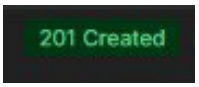
UNIVERSITI TEKNIKAL MALAYSIA MELAKA
FAKULTI TEKNOLOGI DAN KEJURUTERAAN ELEKTRONIK DAN KOMPUTER

EXERCISE 3

Code of Subject	**BERR 2243**	
Name of Subject	**DATABASE AND CLOUD SYSTEM**	
Title	**BUILDING A RIDE-HAILING REST API WITH EXPRESS.JS**	
Year	**2** **Semester: 2**	
Course	**BERR**	
Section	**2**	
Name of Student:		**Matrice No**
1.PUTRY BALQIS NABILAH BINTI MOHD JAMIL		B122310468
2. SITI NUR AQILAH BINTI SAZALI		B122310484
3. AZWA NABILA BINTI WANDI SURYA		B122310628
Lecturer / Instructor's Engineer / Tutor	**DR. SOO YEW GUAN**	
Marks		
Comments		**Stamp of submission**

Lab Question

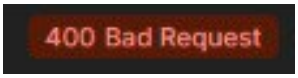
Answer by testing your API in Postman and observing responses.

1. POST Request:
   - What HTTP status code is returned when a ride is created successfully?
     Answer:
     

   - What is the structure of the response body?
     Answer:
     

2. GET Request:
   - What happens if the rides collection is empty?
     Answer:
     empty array
   - What data type is returned in the response (array/object)?
     Answer:
     Array

3. Fix PATCH and DELETE Error:
   - Catch the error when requesting PATCH or DELETE API, then try to fix the issue reported.
     Answer:
     Error:

PATCH API	DELETE API
	

Fixed code:

```
const { ObjectId } = require('mongodb');

// In your PATCH route:
app.patch('/rides/:id', async (req, res) => {
    if (!ObjectId.isValid(req.params.id)) {
        return res.status(400).json({ error: "Invalid
ride ID format" });
    }
    try {
        const result = await
db.collection('rides').updateOne(
            { _id: new ObjectId(req.params.id) },
            { $set: { status: req.body.status } }
        );
        // ... rest of your code
    } catch (err) {
        res.status(400).json({ error: "Invalid ride ID
or data" });
```

```
        }
    });

    // In your DELETE route:
    app.delete('/rides/:id', async (req, res) => {
        if (!ObjectId.isValid(req.params.id)) {
            return res.status(400).json({ error: "Invalid
    ride ID format" });
        }
        try {
            const result = await
    db.collection('rides').deleteOne(
                { _id: new ObjectId(req.params.id) }
            );
            // ... rest of your code
        } catch (err) {
            res.status(400).json({ error: "Invalid ride ID"
    });
        }
});
```

- If you try to update a non-existent ride ID, what status code is returned?
  Answer:

- What is the value of updated in the response if the update succeeds?
  Answer:
  1
- How does the API differentiate between a successful deletion and a failed one?
  Answer:

Deletion	Failed

JSON response body:
{
"deleted": 1
}
	JSON response body:
	{
	"error": "Ride not found"
	}

4. Users Endpoints:

- Based on the exercise above, create the endpoints to handle the CRUD operations for user's account
  Answer:
  i. Created new route paths starting with /users/
  ii. Changed the database collection name from 'rides' to 'users' in all the new endpoint handlers.
  iii. Adjusted the request and response structures to handle user-specific data (like username, email, password) instead of ride details.

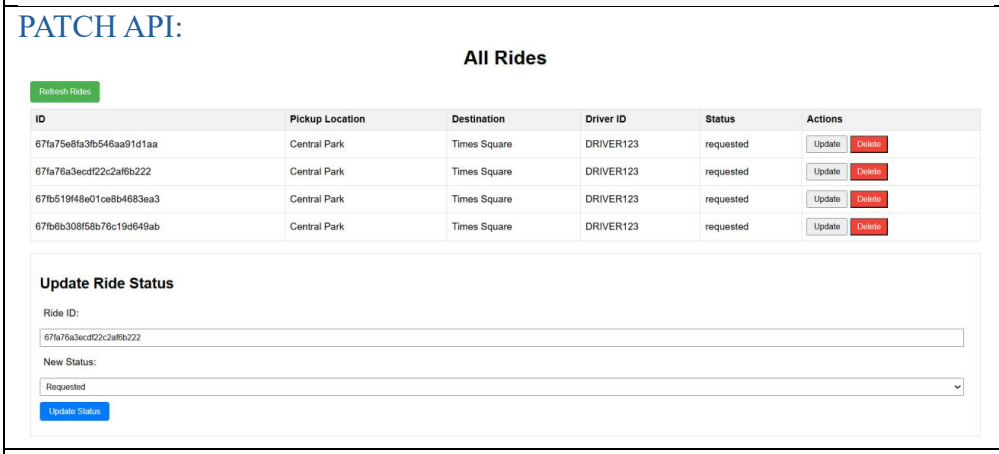
5. FrontEnd:

- Upload the Postman JSON to any AI tools, and generate a simple HTML and JS Dashboard for you
  Answer:

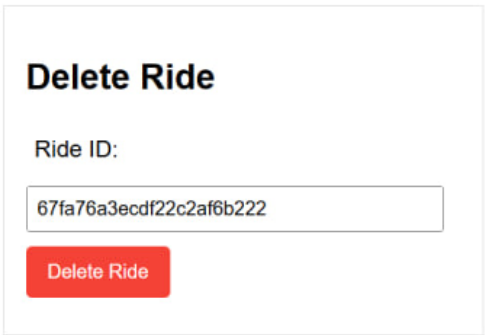
POST API:

GET API:

PATCH API:



DELETE API:



Submission Requirements

1. GitHub Repository with:
   - Complete code.
     https://github.com/PutryNabilah/berr2243-2

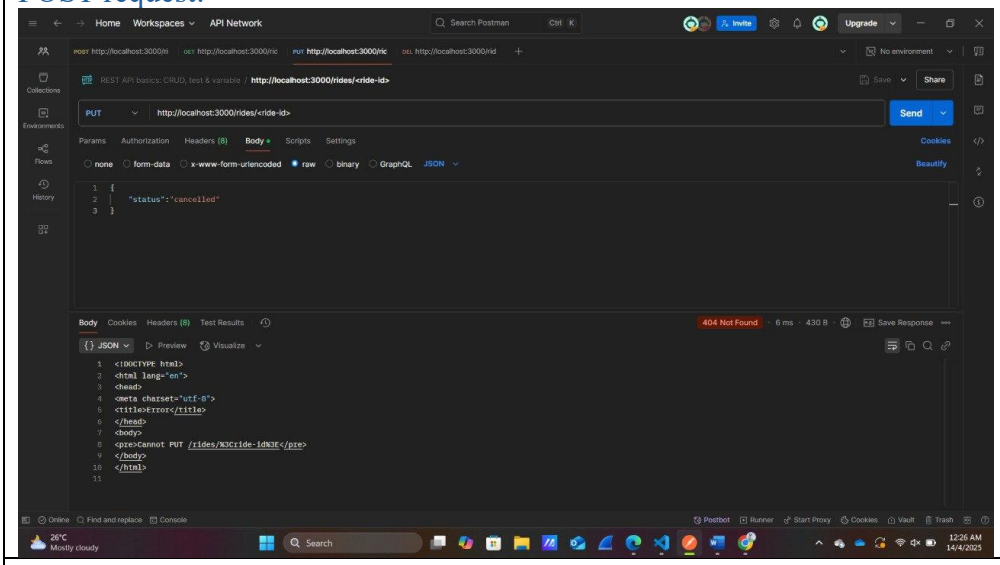
2. Postman Collection:

- Export and include the collection file.

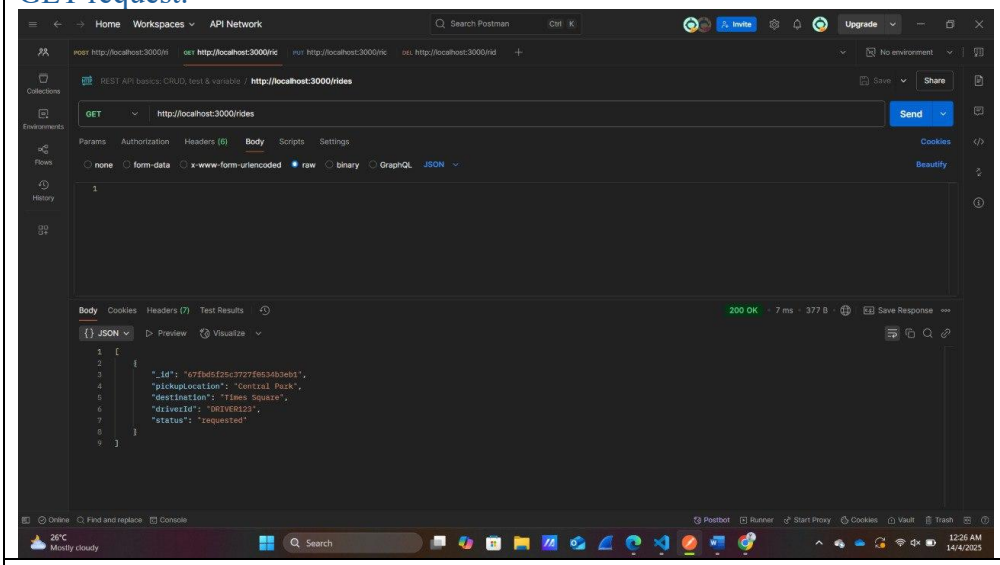
REST API basics-
CRUD, test & variable

3. Exercise Report:
   - Screenshots of Postman requests/responses.

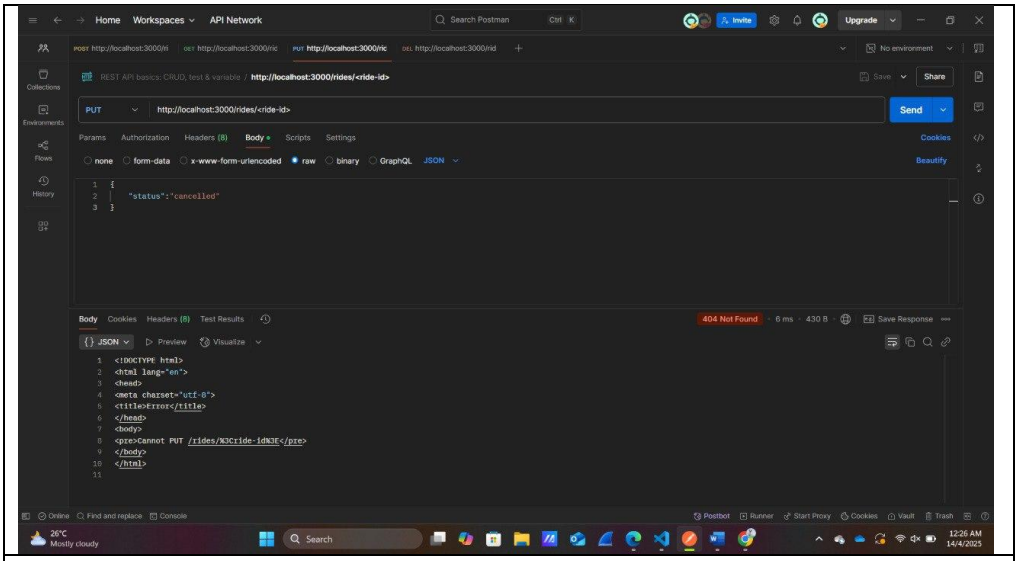
POST request:



GET request:



PATCH request:



DELETE request: