SWOOP PROJECT PRESENTATION FOR ABHISHEK...(AND STUDENTS)

Oscar Diaz Vega, Shreyass Prem Sankar, Arjun Bemarkar

Group #9

TABLE OF CONTENTS

WHOA!

There's a lot of Carbon >.<
...anyways

01

ABOUT SWOOP

More than just a ride-share service

INTRODUCTION

In the United States, automobiles are the primary source of transportation. There are both financial and environmental key factors with traveling by automobile. Swoop provides clear data visualization for these critical factors for any planned trip. With Swoop, you can see a visual representation of carbon emission levels that will be exerted for a planned trip, along with the cost of the trip by using average gas prices. Swoop also provides a carpool option to address carbon pollution levels.

Swoop Application Timeline

01 Login/Sign up page

User can sign up and then login

02 Select type of user

User can be either a rider or driver

03 Create trip/carbon goal

Driver can create planned trips and rider can create a carbon saving goal

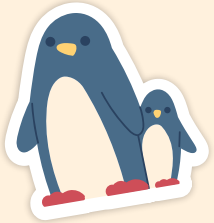
04 View impact/ Request ride

Rider can either view data graphs for their planned trips or book a ride

05 Join ride

After requesting a ride, user can join the ride

Aww look at da cute penguinos



02

Design Patterns

We used not one....but TWO WHOLE DESIGN PATTERNS :O

MVC

- Our back-end implements the models and controllers, while our front-end fetches data from the back-end controllers to update the views using React.js

User Model

```
public void setFullName(String fullName) {
    this.fullName = fullName;
}

public void setEmail(String email) {
    this.email = email;
}

public void setPassword(String password) {
    this.password = password;
}

public void setCurrentUserType(UserType currentUserType) {
    this.currentUserType = currentUserType;
}

public String getFullName() {
    return fullName;
}

public String getEmail() {
    return email;
}
```

User API Request Example


```
{
    "2": {
        "CO2": 20.0,
        "start": "SJSU",
        "end": "SFO",
        "distance": 30.0,
        "users": [
            {
                "fullName": "Arjun Bemarkar",
                "email": "a@b.com",
                "password": "lol",
                "rides": [
                    2
                ],
                "currentUserType": null,
                "inRide": true
            }
        ],
        "id": 2,
        "startCoordinates": [
            37.33548,
            -121.89303
        ],
        "endCoordinates": [
            10.0,
            10.0
        ],
        "status": null
    }
}
```


User Controller

```
@CrossOrigin(origins = "http://localhost:3000")
@GetMapping(path = "{email}")
public Optional<User> getUserByEmail(@PathVariable("email")
    return userService.getUserByEmail(email);
}
```

User Views

```
async function getUser(email) {
  try{
    axios.get('http://localhost:8080/api/v1/user/' + email)
```


```
"fullName": "Oscar Diaz Vega",
"email": "oscar.diazvega@sjsu.edu",
"password": "Sjsu123))",
"rides": [],
"currentUserType": null,
"inRide": false,
"carbonGoal": 0.0
```

Singleton

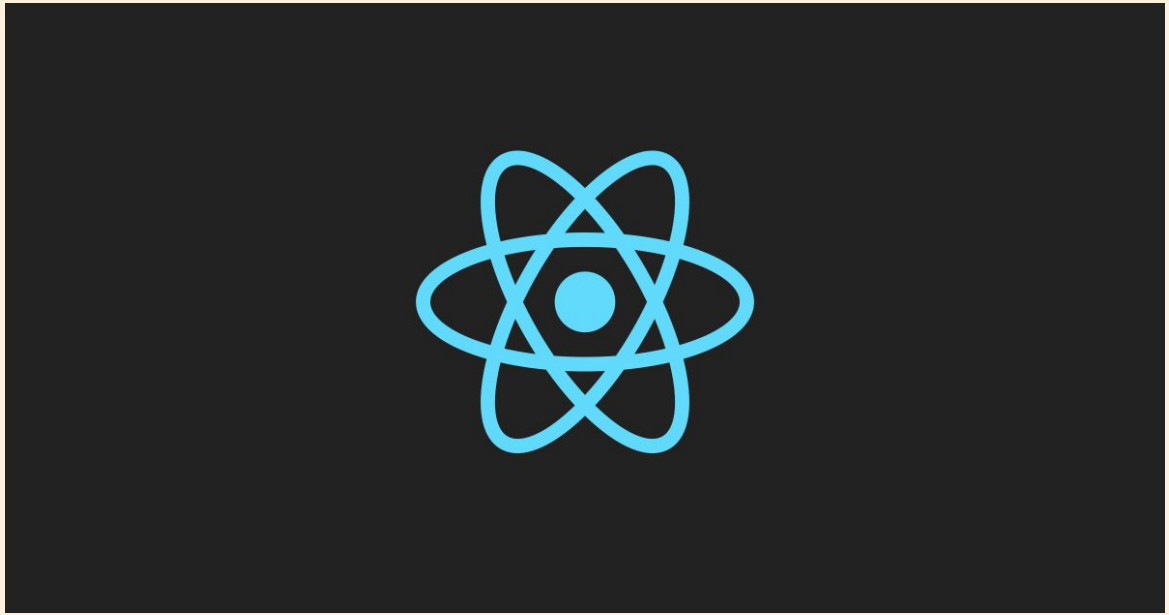
- Since we only want one database to be created, we saw the perfect opportunity to implement the singleton design pattern for our database.

```
private Database(){ }

public static synchronized Database getInstance(){
    if(dbSingleton == null){
        dbSingleton = new Database();
        DB = new HashMap<>();
    }
    return dbSingleton;
}
```

FRONT-END

- Our front-end was built with React.js



BACK-END

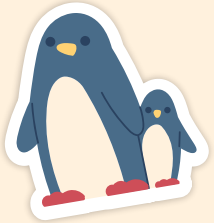
- Our back-end was built with Spring Boot

BING MAPS API

- For the trip calculations, we used Microsoft's Bing Maps API

```
function calculateDistanceBetweenOriginAndDest() {
  axios
  .get(
    `https://dev.virtualearth.net/REST/v1/Routes/DistanceMatrix?origins=${originLat},${originLong}&destinations=${destLat},${destLong}&travelMode=driving&key=
  )
  .then((response) => {
    setTotalDistance(
      convertKilometersToMiles(
        response.data.resourceSets[0].resources[0].results[0].travelDistance
      )
    );
  });
```

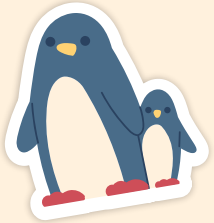
```
axios
  .get(
    `http://dev.virtualearth.net/REST/v1/Locations/US/${originState}/-/${originCity}/-?&key
  )
  .then((response) => {
    setOriginLat(
      response.data.resourceSets[0].resources[0].geocodePoints[0]
        .coordinates[0]
    );
    setOriginLong(
      response.data.resourceSets[0].resources[0].geocodePoints[0]
        .coordinates[1]
    );
  });
```



03

Project Demo

Time for the live demo!



04

Q&A

Any questions?