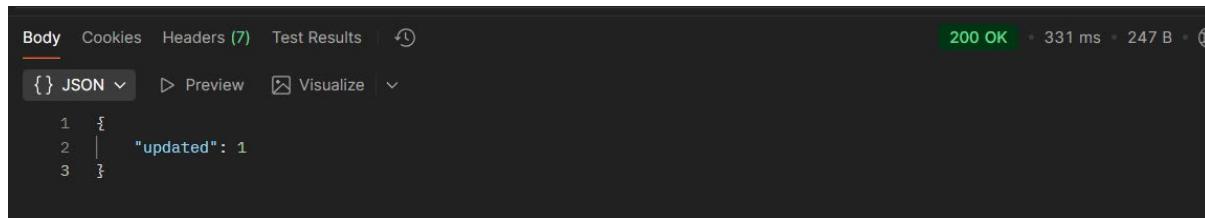
HTTP MAXIM SYSTEM / DRIVER / Driver Accept Ride Request

PATCH <http://localhost:3000/rides/69146d5e997bf0d4fcb5e994/accept>

Params Authorization Headers (9) **Body** • Scripts Settings

none  form-data  x-www-form-urlencoded  raw  binary  GraphQL **JSON** ▾

```
1 {  
2   "driver_id": "69146b08997bf0d4fcb5e993"  
3 }
```



Body Cookies Headers (7) Test Results ⏱

200 OK • 331 ms • 247 B • ⏱

{ } JSON ▾ ▶ Preview 🖼 Visualize ▾

```
1 {  
2   "updated": 1  
3 }
```

## V. View Earnings

MAXIM SYSTEM / DRIVER / Driver View Earnings

GET http://localhost:3000/drivers/69146b08997bf0d4fcb5e993/earnings

Params Authorization Headers (7) Body Scripts Settings

Body Cookies Headers (7) Test Results +  
200 OK • 68 ms • 255 B

{ } JSON ▾ ▷ Preview Visualize ▾

```
1 {  
2   "total_earnings": 0  
3 }
```

### 3. Result from postman (ADMIN)

#### I. Admin Login

Body Cookies Headers (7) Test Results +  
200 OK • 65 ms • 320 B

{ } JSON ▾ ▷ Preview Visualize ▾

```
1 {  
2   "_id": "6914736d3b334755421cb35a",  
3   "username": "yuan@gmail.com",  
4   "password": "admin1234"  
5 }
```

#### II. Block user

MAXIM SYSTEM / ADMIN / Admin Block User

PATCH http://localhost:3000/admin/block/6914699a997bf0d4fcb5e991

Params Authorization Headers (9) Body • Scripts Settings

none  form-data  x-www-form-urlencoded  raw  binary  GraphQL  JSON ▾

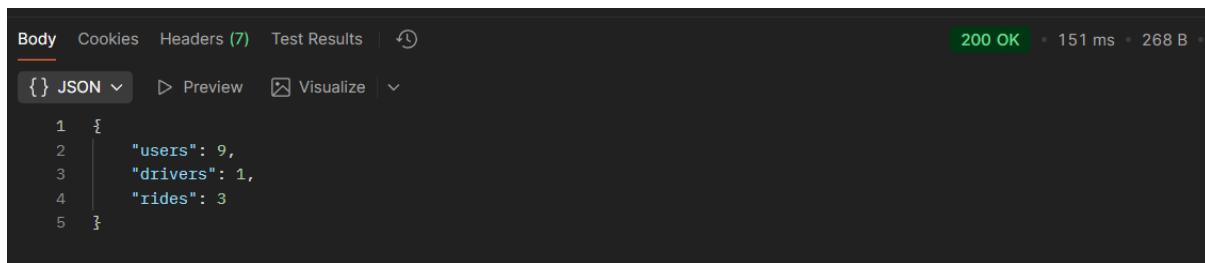
```
1 {  
2   "reason": "Violation of terms and conditions"  
3 }
```

Body Cookies Headers (7) Test Results +  
200 OK • 747 ms • 269 B

{ } JSON ▾ ▷ Preview Visualize ▾

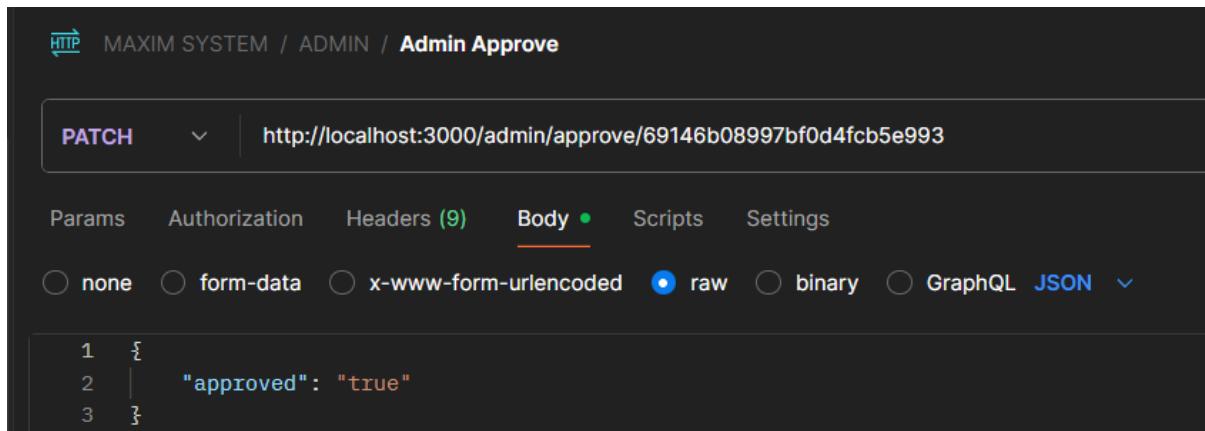
```
1 {  
2   "message": "Blocked successfully"  
3 }
```

### III. View System Analytic



```
200 OK • 151 ms • 268 B •  
{} JSON ▾ ▷ Preview ⚡ Visualize ▾  
1 {  
2   "users": 9,  
3   "drivers": 1,  
4   "rides": 3  
5 }
```

### IV. Approve Driver Registration



HTTP MAXIM SYSTEM / ADMIN / Admin Approve

PATCH ▾ | http://localhost:3000/admin/approve/69146b08997bf0d4fcb5e993

Params Authorization Headers (9) **Body** • Scripts Settings

none  form-data  x-www-form-urlencoded  raw  binary  GraphQL **JSON** ▾

```
1 {  
2   "approved": "true"  
3 }
```



```
200 OK • 17 ms • 247 B •  
{} JSON ▾ ▷ Preview ⚡ Visualize ▾  
1 {  
2   "updated": 1  
3 }
```