Driver-Notification-System

A notification service that sends a message according to the completion rate of the last 100 rides up to yesterday. Requires the driver ID as input.

This service is for ride sharing apps such as Uber, PATHAO, etc. The purpose of this project is to increase driver ride completion rate. Thus, improving rider service quality.

The core purpose of this project is to help the companies provide standard service by choosing skilled and responsible human resources who represent their company on the field and can ensure a quality service. Using this project, companies can send messages to all the riders based on their completion rate. In this way, companies can monitor their rider's active participation in the field and can provide them feedback through messages so that they can improve their service. Moreover, through this service, Admins can also monitor the performance of the riders from the very beginning of their joining day.

Getting Started

To clone the repository, go to the directory you want to clone the repository into from command line and enter:

```
git clone https://github.com/Warrior-47/Driver-Notification-System.git
```

Then, go to the project directory (where package.json exists) and execute:

```
npm install --save
```

Now, to start the server:

```
npm start
```

Note: A MySql server must be running for the project to work

Details

Routes

- 127.0.0.1/register: Method = POST
- 127.0.0.1/place_order: Method = POST
- 127.0.0.1/admin: Method = POST
- 127.0.0.1/driver/:driver_id?token=[insert auth token here]: Method = GET
- 127.0.0.1/notify?driver_id=[insert id here]: Method = GET

127.0.0.1/register

Takes a JSON object containing rider's information like,

{

"name": "[Rider's name]",

"nid_number": "[Rider's NID number]",

"phone": "[Rider's phone number]",

"vehicle_id": "[Rider's vehicle ID]"

}

and inserts the rider's information into the database. Returns a success message as a JSON object if successfully registered.

When all the inputs are valid :

{

"success": true,

"message": “Successfully Registered"

}

When NID number or vehicle ID is already in the database, it returns

{

"success": false,

"Message": "NID number or Vehicle ID already registered"

}

When NID number is not valid , it returns

{

"success": false,

"Message": "Data too long for column 'nid_number' at row 1"

}

When vehicle ID is invalid, it returns

{

"success": false,

"message": "Data too long for column 'vehicle_id' at row 1"

}

127.0.0.1/place_order

Takes a JSON object containing order information like,

{

"order_id": "[Order ID]",

"driver_id": "[Rider's ID who accepted the order]",

"status": "[Status of the ride, '1' for COMPLETED and '0' for CANCELLED]"

}

and inserts the order information into the database. Returns a success message as a JSON object if successfully inserted.

When inputs are valid, it returns


```
{
    "success": true,
    "message": "Inserted Order Successfully"
}
```


When order ID is in the database, it returns


```
{
    "success": false,
    "message": “Duplicate entry '01Ffj0TJKI' for key 'PRIMARY'"
}
```


When order ID is invalid, it returns


```
{
    "success": false,
    "message": "Data too long for column 'order_id' at row 1"
}
```

When rider ID is not in the database, it returns

{

"success": false,

"message": "Cannot add or update a child row: a foreign key constraint fails (`notification_system`.`order_completion`, CONSTRAINT `order_completion_ibfk_1` FOREIGN KEY (`driver_id`) REFERENCES `drivers` (`driver_id`) ON DELETE CASCADE ON UPDATE CASCADE)"

}

127.0.0.1/admin

Takes a JSON object containing admin login information like,

{

"username": "[admin username]",

"password": "[admin password]"

}

and logs in the user as admin. Returns a JSON object that contains an authorization token if successfully registered.

When all inputs are valid, it returns

{

"success": true,

"token": "dummy"

}

When any of the inputs is invalid, it returns

{

"success": false

}

Default Username: admin

Default Password: asd

127.0.0.1/driver/:driver_id?token=[insert auth token here]

Takes a rider's ID as parameter and an authorization token as query and fetches the corresponding rider's information. Information returned as a JSON object like,

{

"driver_id": "[ID]",

"name": "[Name]",

"nid_number": "[NID Number]",

"phone": "[Phone Number]",

"vehicle_id": "[Vehicle ID]",

"rides_cancelled": [Rides Cancelled],

"rides_completed": [Rides Completed],

"total_rides": [Total Rides]

}

Requires admin token to fetch data. Without it, an error message is shown.

Also, returns an error message if rider ID is invalid and not in the database.

{

"Error": "invalid driver id"

}

If token is not given in the URL, it returns,

{

"message": "Authorization failed"

}

Default Authorization token: dummy

127.0.0.1/notify?driver_id=[insert id here]

Takes a rider's ID as a query, then calculates completion rate and returns a message corresponding to the completion rate. Example:

If rider's completion rate is greater than or equal 0.7, then returns

{

"Completion_rate": "completion rate",

"Message": "Please complete more to get more requests."

}

If rider's completion rate is less than or equal 0.5, then returns

{

"Completion_rate": "completion rate",

"Message": "Your completion rate is very low. You will get suspended if you do not increase your completion rate."

}

If rider's completion rate is greater than or equal 0.71 or he got less than 100 orders up to yesterday ,then returns

{

"Completion_rate": "completion rate",

"Message": "Please complete more to get more requests."

}

Also returns an error message if rider ID is invalid and not in the database.

When rider's ID is not present in the database , it returns

{

"Error": "invalid driver id"

}

Current Status

Features:

- Admin password is stored in the database as a hash and the given password is verified against the hashed password to authenticate the admin
- New riders can register through verification using NID number and vehicle ID
- Database update when new orders are placed
- Calculates completion rate and sends a notification message

Summary:

From the given problem, the achievement of our project is that it can verify the data before storing and provide an error message for invalid inputs or missing rider ID. It can provide a notification message to our riders corresponding to their completion rate. Only admin can access a rider's information.

Risk:

- Missing Authorization system

Progress:

All the tasks given in the problem are done. But some of the new features will definitely provide a better experience to our riders. Our future plan is to implement those features.

Bugs:

So far there is no bug we have found. In the future, some problems may arise because of the release of the latest version but hopefully that can be resolved with a little bit of modification.

Future Plan:

- Create a front-end using React js to complete the client-server architecture to make it more attractive and user friendly
- The front end will notify the riders at a fixed interval using the notification service
- An Authorization system will be added to make sure that only admins can access a rider's information