CPSC 417 Project Final Report

Transit Database Management System For Scheduling and Data Analytics

Erick Power 10147457

Sifan Xu 10146334

INTRODUCTION

Efficient public transportation is a challenge in large municipalities. There is a great deal of logistical work involved in management and allocation of fleet resources, such as vehicles and operators, to the many transit routes that traverse a city. Further, without a data collection and management system in place, it is impossible to accurately track valuable data about transit usage, such as ridership, which could be valuable to improving services.

The goal of this project is to create a web application for a database system for a hypothetical municipal transit system that manages data necessary to resolve challenges relating to the Following:

1. Scheduling of employees (drivers and train operators) and buses to established transit routes, and
2. Operational decisions regarding frequency of service and/or establishment of new routes to better serve the public.

The system allows administrators to more easily manage scheduling routes, adding or removing a stop or employee in the database, allow customers to easily access route information for planning their travels and register for transit fare smart cards if they so desire, and data analysts to access statistics about ridership to make service requests to administrators.

In summary, the goal of our project is to produce a web application that provides a concise system for transit scheduling, user interactions from clients, operators, analysts and administrators. Decision-makers will be able to track usage and make decisions regarding deployment of additional services, or modification of existing ones. Passengers as well as transit service-providers will be able to benefit from the implementation of this application.

SECTION I: PROJECT DESIGN

USER TYPES

Unauthenticated users can perform the following actions:

- view all the existing routes, their scheduled runs, route legs
- view all the existing stops and when the next runs for each route (which includes a route leg containing that stop) are scheduled to arrive
- register an account as a **client**
- login to an existing account

Furthermore there are 4 classes of mutually exclusive **authenticated users** for our system, who inherit the same transactions as unauthenticated users and have additional functionality:

Client

- change name and/or email
- make a deposit onto their account, corresponding to their id number on the system

Admin

- create/edit/delete **routes** with route id, name, start stop, end stop
- create/delete **runs** for each route with service start time, operator, and vehicle
- create/delete **route legs** for each route with start stop, end stop, and duration
- create/delete **stops**
- view/create/delete **vehicles** in the database with license, capacity
- view service requests made by **analysts**
- add existing **client** users as an **employee**: either **admin, operator** or **analyst**
- remove existing **employees** and demote them to **clients**

Operator

- view **runs** for which they have been assigned

Analyst

- view **max ridership** data for each route
- create **service requests** for each route

SECTION II: ENTITY RELATION DIAGRAM AND RELATIONAL MODEL

There are several changes we made to the database model which are reflected in both diagrams:

- Renaming REGISTERED_USER to USER

- Removing the phone number and address attributes from USER. This is because an Admin is not responsible for any direct communication to client, so personal information like such should not be kept in the database.

- Removing the EMERGENCY_CONTACT entity. This database system is meant to provide employees with information on transit schedules and analytics, and not meant as a general database for employee information. It was also removed as a security measure, as the database will be online, and could possibly be hacked, and personal information leaked. Thus, this entity is not needed.

- Moving the service_start_time attribute from ROUTE to RUN. A ROUTE is composed of runs, thus its service_start_time is actually the service_start_time of the earliest RUN assigned to the ROUTE.

- combining BUS_OPERATOR and TRAIN_OPERATOR into a single entity, OPERATOR. The rationale behind this decision was to provide a level of abstraction to the database and make it quicker to implement the database using OPERATOR to denote all employees who all qualified to operate any VEHICLE.

- combining TRAIN, CARRIAGE and BUS into a single entity, VEHICLE. The reasoning is similar to the rationale described above for OPERATOR. An ADMINISTRATOR only needs to know which VEHICLE and OPERATOR is assigned to. Given the simplification that an OPERATOR can operate any VEHICLE, it is simpler to implement.

- Removing the TRIP and LEGS_OF_TRIP entities. Based on client experience, the majority of people who commute do so on a daily basis and have their schedules memorized. Thus, this information does not have to be stored for the client to view.

- Removing LEG_INSTANCE entity. Since the legs of each run are the exact same throughout a route, keeping this entity would have created a lot of unnecessary redundancy in the database.

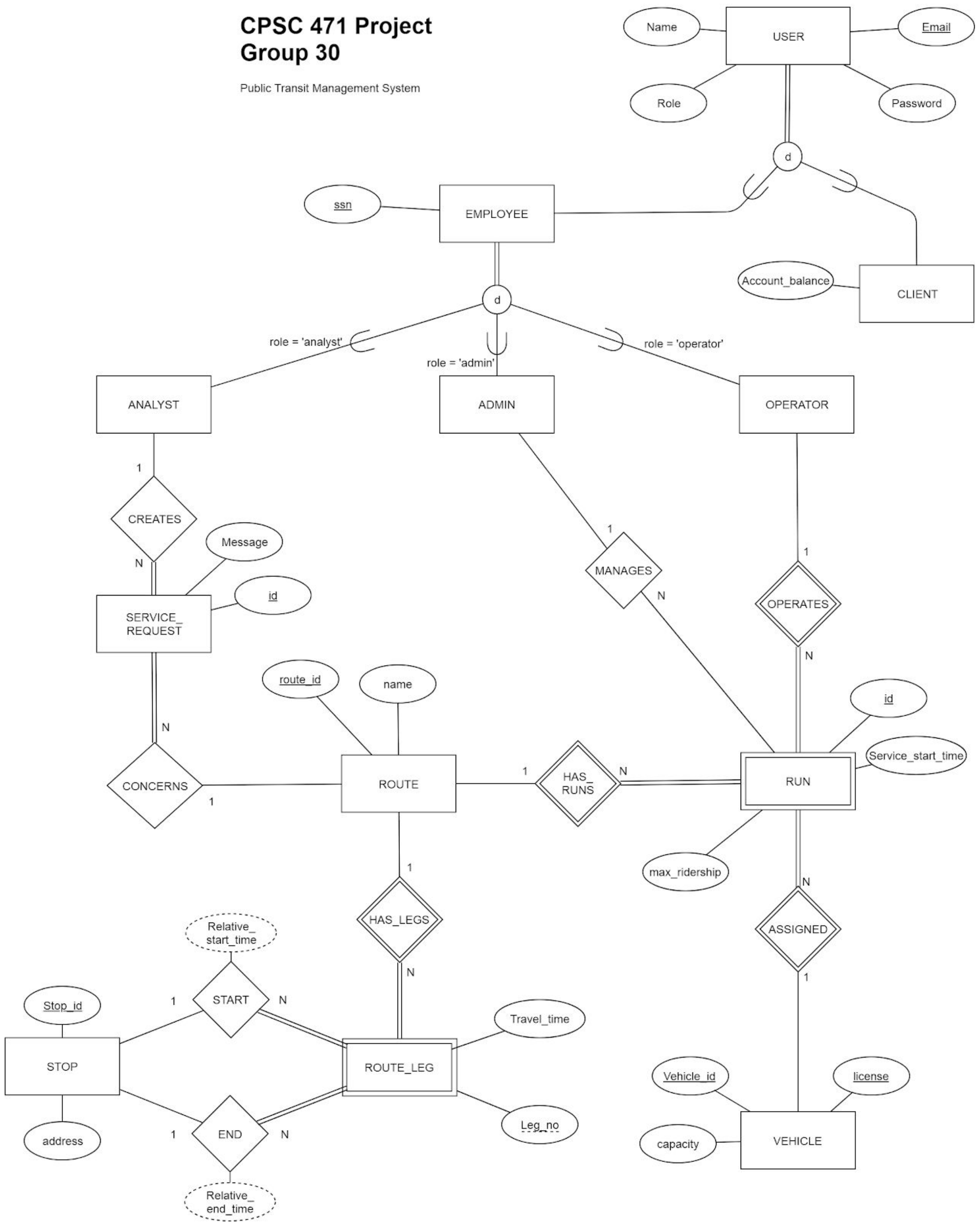


Figure 1: Entity-Relationship Diagram

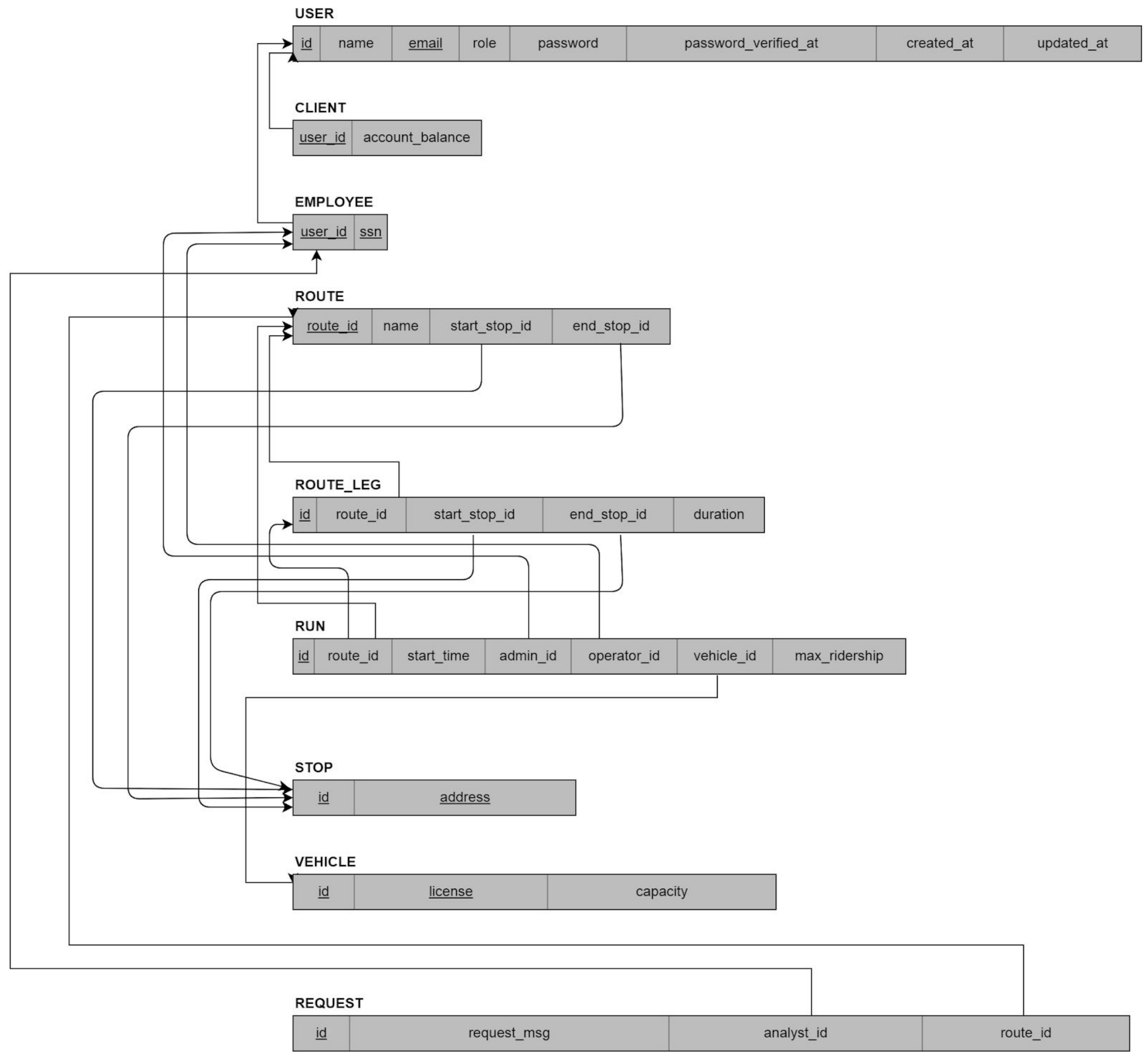


Figure 2: Relationship Model

SECTION III: IMPLEMENTATION

To implement this project, we used MySQL and phpMyAdmin in conjunction with Laravel, a PHP framework. The SQL statements we used to implement each transaction are listed below:

```
-- create a new database
DROP DATABASE IF EXISTS Transit;
CREATE DATABASE Transit;

-- seed the database
CREATE TABLE Stops (
	id bigint(20) UNSIGNED NOT NULL AUTO_INCREMENT,
	address varchar(255) NOT NULL UNIQUE,
	PRIMARY KEY (id)
);

CREATE TABLE Vehicles (
	id bigint(20) UNSIGNED NOT NULL AUTO_INCREMENT,
	license varchar(255) NOT NULL,
	capacity int(11) NOT NULL,
	PRIMARY KEY (id)
);

CREATE TABLE Clients (
	user_id bigint(20) UNSIGNED NOT NULL,
	account_balance int(11) DEFAULT 0,
	FOREIGN KEY(user_id) REFERENCES Users(id) ON UPDATE CASCADE ON
DELETE CASCADE
);

CREATE TABLE Employees (
	user_id bigint(20) UNSIGNED NOT NULL,
	ssn int(11) NOT NULL,
	FOREIGN KEY (user_id) REFERENCES Users(id) ON UPDATE CASCADE ON
DELETE CASCADE
);

CREATE TABLE Routes (
	id bigint(20) UNSIGNED NOT NULL AUTO_INCREMENT,
	name varchar(255) NOT NULL,
	start_stop_id bigint(20) UNSIGNED NOT NULL,
	end_stop_id bigint(20) UNSIGNED NOT NULL,
	PRIMARY KEY (id),
	FOREIGN KEY (start_stop_id) REFERENCES Stops(id) ON UPDATE CASCADE ON
DELETE CASCADE,
	FOREIGN KEY (end_stop_id) REFERENCES Stops(id) ON UPDATE CASCADE ON
DELETE CASCADE
);

CREATE TABLE Route_legs (
	id bigint(20) UNSIGNED NOT NULL AUTO_INCREMENT,
	route_id bigint(20) UNSIGNED NOT NULL,
	start_stop_id bigint(20) UNSIGNED NOT NULL,
	end_stop_id bigint(20) UNSIGNED NOT NULL,
	duration int(11) NOT NULL,
	PRIMARY KEY (id),
	FOREIGN KEY (route_id) REFERENCES Routes(id) ON UPDATE CASCADE ON
DELETE CASCADE,
	FOREIGN KEY (start_stop_id) REFERENCES Stops(id) ON UPDATE CASCADE ON
DELETE CASCADE,
	FOREIGN KEY (end_stop_id) REFERENCES Stops(id) ON UPDATE CASCADE ON
DELETE CASCADE
);

CREATE TABLE Runs (
	id bigint(20) UNSIGNED NOT NULL AUTO_INCREMENT,
	route_id bigint(20) UNSIGNED NOT NULL,
	start_time time NOT NULL,
	admin_id bigint(20) UNSIGNED NOT NULL,
	operator_id bigint(20) UNSIGNED NOT NULL,
	vehicle_id bigint(20) UNSIGNED NOT NULL,
	max_ridership bigint(20) UNSIGNED NOT NULL DEFAULT 0,
	PRIMARY KEY (id),
	FOREIGN KEY (route_id) REFERENCES Routes(id) ON UPDATE CASCADE ON
DELETE CASCADE,
	FOREIGN KEY (admin_id) REFERENCES Employees(user_id) ON UPDATE
CASCADE ON DELETE CASCADE,
	FOREIGN KEY (operator_id) REFERENCES Employees(user_id) ON UPDATE
CASCADE ON DELETE CASCADE,
	FOREIGN KEY (vehicle_id) REFERENCES Vehicles(id) ON UPDATE CASCADE ON
DELETE CASCADE
);

CREATE TABLE Requests (
	id bigint(20) UNSIGNED NOT NULL AUTO_INCREMENT,
	req_message text NOT NULL,
	analyst_id bigint(20) UNSIGNED NOT NULL,
	route_id bigint(20) UNSIGNED NOT NULL,
	PRIMARY KEY (id),
	FOREIGN KEY (analyst_id) REFERENCES Employees(user_id) ON UPDATE
CASCADE ON DELETE CASCADE,
	FOREIGN KEY (route_id) REFERENCES Routes(id) ON UPDATE CASCADE ON
DELETE CASCADE
);

ALTER TABLE stops AUTO_INCREMENT = 1000;

INSERT INTO Clients (user_id, account_balance) VALUES
(1, 0);

INSERT INTO Employees (user_id, ssn) VALUES
(2, 222222222),
(3, 333333333),
(4, 444444444),
(5, 555555555),
(6, 666666666);

INSERT INTO Vehicles (id, license, capacity) VALUES
(1, 'DD8-392A', 40),
(2, 'DD8-392B', 40),
(3, 'DD8-392C', 40),
(4, 'DD8-392D', 60),
(5, 'DD8-392E', 60),
(6, 'DD8-392F', 60);

INSERT INTO Stops (id, address) VALUES
(1000, '0th Ave 1000'),
(1010, '1st Ave 1010'),
(1020, '2nd Ave 1020'),
(1030, '3rd Ave 1030'),
(1040, '4th Ave 1040'),
(1050, '5th Ave 1050'),
(1060, '6th Ave 1060');

INSERT INTO Routes (id, name, start_stop_id, end_stop_id) VALUES
(1, 'Elbow Drive', 1000, 1020),
(2, 'Fish Creek', 1020, 1040),
(3, 'Bridlewood', 1040, 1060);

INSERT INTO Runs (route_id, start_time, admin_id, operator_id, vehicle_id, max_ridership)
VALUES
(1, '6:00', 2, 3, 1, 17),
(1, '7:00', 2, 4, 2, 30),
(1, '8:00', 2, 3, 3, 40),
(2, '12:00', 2, 4, 4, 22),
(3, '12:00', 2, 5, 5, 25);

INSERT INTO Route_legs (route_id, start_stop_id, end_stop_id, duration) VALUES
(1, 1000, 1010, 5),
(1, 1010, 1020, 5),
(2, 1020, 1030, 5),
(2, 1030, 1040, 5),
(3, 1040, 1050, 5),
(3, 1050, 1060, 5);

INSERT INTO Requests (id, req_message, analyst_id, route_id) VALUES
(1, 'TEST REQUEST: RUN 3 reaches maximum capacity. Needs another run shortly before or
shortly after RUN 3.', 6, 1);
```

-- *The following SQL queries were created and executed using PHP in the Laravel framework*

/*EMPLOYEES*/

-- select all employees

```
        $sql = 'SELECT * FROM users u, employees e WHERE u.id = e.user_id';
        $employees = DB::select($sql);
```

-- select all employees who are operators

```
        $sql = 'SELECT * FROM employees e, users u WHERE e.user_id = u.id AND u.role =
                "operator"';
        $operators = DB::select($sql);
```

-- create a new employee

```
        $sql = 'DELETE FROM clients WHERE user_id = ' . $id;
        DB::delete($sql);

        $sql = 'INSERT INTO employees (user_id, ssn) VALUES (?, ?)';
        $fields = [$id, $ssn];
        DB::insert($sql, $fields);

        $sql = 'UPDATE users SET role = ? WHERE id = ?';
        $fields = [$role, $id];
        DB::update($sql, $fields);
```

-- delete employee

```
        $sql = 'DELETE FROM employees WHERE user_id = ' . $id;
        DB::delete($sql);

        $sql = 'INSERT INTO clients (user_id, account_balance) VALUES (?, ?)';
        $fields = [$id, 0];
        DB::insert($sql, $fields);
```

/*CLIENTS*/

-- select all clients

```
        $sql = 'SELECT * FROM users u, clients c WHERE u.id = c.user_id';
        $clients = DB::select($sql);
```

-- create a new client

```
        $sql = 'INSERT INTO clients (user_id, account_balance) VALUES (?, ?)';
        $fields = [$user->id, 0];
        DB::insert($sql, $fields);
```

/*VEHICLES*/

-- select all vehicles
```
        $sql = 'SELECT * FROM vehicles';
        $vehicles = DB::select($sql);
```

-- create new vehicle
```
        $sql = 'INSERT INTO vehicles (license, capacity) VALUES (?, ?)';
        $fields = [$license, $capacity];
        DB::insert($sql, $fields);
```

-- delete vehicle
```
        $sql = 'DELETE FROM vehicles WHERE id = ' . $id;
        DB::delete($sql);
```

/*ROUTES*/

-- select all routes
```
        $sql = 'SELECT * FROM routes';
        $routes = DB::select($sql);
```

-- select all routes where id = $id
```
        $sql = 'SELECT * FROM routes WHERE id = ' . $id;
        $route = collect(DB::select($sql))->first();
```

-- create route
```
        $sql = 'INSERT INTO routes (id, name, start_stop_id, end_stop_id) values (?, ?, ?, ?)';
        $fields = [request('id'), request('name'), request('start_stop_id'), request('end_stop_id')];
        DB::insert($sql, $fields);
```

-- update route
```
        $sql = 'UPDATE routes SET id = ?, name = ?, start_stop_id = ?, end_stop_id = ?
                WHERE id = ?';
        $fields = [request('id'), request('name'), request('start_stop_id'), request('end_stop_id'),
                $id];
        DB::update($sql, $fields);
```

-- delete a route
```
        $sql = 'DELETE FROM routes WHERE id = ' . $id;
        DB::delete($sql);
```

/*RUNS*/

```
-- select all runs where route_id = $id
        $sql = 'SELECT * FROM runs WHERE route_id = ' . $id;
        $runs = DB::select($sql);

-- create new run
        $sql = 'INSERT INTO runs (route_id, start_time, admin_id, operator_id, vehicle_id,
                max_ridership) values (?, ?, ?, ?, ?, ?)';
        $fields = [$route_id, request('start_time'), Auth::user()->id, request('operator_id'),
                request('vehicle_id'), request('max_ridership')];
        DB::insert($sql, $fields);

-- delete run
        $sql = 'DELETE FROM runs WHERE id = ' . $id;
        DB::delete($sql);
```

/*ROUTE_LEGS*/

```
-- select all route_legs where route_id = $id
        $sql = 'SELECT * FROM route_legs WHERE route_id = ' .$id;
        $route_legs = DB::select($sql);

-- select all route_legs where route_id = $id
        $sql = 'SELECT * FROM route_legs WHERE route_id = ' . $id . ' ORDER BY
                start_stop_id';
        $route_legs = DB::select($sql);

        $sql = 'SELECT * FROM stops';
        $stops = DB::select($sql);

-- create a new route leg
        $sql = 'INSERT INTO route_legs (route_id, start_stop_id, end_stop_id, duration)
                VALUES (?, ?, ?, ?)';
        $fields = [$id, request('start_stop_id'), request('end_stop_id'), request('duration')];
        DB::insert($sql, $fields);

-- delete a route leg
        $sql = 'DELETE FROM route_legs WHERE id = ' . $route_leg_id;
        DB::delete($sql);

-- update a route
        $sql = 'UPDATE routes SET id = ?, name = ?, start_stop_id = ?, end_stop_id = ?
                WHERE id = ?';
        $fields = [request('id'), request('name'), request('start_stop_id'), request('end_stop_id'),
                $id];
        DB::insert($sql, $fields);
```

/*STOPS*/

```
-- select all stops & order by id
        $sql = 'SELECT * FROM stops ORDER BY id';
        $stops = DB::select($sql);

-- create a new stop
        $sql = 'INSERT INTO stops (address) VALUES (?)';
        $fields = [request('address')];
        DB::insert($sql, $fields);

-- delete a stop
        $sql = 'DELETE FROM stops WHERE id = ' . $id;
        DB::delete($sql);
```

/*SERVICE REQUESTS*/

```
-- select all service requests
        $sql = 'SELECT * FROM requests ORDER BY route_id';
        $requests = DB::select($sql);

-- create service request
        $sql = 'INSERT INTO Requests (req_message, analyst_id, route_id) VALUES (?, ?, ?)';
        $fields = [request('req_message'), Auth::user()->id, request('route_id')];
        DB::insert($sql, $fields);
```

User Manual

Getting Started With The Calgary Transit Web App

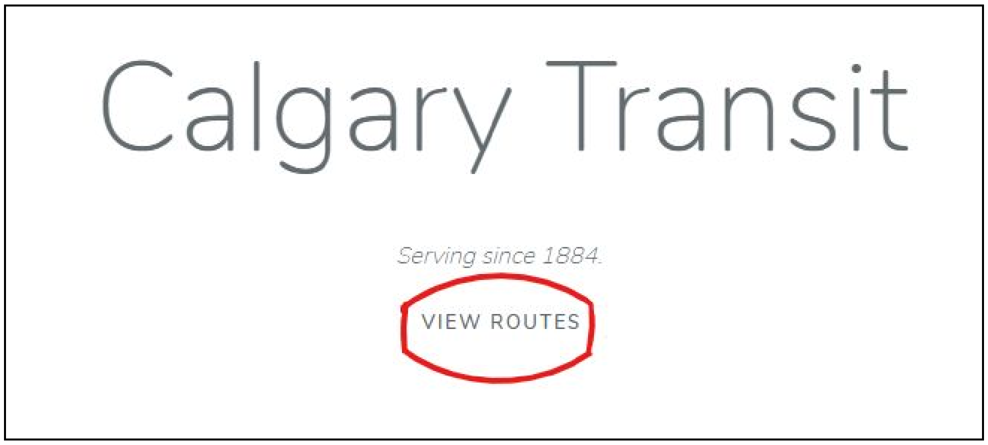
Section 1: General Info

1.1: Routes

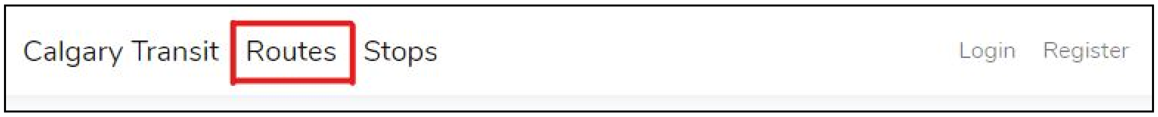
Any user, registered or unregistered, is able to access the detailed viewing of routes. This includes the start and end stop of an entire route, the start and end stops and address of each route's legs, and the information specific to each run of the route throughout the day.

This information is accessed two ways:

The first way is by clicking on the "**View Routes**" button on the website's homepage:



The second way is by clicking on the "**Routes**" tab in the top bar (Note: any "tabs" that are mentioned will be located on the top bar) that is visible from any other page:

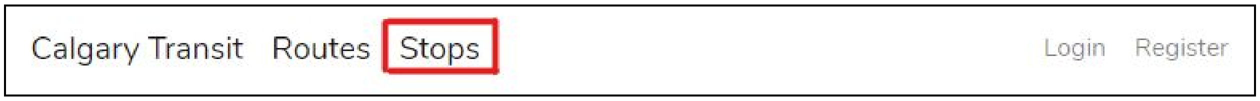


Once on the "Routes" page, the user can then click on the specific route they would like more detailed information about.

1.2: Stops

Any user is able to view the daily schedule of any specific stop. The schedule will show the time that each run of each route is at the chosen stop. The info is sorted by route number, and then by time.

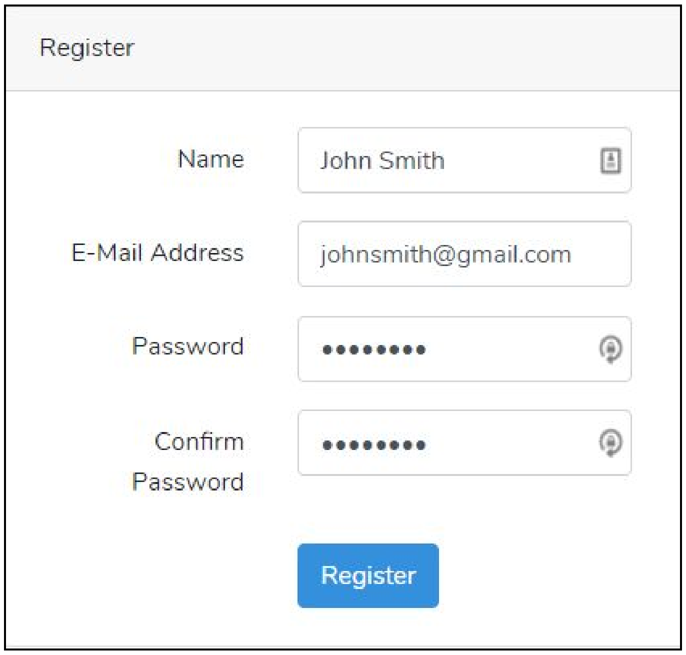
The schedule is accessed by clicking on the "Stops" tab, and then selecting the desired stop number:



Section 2: Client Account

2.1: Registering

Anyone can register for a user account. This is done by clicking on "**Register**" in the top bar, and then entering a **name**, a valid **e-mail** address, and a **password** that has 8 or more characters:



Register

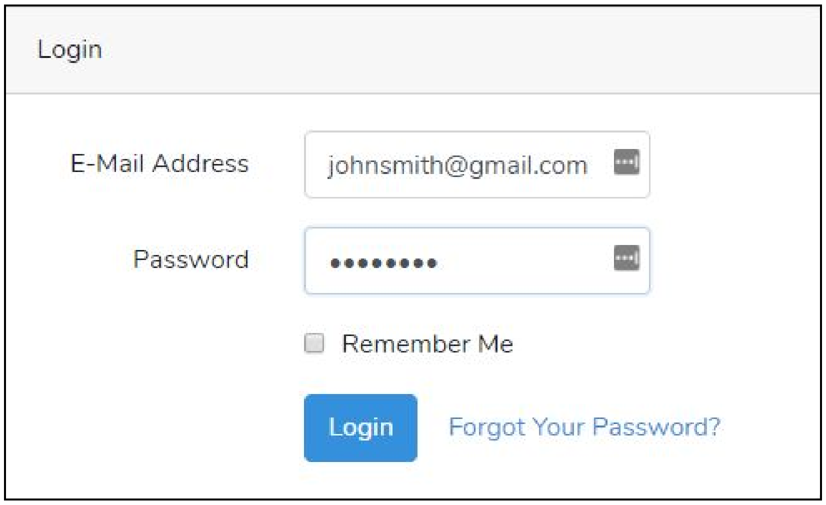
Name	John Smith
E-Mail Address	johnsmith@gmail.com
Password	●●●●●●●●
Confirm Password	●●●●●●●●

By default, the new account will be registered as a "client". The client has an account balance tied to their account, which they can use to pay for their transit tickets.

A new employee must first register for an account before they are added to the employee database. This will be detailed in **Section 3.1**.

2.2: Logging In

Once a user has registered for an account, they may login by clicking "**Login**" in the top bar and then entering their **username** and **password**:



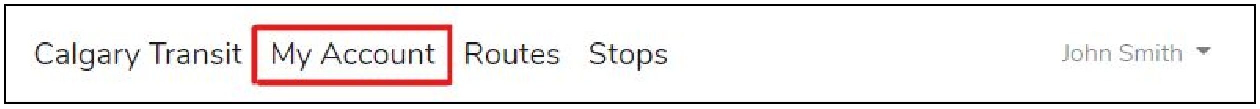
Login

E-Mail Address	johnsmith@gmail.com
Password	●●●●●●●●

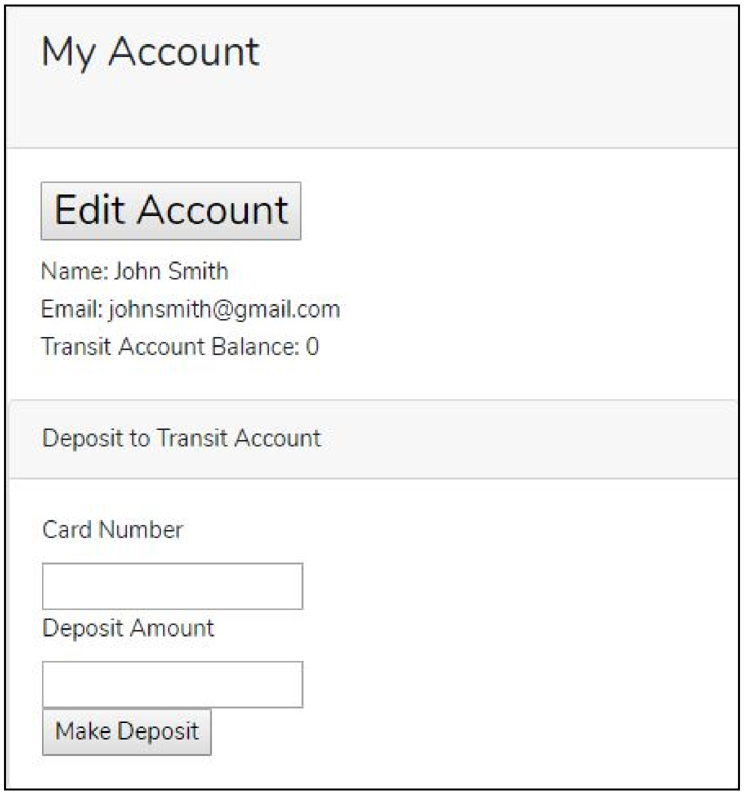
Remember Me

2.3: My Account

Once a user has a registered account, they can access their account info by clicking on the "**My Account**" tab:



This will show a page containing the user's name, email address, and their account balance. This page is also where the user will add funds to their account:



My Account

Edit Account

Name: John Smith
Email: johnsmith@gmail.com
Transit Account Balance: 0

Deposit to Transit Account

Card Number

Deposit Amount

Make Deposit

Funds are added by entering the payment card number, as well as how many dollars they wish to add to their account.

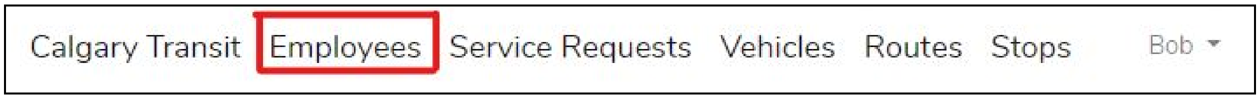
The user may edit their name and/or email by clicking "**Edit Account**" and then clicking "**Update Account Info**"

Section 3: Admin

The admin has the ability to view, edit, and remove employees, vehicles, routes, and stops. They also have the ability to view all the service requests that the analysts have created.

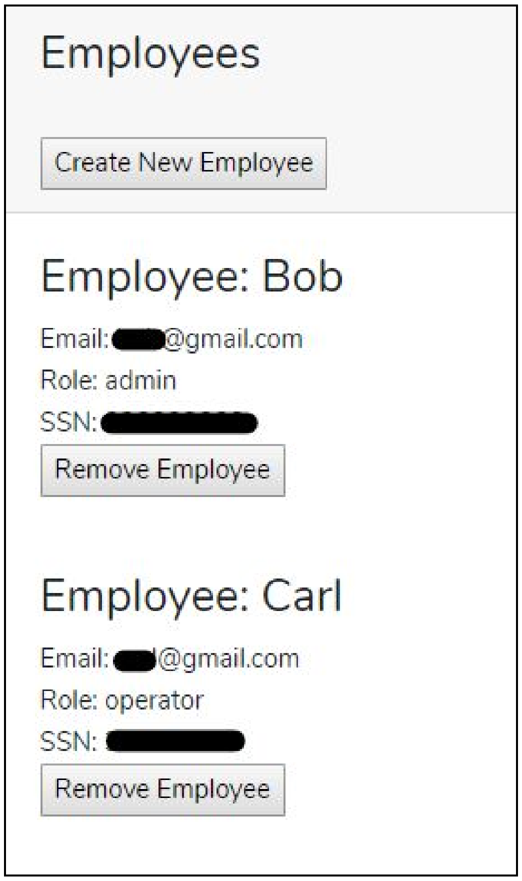
3.1: Employees

The admin views all employees by clicking on the "**Employees**" tab:



All the employees are then listed on the following page. The admin can delete an employee entry by clicking "**Remove Employee**" under the desired employee entry.

They can also change the role of a user account from "client" to an employee by clicking "**Create New Employee**".


Employees

Create New Employee

Employee: Bob

Email: [redacted]@gmail.com

Role: admin

SSN: [redacted]

Remove Employee

Employee: Carl

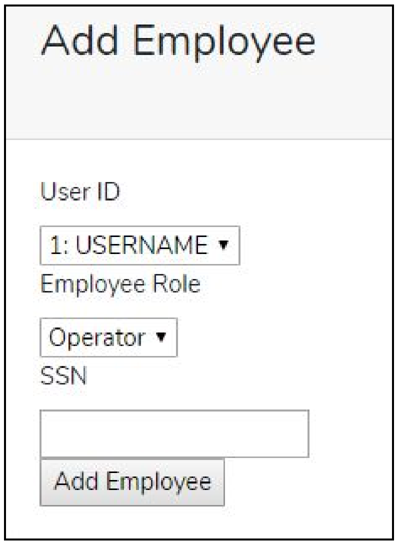
Email: [redacted]@gmail.com

Role: operator

SSN: [redacted]

Remove Employee

On the following page, they will then select the **User ID** of the user to be changed, the **employee role** that this user will take, and enter the employee's **SSN**.


Add Employee

User ID: 1: USERNAME

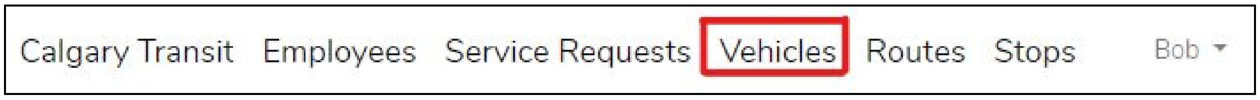
Employee Role: Operator

SSN:

Add Employee

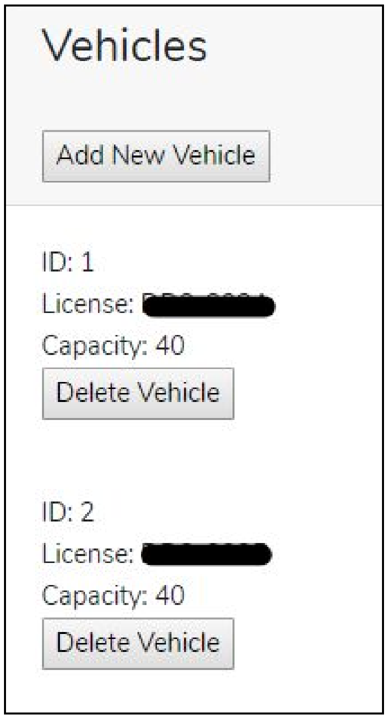
3.2: Vehicles

The admin has the ability to view all vehicles stored in the database by clicking on the "**Vehicles**" tab:



The "vehicles" page shows a list of all stored vehicles, consisting of their ID, license, and capacity. The admin has the ability to delete any stored vehicle by clicking "**Delete Vehicle**" under the desired vehicle.

The admin can also add new vehicles to the database by clicking "**Add New Vehicle**"

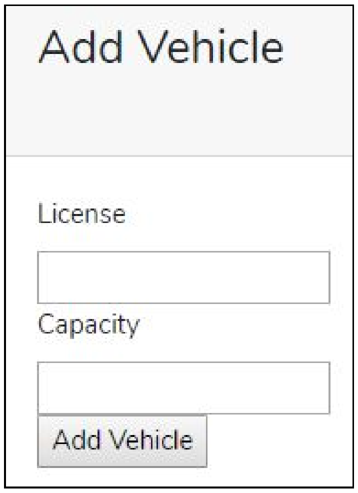

Vehicles

Add New Vehicle

ID: 1
License: [redacted]
Capacity: 40
Delete Vehicle

ID: 2
License: [redacted]
Capacity: 40
Delete Vehicle

On the following page, the admin will need to fill in the **license** and maximum **capacity** of the vehicle they wish to add to the database.


Add Vehicle

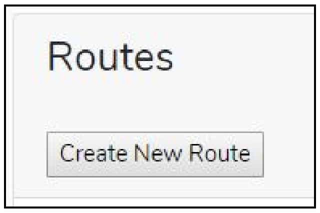
License

Capacity

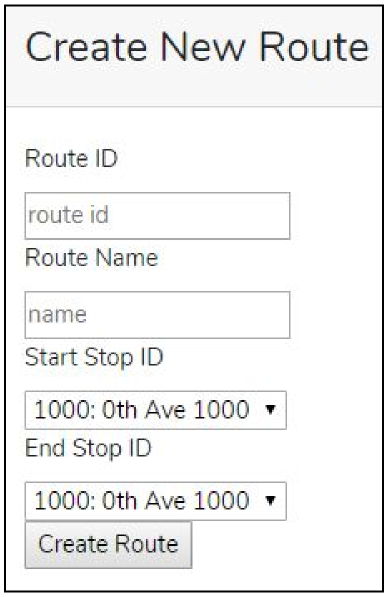
Add Vehicle

3.3 Routes

The admin's view of the main "Routes" page is almost identical to that of a "non-admin" user. The difference is that the admin has the ability to add a new route by clicking "**Create New Route**"



On the following page, the admin is required to enter the route's **ID** and **Name** and then select the **Start Stop ID** and **End Stop ID** from a dropdown menu of all existing stops.


Create New Route

Route ID: route id

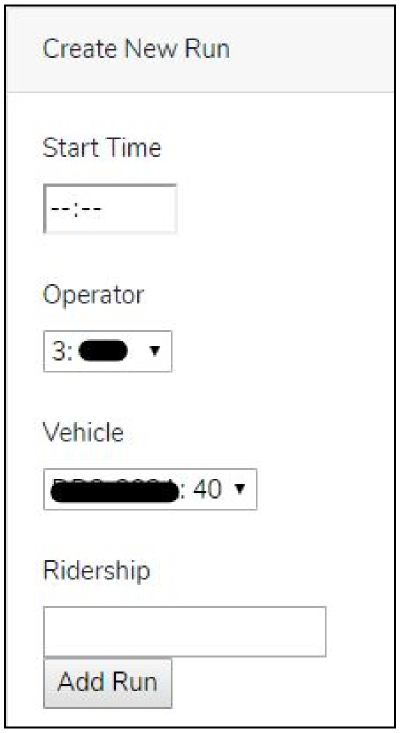
Route Name: name

Start Stop ID: 1000: 0th Ave 1000

End Stop ID: 1000: 0th Ave 1000

Once the admin selects a specific route from the list of routes, they can view details about that route's legs and runs. The admin has the ability to delete a run by clicking "**Delete Run**" below the desired run. They can also delete a leg from the route by clicking "**Delete Leg**" under the desired leg.

The admin can add runs to the route by filling in the fields at the bottom of the page:


Create New Run

Start Time: --:--

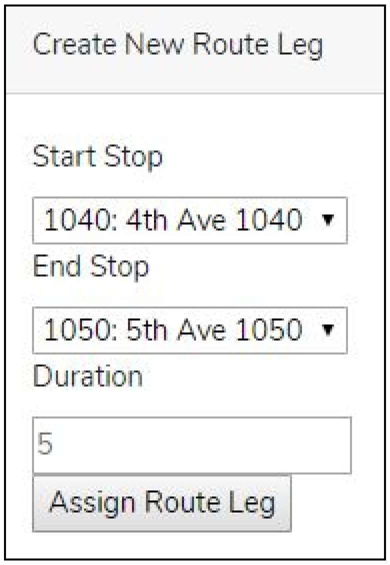
Operator: 3: [redacted]

Vehicle: [redacted]: 40

Ridership:

The admin can edit the route by selecting "**Edit Route**" which brings up a page almost identical to the "Create New Route" page, but with "**Update Route**" and "**Delete Route**" buttons.

If the admin wishes to add a leg to the existing route, they click on "**Add Route Leg**", select the **Start Stop** and **End Stop** from a dropdown, and enter the **duration**, in minutes, of the leg on the following page:


Create New Route Leg

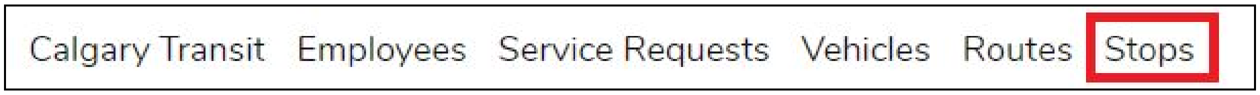
Start Stop: 1040: 4th Ave 1040

End Stop: 1050: 5th Ave 1050

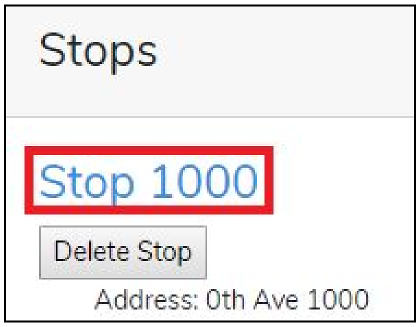
Duration: 5

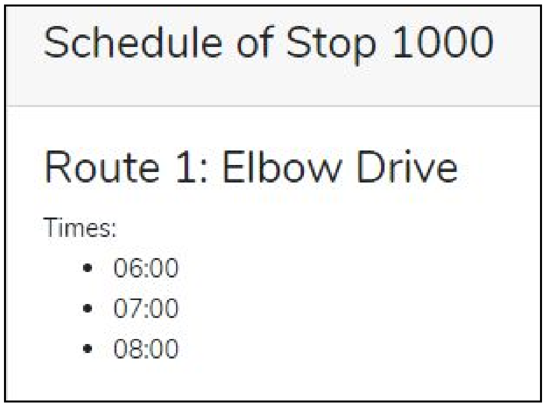
3.4 Stops

The admin's view of the"Stops" page is almost identical to that of a "non-admin" user. The difference is that the admin has the ability to add a new route by clicking "**Create New Stop**" at the bottom of the page, as well as "**Delete Stop**".



If the admin clicks on the stop id, they will be directed to a page containing the **time** at which each vehicle arrives at the stop, sorted by **route number**, and then by time.





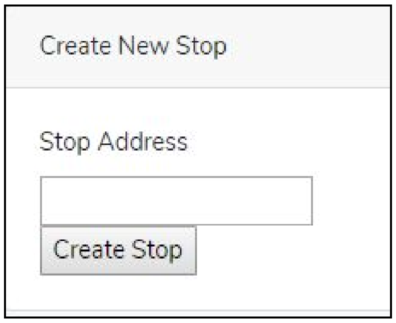
Schedule of Stop 1000

Route 1: Elbow Drive

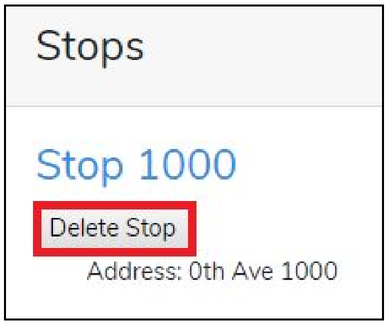
Times:

- 06:00
- 07:00
- 08:00

An admin can **create a stop** by filling in the **stop address** field at the bottom of the page and then clicking "**Create Stop**". The stop ID will be set automatically.

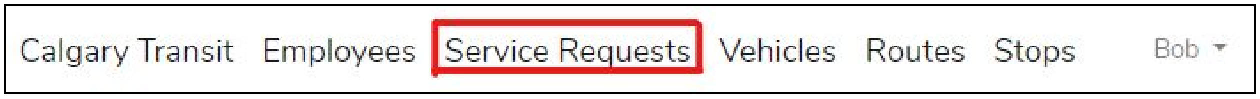


The admin can **delete a stop** by clicking the "**Delete Stop**" button below the desired stop. This will also delete all route legs and routes which contain that stop either as a start stop or end stop.

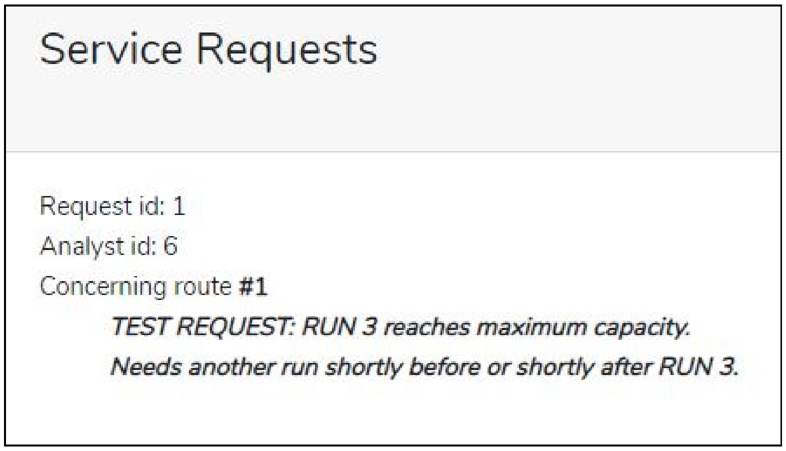


3.5: Service Requests

The admin can view service requests by clicking on the "**Service Requests**" tab.



This will bring them to a page where all the service requests created by analysts are listed. They can see the **ID** of the **request**, the **ID** of the **analyst** that created the request, the **route number** that the request is for, and the analyst's **message.**


Service Requests

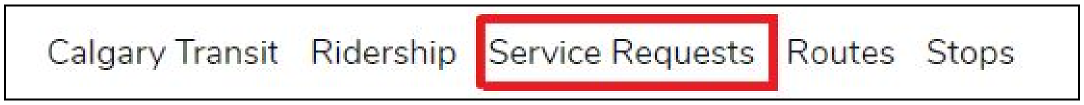
Request id: 1
Analyst id: 6
Concerning route **#1**
TEST REQUEST: RUN 3 reaches maximum capacity. Needs another run shortly before or shortly after RUN 3.

The creation of these requests will be explained in **section 4.1**.

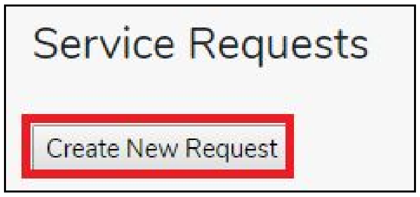
Section 4: Analyst

4.1: Service Requests

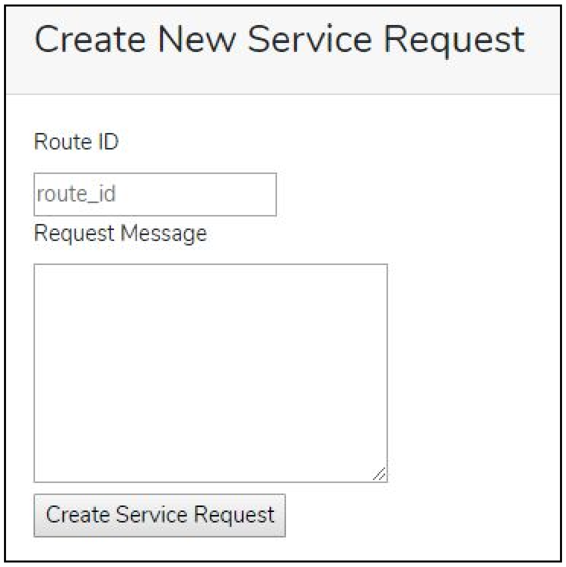
The analyst can view and create service requests by clicking on the “**Service Requests**” tab.



An analyst has access to the same information in service requests as an admin, but they can also create a new service request by clicking the “**Create New Request**” button:

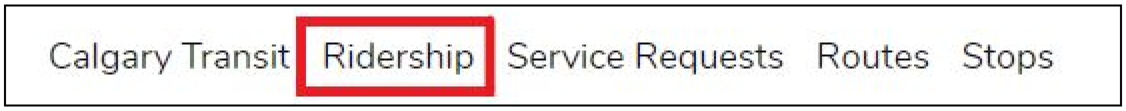


This takes the analyst to a form, which they can fill out by entering the **Route ID** the request is for, and the **message** for the request. They then click on “**Create Service Request**”, and then the request will be visible by all admins and analysts.

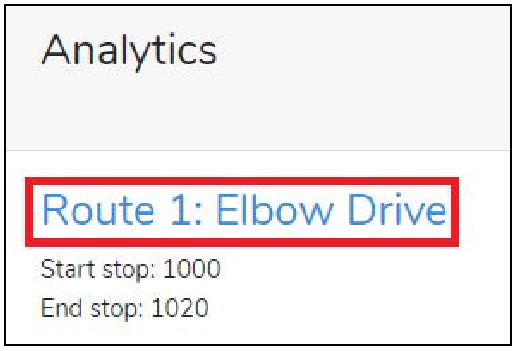


4.2: Ridership

The analyst can access the **Ridership** of each run by clicking the "**Ridership**" tab:



The page shows a list of all the routes. The analyst can then select any of the routes to see the ridership of that route's runs.



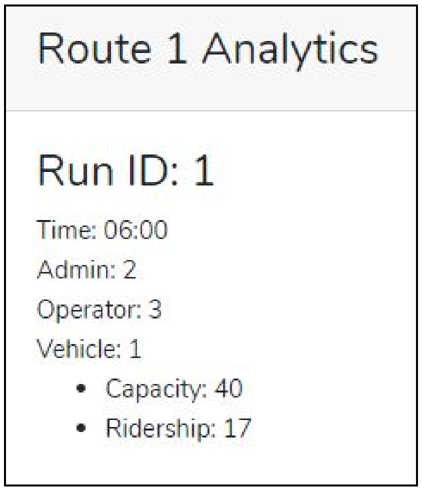
Analytics

Route 1: Elbow Drive

Start stop: 1000

End stop: 1020

Once the route is selected, the analyst can see all the runs, as well as their **start time**, the **admin** that scheduled the run, the run's **operator**, the assigned **vehicle** and it's maximum **capacity**, and the **ridership** for the run.



Route 1 Analytics

Run ID: 1

Time: 06:00

Admin: 2

Operator: 3

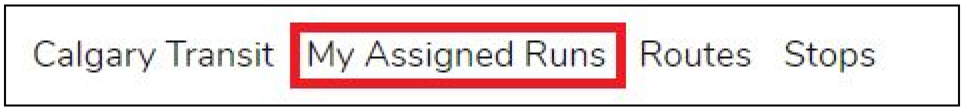
Vehicle: 1

- Capacity: 40
- Ridership: 17

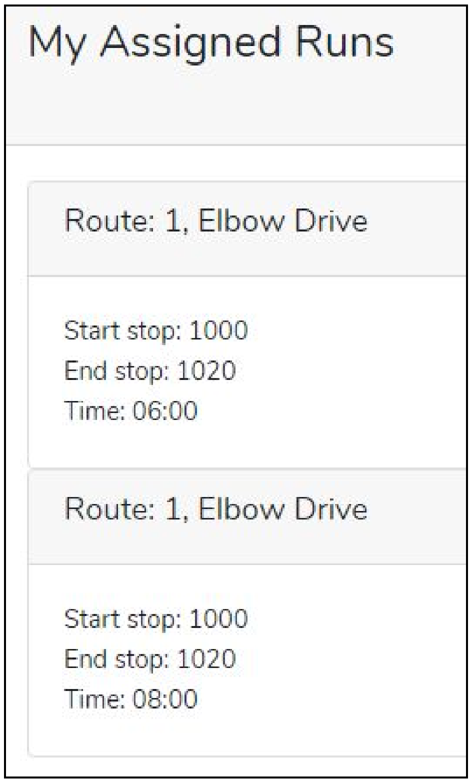
Section 5: Operator

5.1: Assigned Runs

The operator has the ability to login and see all the runs to which they have been assigned by accessing the tab which is exclusively visible to an operator: **"My Assigned Runs"**.



The link will redirect the operator to a page which lists all the runs to which they are assigned.



My Assigned Runs

Route: 1, Elbow Drive

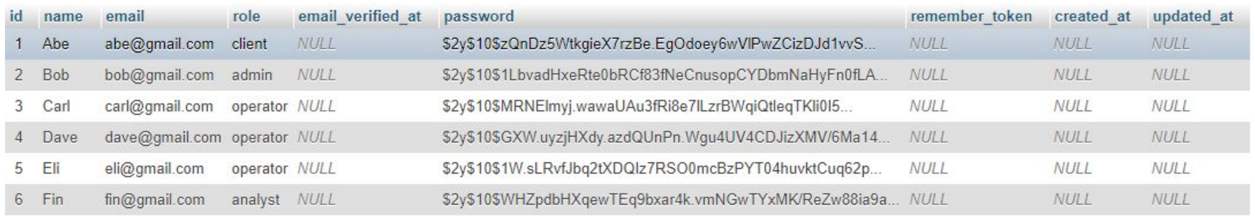
Start stop: 1000
End stop: 1020
Time: 06:00

Route: 1, Elbow Drive

Start stop: 1000
End stop: 1020
Time: 08:00

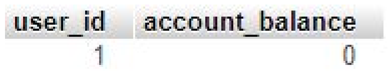
APPENDIX A: Sample Data Records

Figure 1: Users



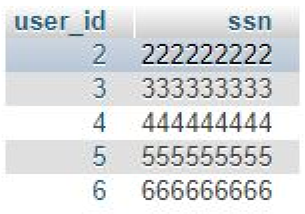
id	name	email	role	email_verified_at	password	remember_token	created_at	updated_at
1	Abe	abe@gmail.com	client	NULL	$2y$10$zQnDz5WtkgieX7rzBe.EgOdoey6wVlPwZCizDJd1vvS...	NULL	NULL	NULL
2	Bob	bob@gmail.com	admin	NULL	$2y$10$1LbvadHxeRte0bRCf83fNeCnusopCYDbmNaHyFn0fLA...	NULL	NULL	NULL
3	Carl	carl@gmail.com	operator	NULL	$2y$10$MRNElmyj.wawaUAu3fRi8e7lLzrBWqiQtleqTKli0I5...	NULL	NULL	NULL
4	Dave	dave@gmail.com	operator	NULL	$2y$10$GXW.uyzjHXdy.azdQUnPn.Wgu4UV4CDJizXMV/6Ma14...	NULL	NULL	NULL
5	Eli	eli@gmail.com	operator	NULL	$2y$10$1W.sLRvfJbq2tXDQlz7RSO0mcBzPYT04huvktCuq62p...	NULL	NULL	NULL
6	Fin	fin@gmail.com	analyst	NULL	$2y$10$WHZpdbHXqewTEq9bxar4k.vmNGwTYxMK/ReZw88ia9a...	NULL	NULL	NULL

Figure 2: Clients



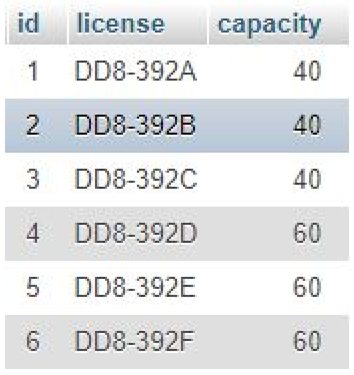
user_id	account_balance
1	0

Figure 3: Employees



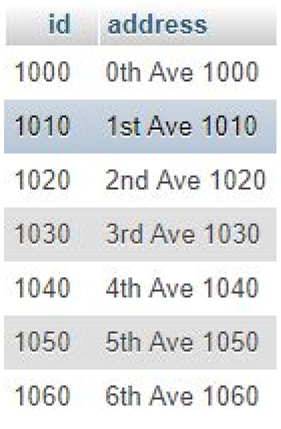
user_id	ssn
2	222222222
3	333333333
4	444444444
5	555555555
6	666666666

Figure 4: Vehicles



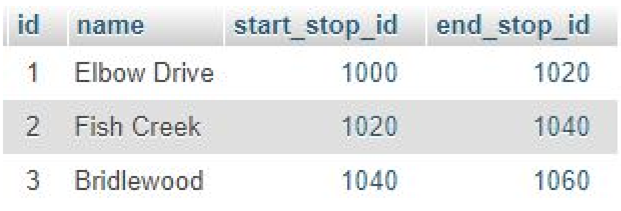
id	license	capacity
1	DD8-392A	40
2	DD8-392B	40
3	DD8-392C	40
4	DD8-392D	60
5	DD8-392E	60
6	DD8-392F	60

Figure 5: Stops



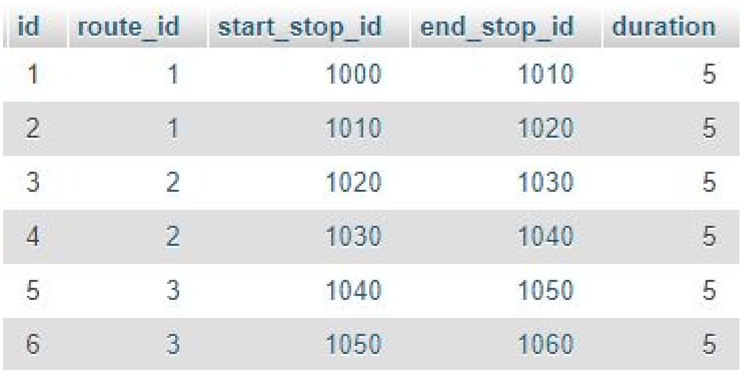
id	address
1000	0th Ave 1000
1010	1st Ave 1010
1020	2nd Ave 1020
1030	3rd Ave 1030
1040	4th Ave 1040
1050	5th Ave 1050
1060	6th Ave 1060

Figure 6: Routes



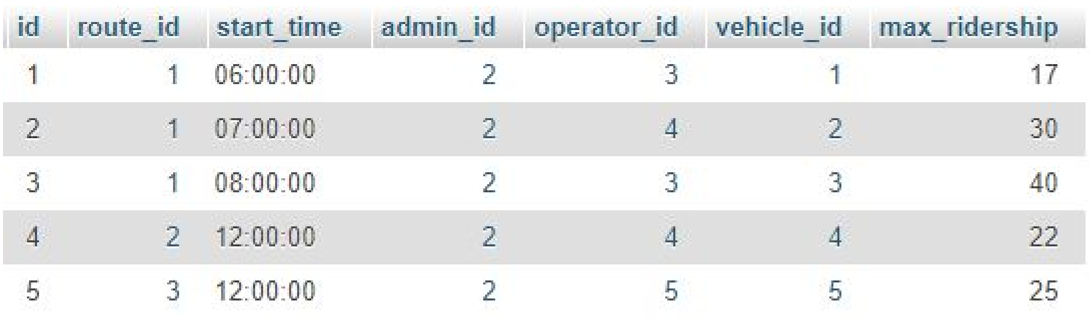
id	name	start_stop_id	end_stop_id
1	Elbow Drive	1000	1020
2	Fish Creek	1020	1040
3	Bridlewood	1040	1060

Figure 7: Route_legs



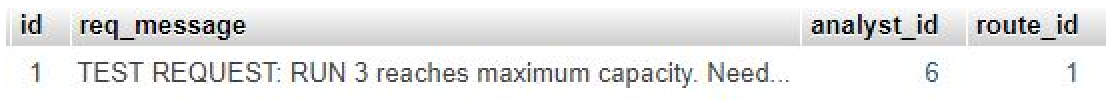
id	route_id	start_stop_id	end_stop_id	duration
1	1	1000	1010	5
2	1	1010	1020	5
3	2	1020	1030	5
4	2	1030	1040	5
5	3	1040	1050	5
6	3	1050	1060	5

Figure 8: Runs



id	route_id	start_time	admin_id	operator_id	vehicle_id	max_ridership
1	1	06:00:00	2	3	1	17
2	1	07:00:00	2	4	2	30
3	1	08:00:00	2	3	3	40
4	2	12:00:00	2	4	4	22
5	3	12:00:00	2	5	5	25

Figure 9: Requests



id	req_message	analyst_id	route_id
1	TEST REQUEST: RUN 3 reaches maximum capacity. Need...	6	1