CPSC 471: Initial Draft Design

Due on March 21
Group Number 3
TA: Ibrahim Karakira

Tanner Collin, Jordan Heinrichs, Ali Waseem

Contents

1 Introduction

This report contains the initial draft design and a progress report for our project Renti, a peer to peer vehicle renting service. The current progress will discuss the technologies chosen for the framework and much work is done implementing it. As our project uses an Object Relational Mapping (ORM) to access our database we have an object orientated model which will be outlined.

2 Current Progress and Technologies

For the back end of the project we are using Node.js with Express.js, Knex, and Bookshelf (this will be our ORM that handles our queries). The database is using SQLite3 with Bookshelf.js to provide the ORM interface. The back end is finished and the API is tested. As the front end is implemented there will likely need to be some changes to the API but it is in a state where work can be started on the front end for the web application and possibly the mobile application.

The front end will be implemented using React.js and Semantic-UI. The boiler plate code will be taken from Ali Waseem's blog including functions like user authentication. Each project member will be responsible for creating components to interact with the back end API. Work is currently started and the projected completion date for the front end is end of March.

3 Object Orientated Model

The object orientated model is is shown in Figure 1. For each of the routes the SQL query is listed.

3.1 Users

The model of the user:

3.1.1 The public routes:

get /api/users/

```
SELECT u.uid, u.first_name, u.last_name, u.address, u.username, u.
    email, u.image, u.summary f.* FROM USER AS u, FEEDBACK_USERS as f
    WHERE u.uid=f.user_has;
```

get /api/users/:id

```
SELECT uid, first_name, last_name, address, username, email, image,
    summary FROM USER WHERE uid="id";
```

post /api/users/signup

```
INSERT INTO USER (image, username, password, first_name, last_name,
    address, summary, date_of_birth, email)
VALUES ('URL_OF_USER_PHOTO', 'tcollin', 'test', 'Tanner', 'Collin', '
    123 place rd NW', 'Hi. I love for you to rent one of my cars!',
    730869558, 'test@someemail.com');
```

post /api/users/signin

```
SELECT username, password FROM USER where username='tcollin', password
    ='test';
```

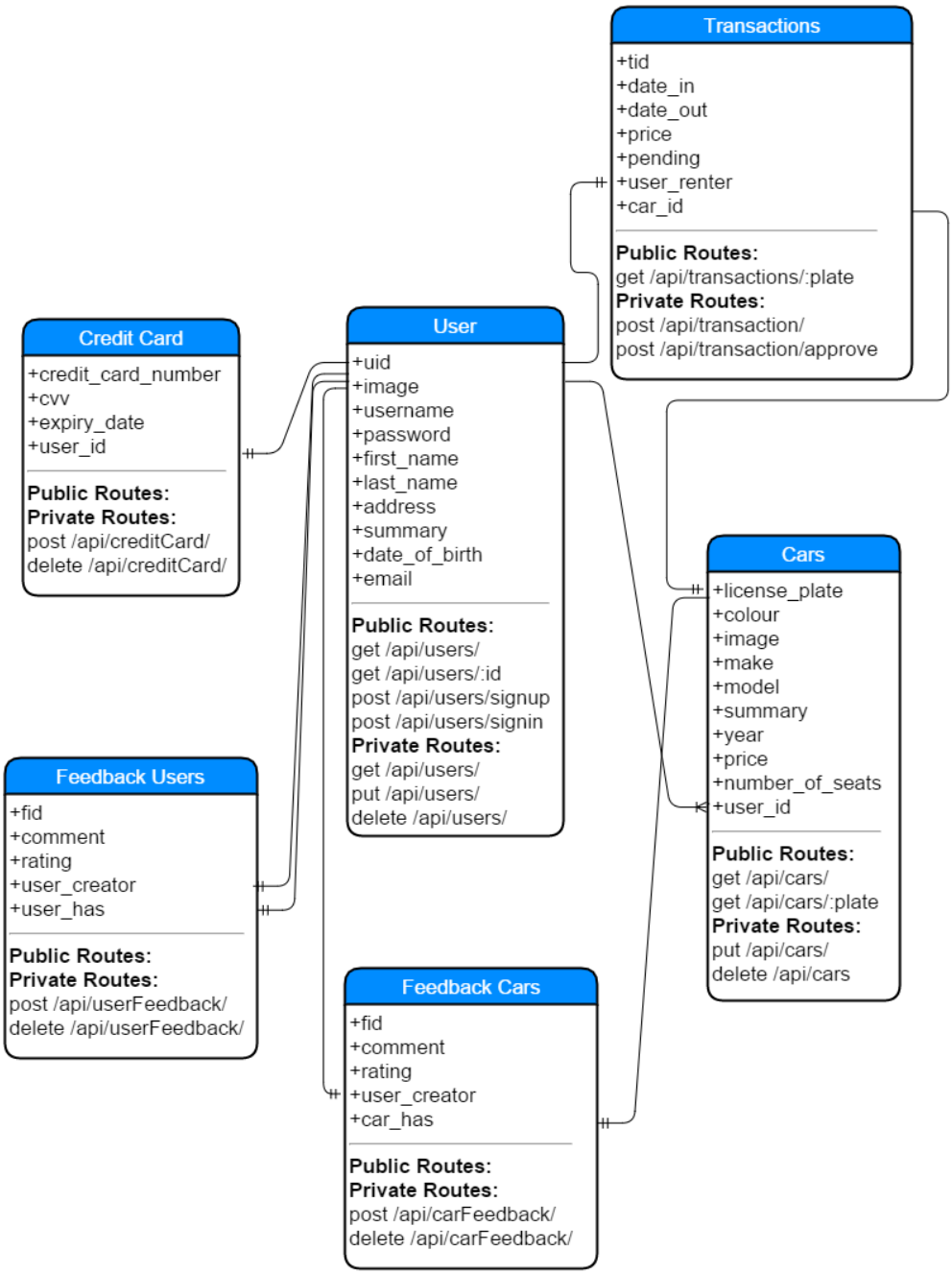

Figure 1: Database Object Orientated Model

3.1.2 The private routes:

get /api/users/

```
SELECT * FROM USER where uid='1', username='tcollin', password='test';
```

put /api/users/

```
UPDATE USER SET summary='hi my name is tanner' WHERE uid='1';
```

delete /api/users/

```
DELETE FROM USER WHERE uid='1', username='tcollin', password='test';
```

3.2 Credit Card

The model of the credit card, there are no public routes for credit cards because of security reasons.

3.2.1 The private routes:

post /api/creditCard/

```
INSERT INTO CREDIT_CARD (credit_card_number, cvv, expiry_date, user_id
    )
VALUES (347249711260948, 433, 0617, 1);
```

delete /api/creditCard/

```
DELETE FROM CREDIT_CARD as c INNER JOIN USER as u ON c.user_id = a.uid
     WHERE
a.username='tcollin' AND a.password='test';
```

3.3 Cars

The model of the cars, contains both public and private routes.

3.3.1 The public routes:

get /api/cars/

```
SELECT c.*, f.* FROM CARS AS c, FEEDBACK_CARS as f WHERE c.
    license_plate=f.car_has;
```

get /api/cars/:plate

```
SELECT c.*, f.*  FROM CARS AS c, FEEDBACK_CARS as f WHERE
    license_plate='plate' AND c.license_plate=f.car_has;
```

3.3.2 The private routes:

post /api/cars/

```
INSERT INTO CARS (license_plate, model, make, year, number_of_seats,
    price, colour, image, summary, user_id)
VALUES ('892WSM', 'Focus RS', 'Ford', 2016, 4, 200, 'Blue', '
    URL_OF_CAR_PHOTO', 'This car is fast', 2);
```

put /api/cars/

```
UPDATE CARS SET model='VW', make='Golf R' WHERE license_plate='892WSM'
     AND user_id='2';
```

delete /api/cars/

```
DELETE FROM CARS AS c WHERE c.uid='2' AND c.license_plate='892WSM';
```

3.4 Transactions

3.4.1 The public routes:

get /api/transactions/

```
SELECT * FROM TRANSACTIONS AS t WHERE t.plate='123ABC';
```

3.4.2 The private routes:

post /api/transactions/

```
INSERT INTO TRANSACTIONS (date_in, date_out, price, pending,
   user_renter, car_id)
VALUES ('2016-3-27', '2016-3-21', 40, 1, '2','123ABC');
```

post /api/transactions/approve

```
UPDATE TRANSACTIONS as t SET pending='0' WHERE c.tid='10' AND EXISTS (
   Select * FROM CARS as c WHERE t.license_plate=c.license_plate AND c
   .user_id='2');
```

3.5 Car Feedback

There are no public routes with car feedback. The feedback is returned with a car get requests.

3.5.1 The private routes:

post /api/carfeedback/

```
INSERT INTO FEEDBACK_CARS (comment, rating, user_creator, car_has)
VALUES ('This car is really shiny.', '5', '2', '892WSM');
```

delete /api/carfeedback/

```
DELETE FROM FEEDBACK_CARS AS f WHERE f.fid='7' AND f.user_creator='2';
```

3.6 User Feedback

There are no public routes with user feedback. The feedback is returned with a user get request.

3.6.1 The private routes:

post /api/userfeedback/

```
INSERT INTO FEEDBACK_USERS (comment, rating, user_creator, user_has)
VALUES ('This user is not friendly.', '2', '2', '1');
```

delete /api/userfeedback/

```
DELETE FROM FEEDBACK_USERS WHERE fid='6' AND user_creator='2';
```

4 Conclusion

The above model and SQL queries are demonstrate how we will access and manipulate the database for our website Renti. The SQL queries will be automatically implemented by the ORM but they accurately demonstrate the database actions that the API presents.