San Jose State University
Fall 2017

KAYAK Project Report
CMPE 273 – Team 7

SUBMITTED BY:
Aditya Parmar
Dishant Kimtani
Meenakshi Paryani
Siddharth Suthar
Sunil Tiwari

SUBMITTED TO:
Prof. Simon Shim

Project Contribution:

- **Aditya Parmar:**
  I helped in creating the design of the project. I created the user interface of the entire Kayak project. I also integrated various modules such as search, booking, payment, user, admin etc. with the UI. I have created user account verification service using email. I have enhanced and managed user session. I have also maintained constant redux state throughout the project.

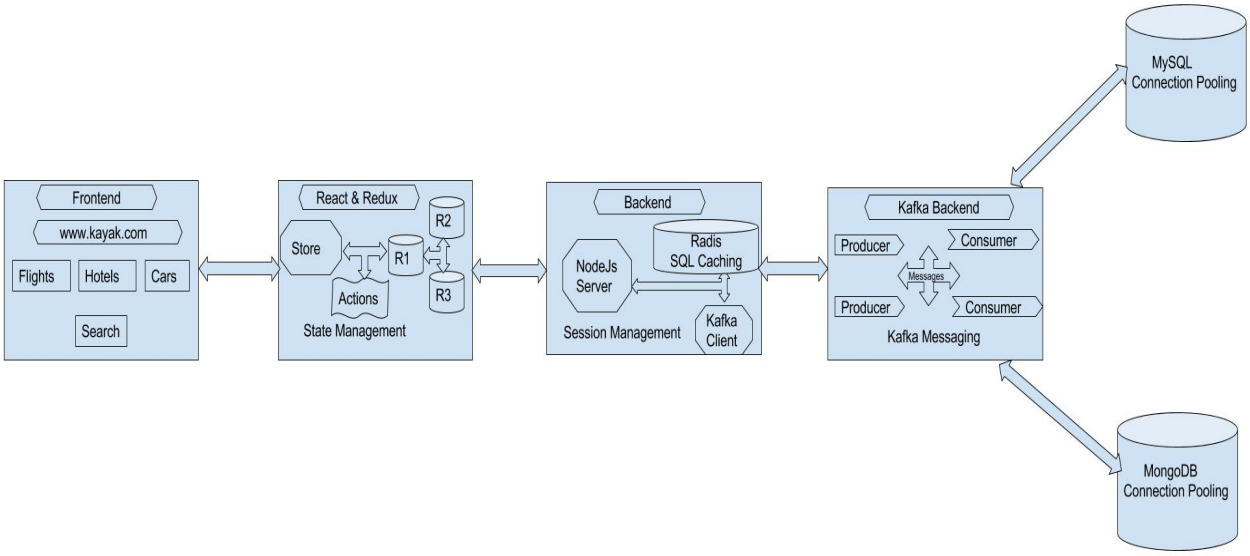
- **Dishant Kimtani:**
  I created the MongoDB schema for kayak. We are storing car, flight and hotel information in MongoDB. Also, we have created separate databases for vendor information in MongoDB. I also created the backend API's and corresponding Kafka services for Searching the flights, hotels and car. I also created the backend API's for adding/deleting/listing the vendors**.** I have also created connection pooling for MongoDB.

- **Meenakshi Paryani:**
  I created the MySQL schema for kayak. We are storing user, logging, user activity and booking information in MySQL. Also, I also created the backend API's and corresponding Kafka services for booking and payments of the flights, hotels and car booking. I also created the front-end components for booking of a flight/hotel and car and integrated with the backend. I also created provision for caching the SQL queries using Redis server. I have also created connection pooling for MySQL.

- **Siddharth Suthar:**
  I created the backend API's and corresponding Kafka services for user and admin module. I also created user authentication service using PassportJS. Also, i created third party (google) authentication using PassportJS. I have also done various other validation such as password encryption and zip code validation.

- **Sunil Tiwari:**
  I created the user analytics & state management part of the project. I create the initial state management for the project using the Redux. I also create a pro provision to track activity of the user such as button click, page navigation etc. I also analyzed the information stored in the database to draw meaningful analytics out it. I also created the dashboard for the admin module.

System Design:



Object Management Policy:

Overall architecture of the system consists of following components

- **Frontend:** The front end of the system is an interactive UI built in React to take the input from the user and respond to the user events and actions.
- **Zip Validation:** Zip is validated for each user while entering the address information.

- **React & Redux:**
  - **State management:** The state of the system is managed using the Redux (Store). Redux provides a predictable state managements architecture for web applications. Events dispatch actions and actions invoke reducers to modify the state of the system. The in redux state is stored as a Single object tree inside the store. There can be only one store in an application.
  - **React State is stored as follows:**
    - User Profile:{
      isLoggedIn,
      firstname,
      lastname,
      isadmin,
      email,
      address,
      zipcode,
      phonenumber,
      imgpath…}

- User History:{UserHistoryInformation…}
- User Search:{UserSearchInformation…}
- User Booking:{ UserBookInformation…}
- Vendor:{UserVendorInformation…}

- **Backend:**

  - **Authentication:** The user authentication process is handled through the PasspotJS. Passport is authentication middleware for Node.js.
  - **API management:** It receives calls to signUp, signIn, Search, Book, History, Profile and other APIs and generates appropriate messages for KAFKA producer.
  - **Encryption**: Encryption is done using the 'crypto' library available for the NodeJS.
  - **Automatic Mail:** We have created Automatic email service for the newly registered users using the Gmail's pop3 service.
  - **Session management:** The session of the system is managed using the Redux provides a predictable state managements architecture for web applications. Events dispatch actions and actions invoke reducers to modify the state of the system. The in redux state is stored as a Single object tree inside the store. There can be only one store in an application

"Heavyweight" Resources Management Policy:

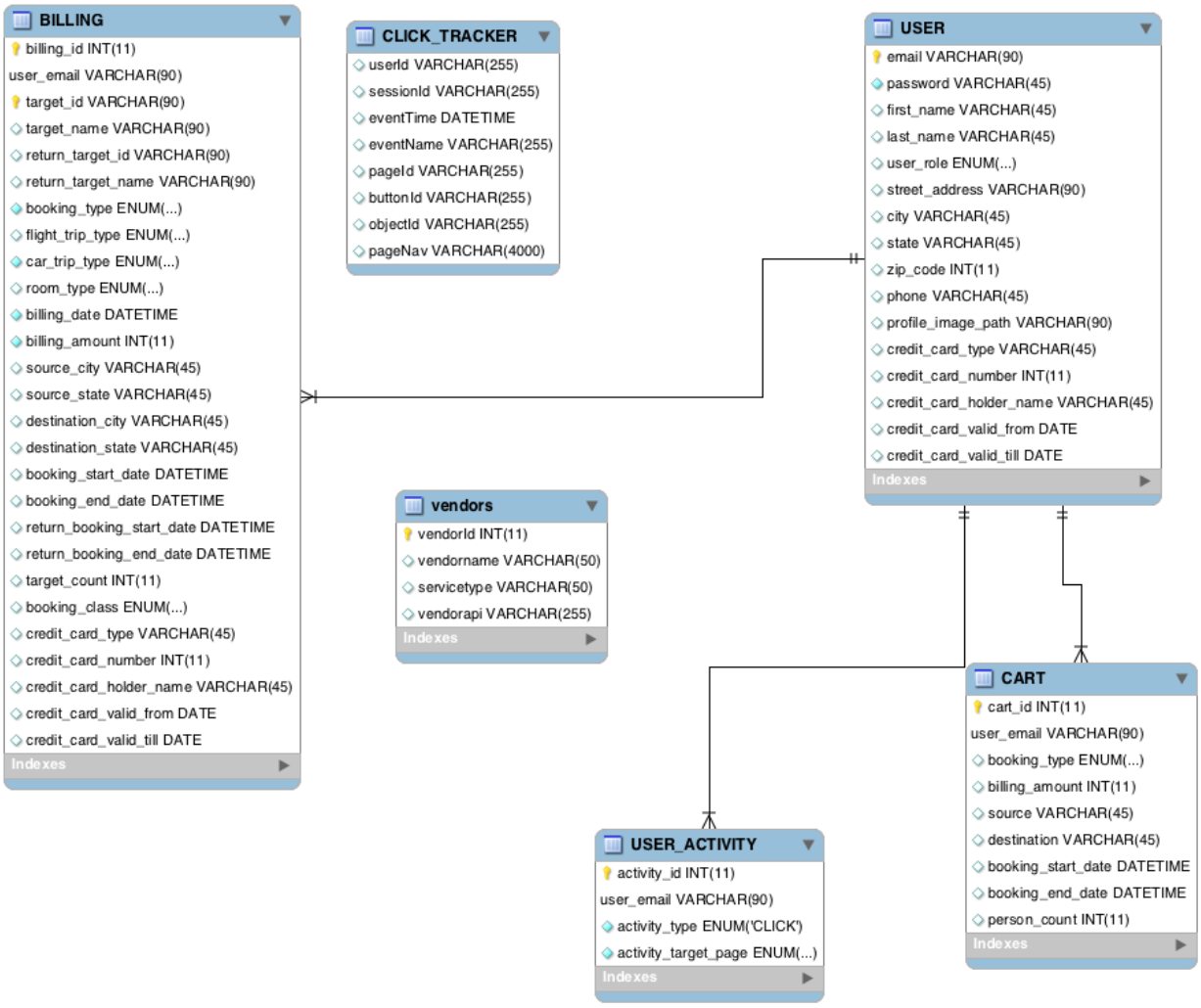
- **Kafka-Backend:**
  - **Messaging system:** Kafka is used as a messaging system. It allows you to create multiple consumers and multiple producers. Producers and consumers can subscribe to various topics in Kafka hence send and receive messages.
  - **Query building:** It consumes Kafka messages and build MongoDB queries. It also builds producers to provide data and acknowledgement from MongoDB.

Read Write Policy:

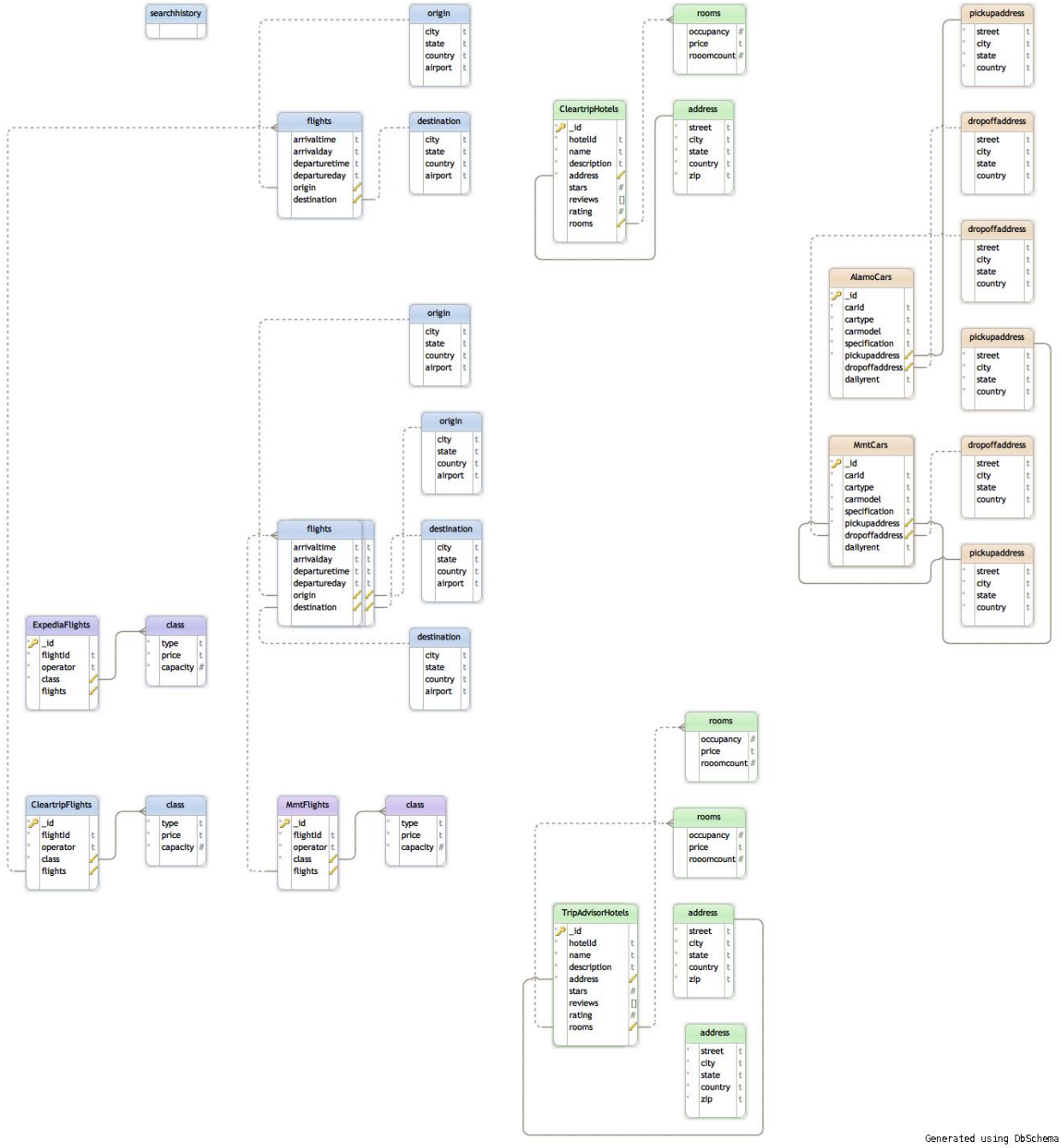
- **Database (MongoDB & MySQL):** The user, user activity, logging and booking information is stored, retrieved and manipulated in the MySQL database. The car, hotel, flights and vendor information is stored, retrieved and manipulated in the MongoDB database.
  - **SQL Caching:** We have used Redis server based SQL caching in the project to cache the queries of the MongoDB database.

- **Connection Pooling:** We have used the built-in connection pooling mechanism to cache the result of the queries in both MongoDB and MySQL. The TTL for caching strategy is set to 60 seconds.

The schema of the MySQL database is as follows:

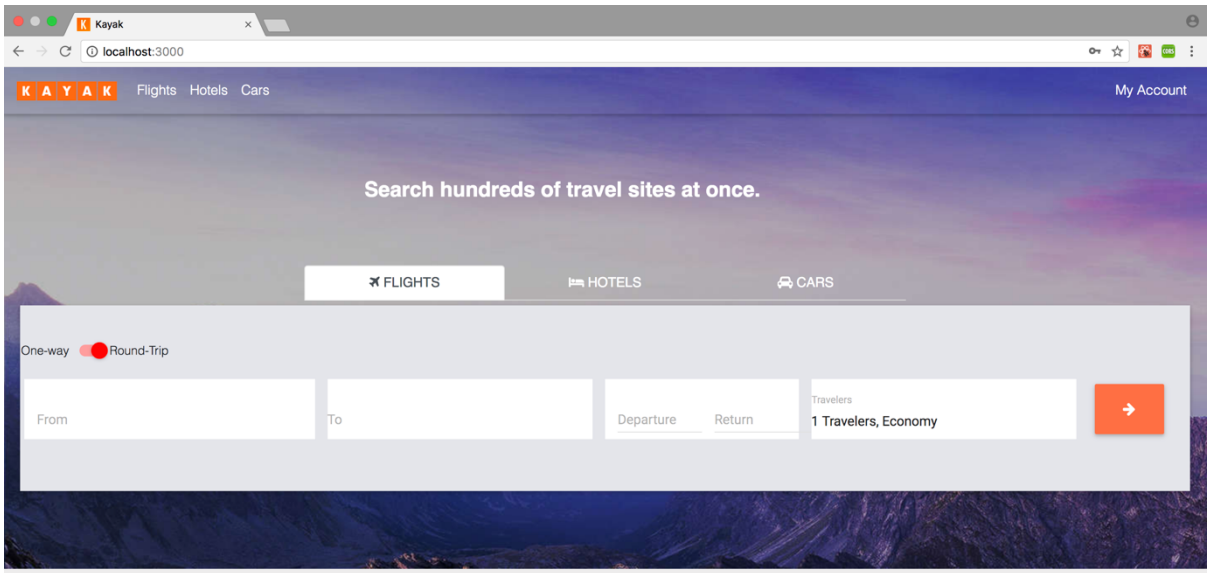


The schema of the MongoDB database is as follows:

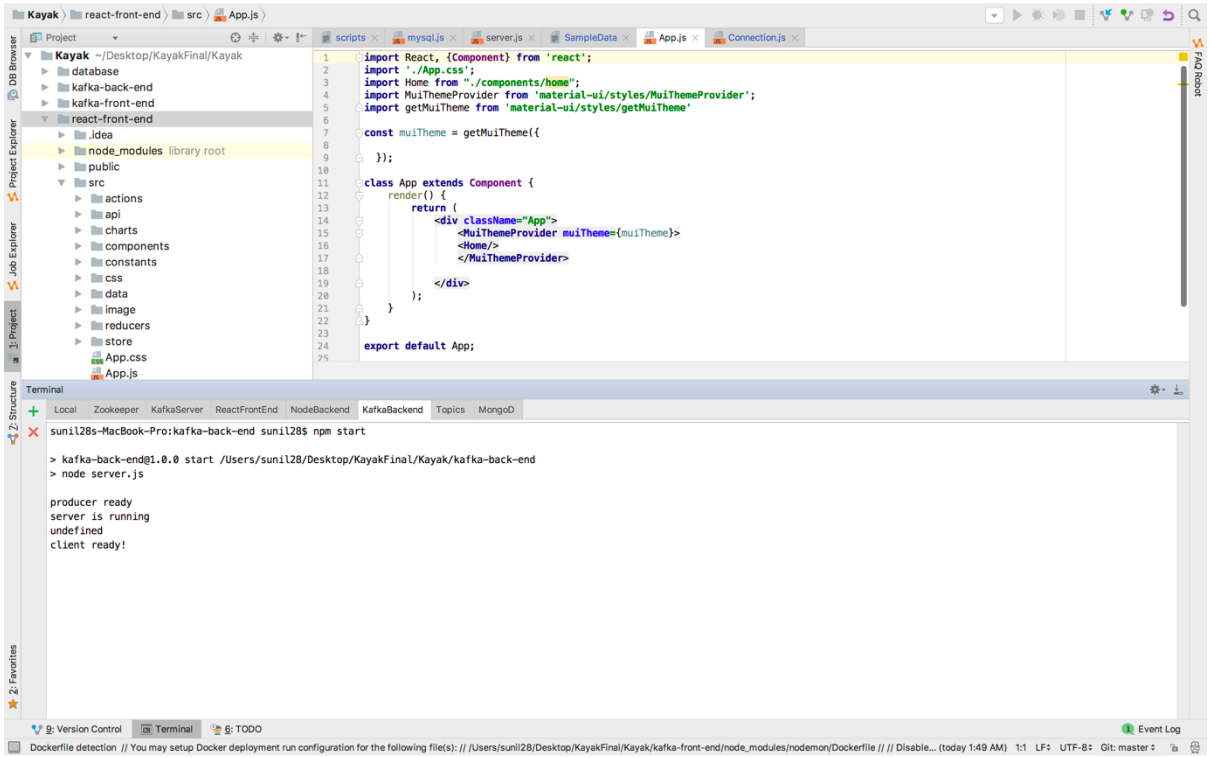


Screenshots:

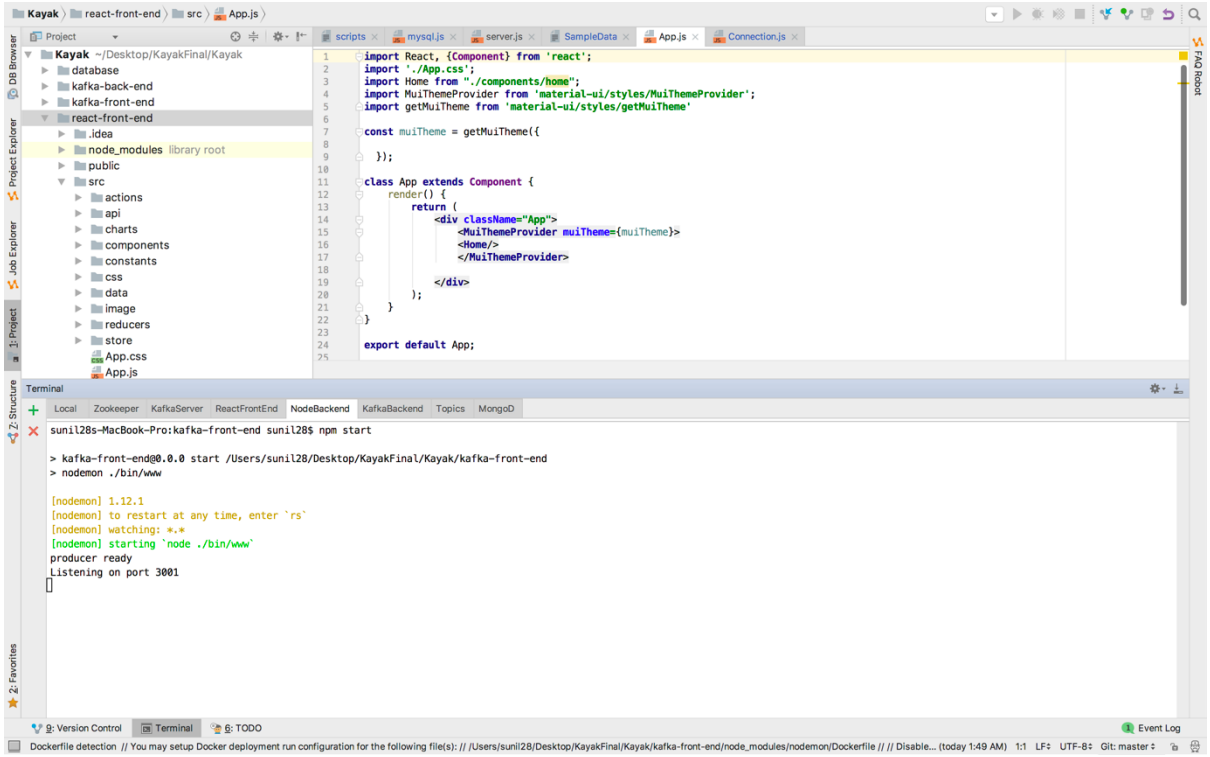
Home Page (UI):



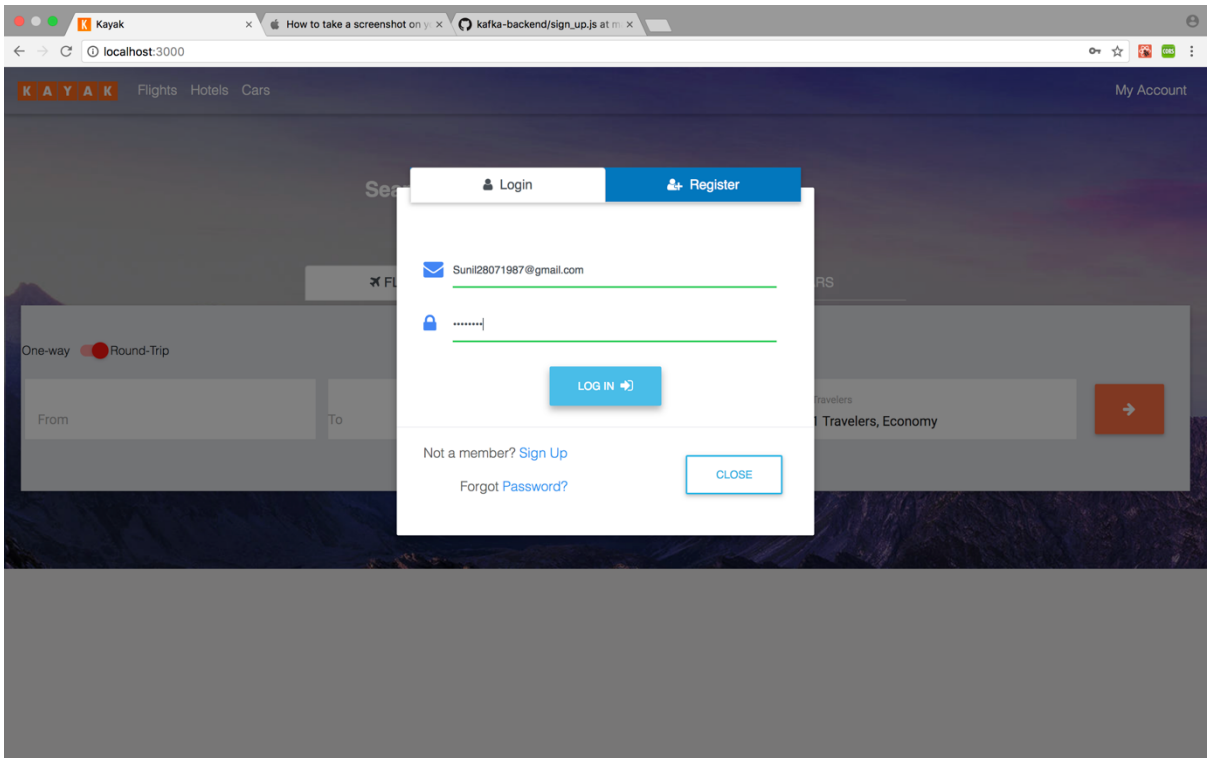
Home Page (Node Backend):



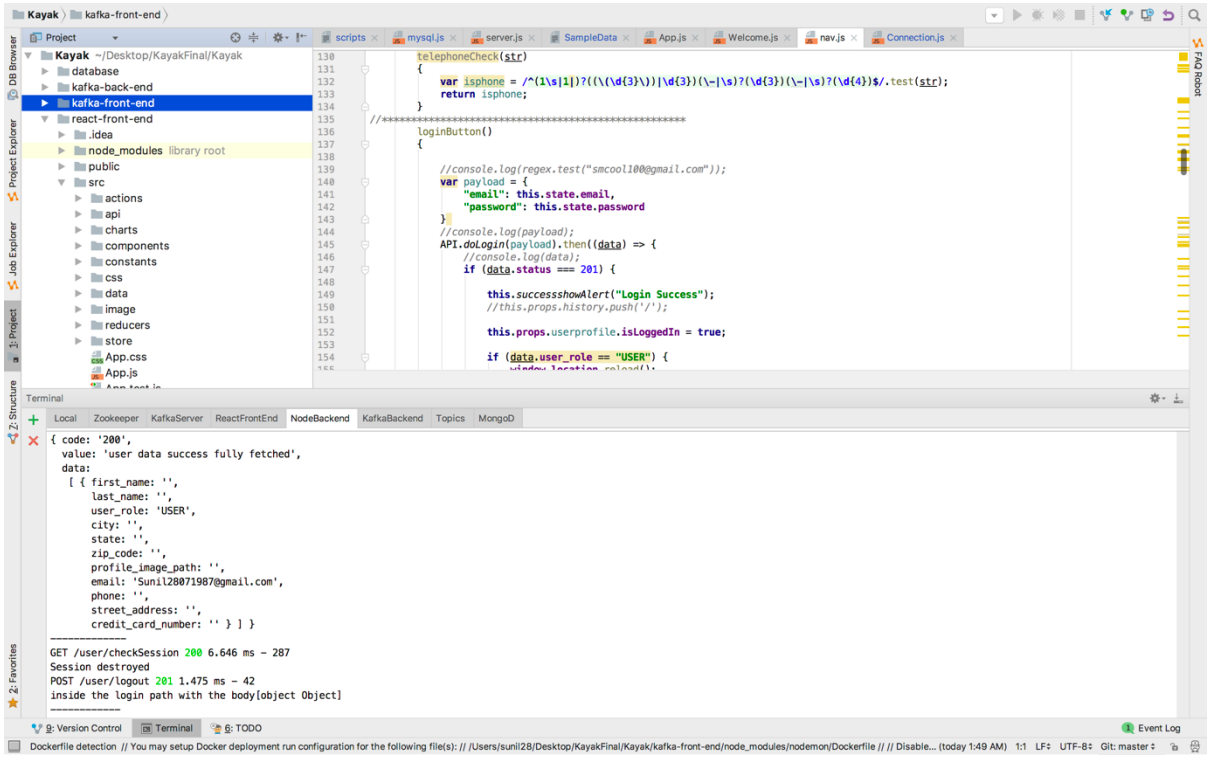
Home Page (Kafka Backend):



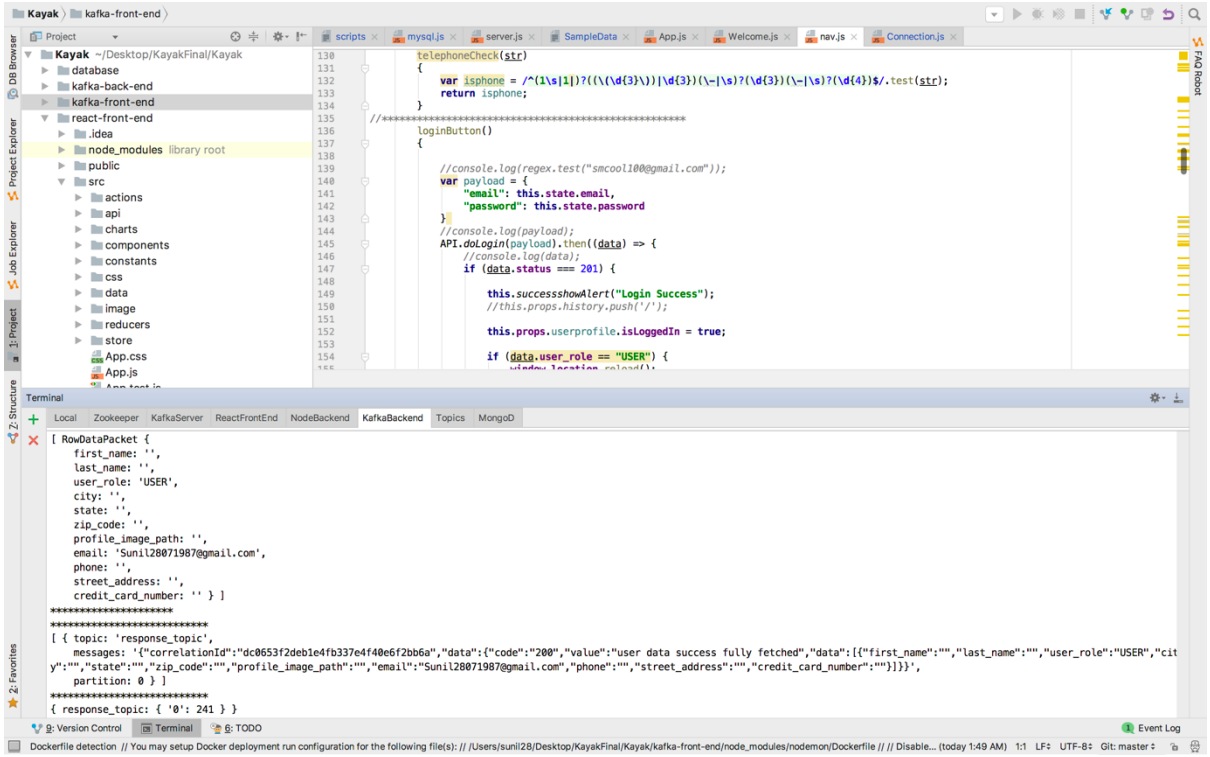
Signup Page (UI):



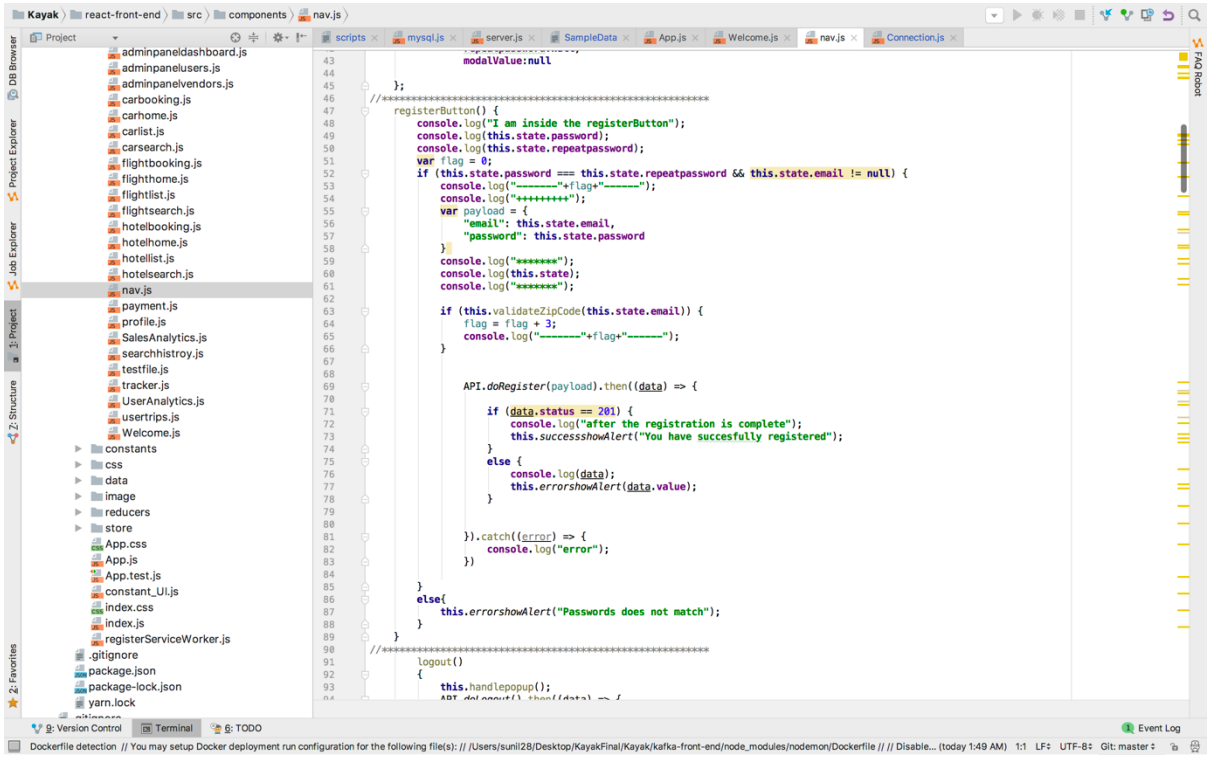
Signup Page (Node Backend):



Signup Page (Kafka Backend):

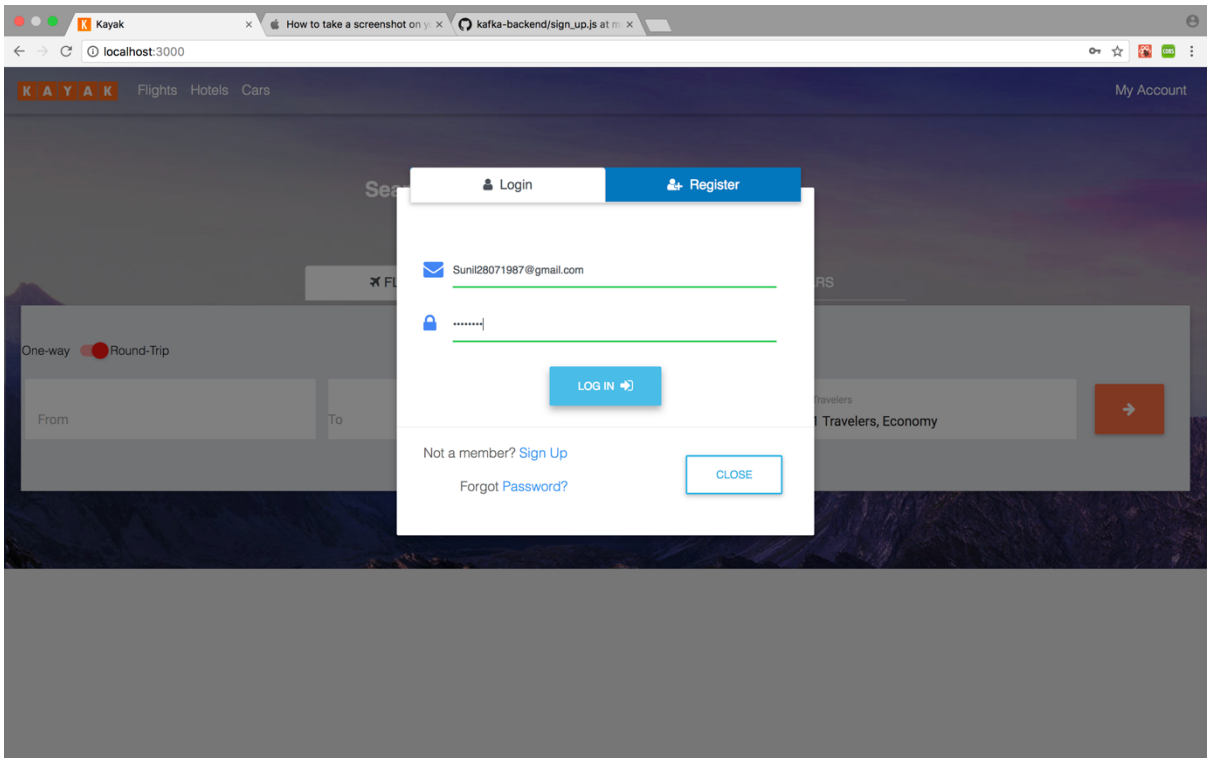


Signup (Business Logic):

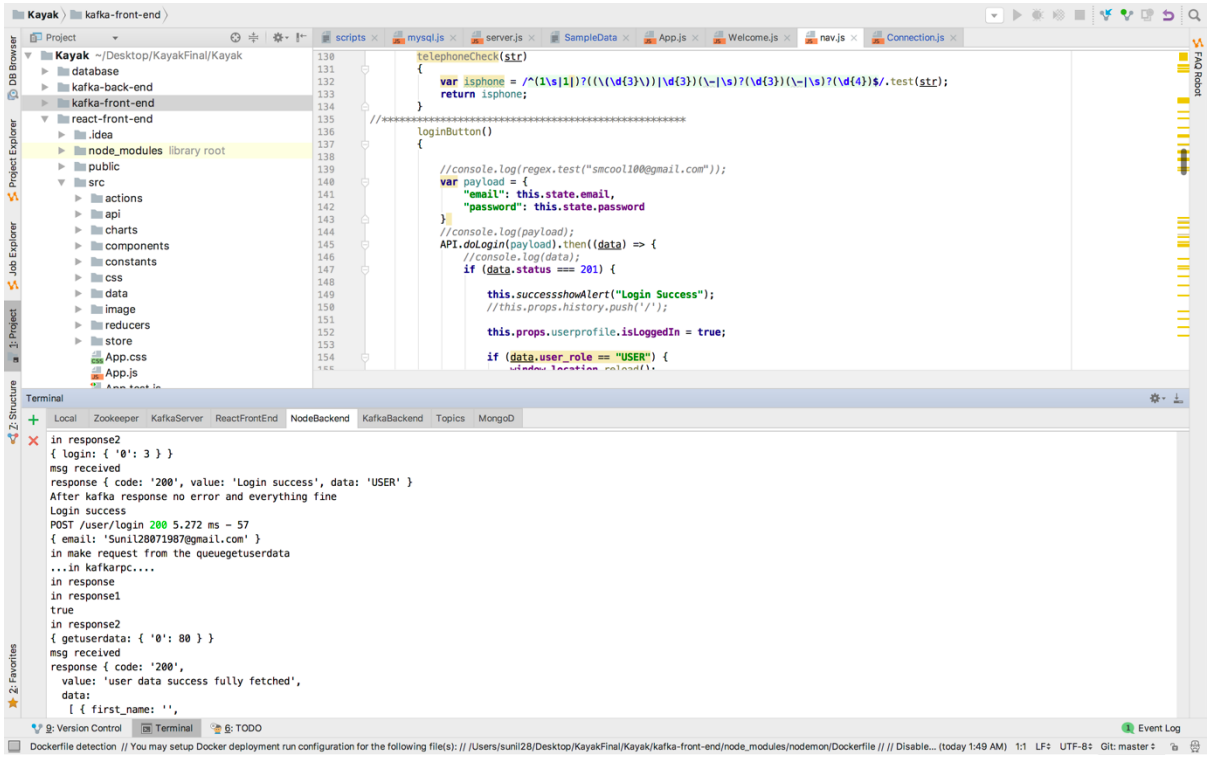


Signup Page (UI):
Signup Page (Node Backend):
Signup Page (Kafka Backend):
Signup (Business Logic):

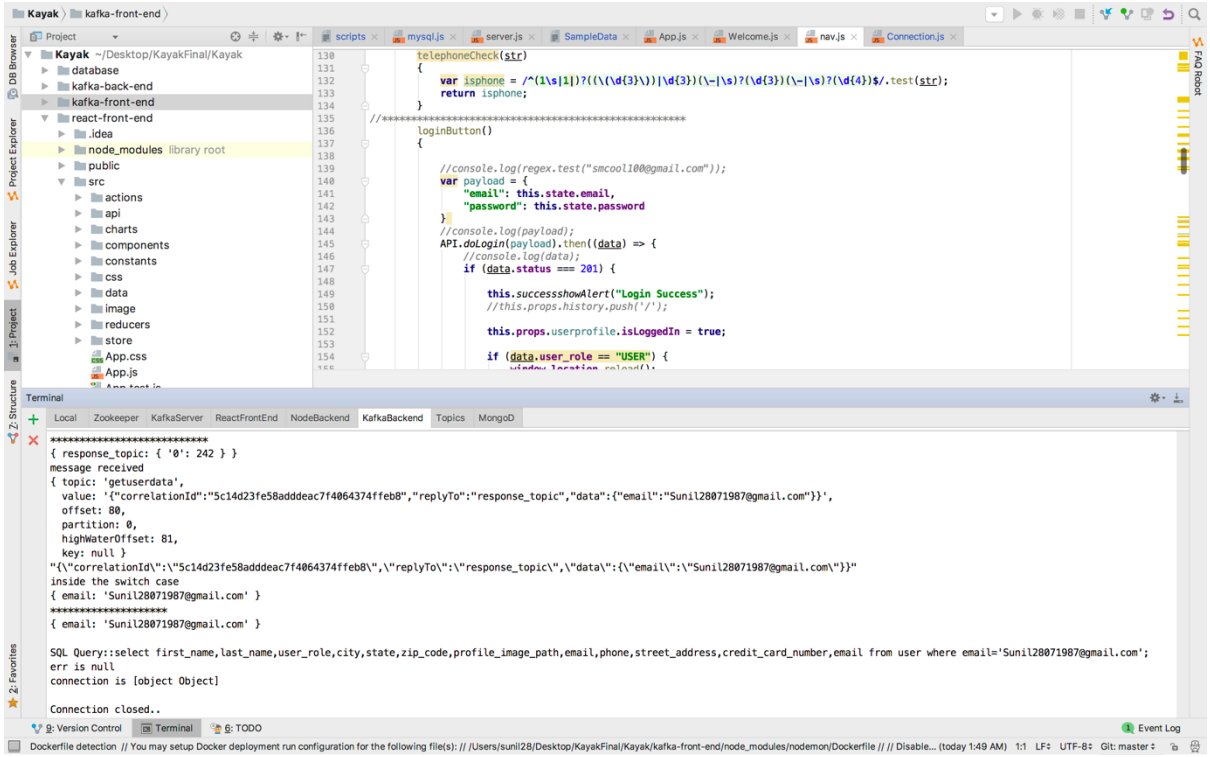
Sign in Page (UI):



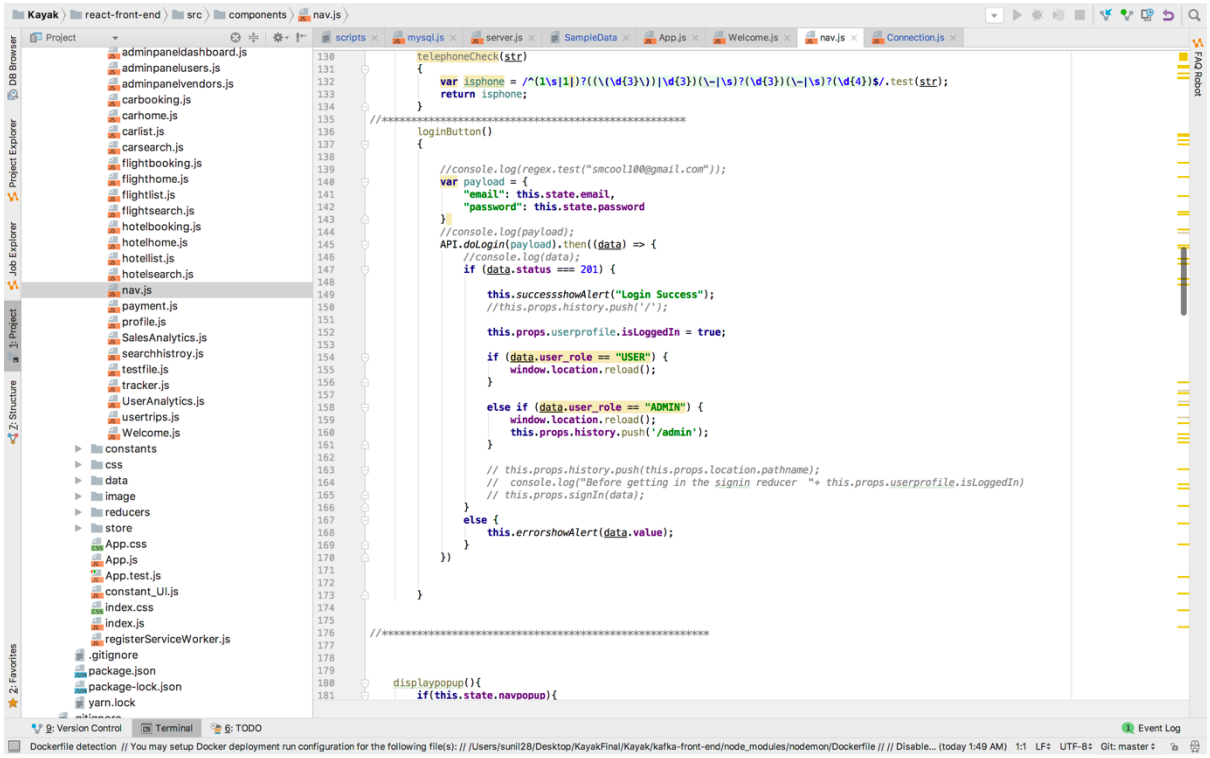
Sign in Page (Node Backend):



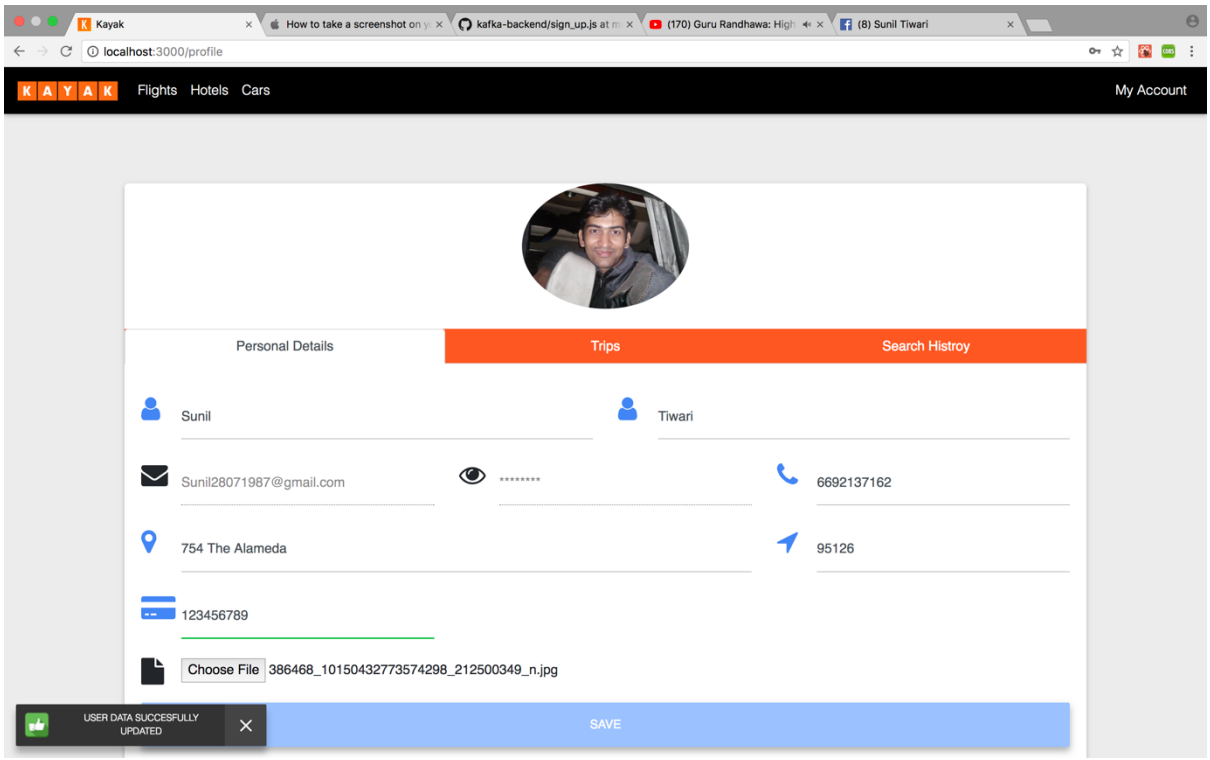
Sign in Page (Kafka Backend):



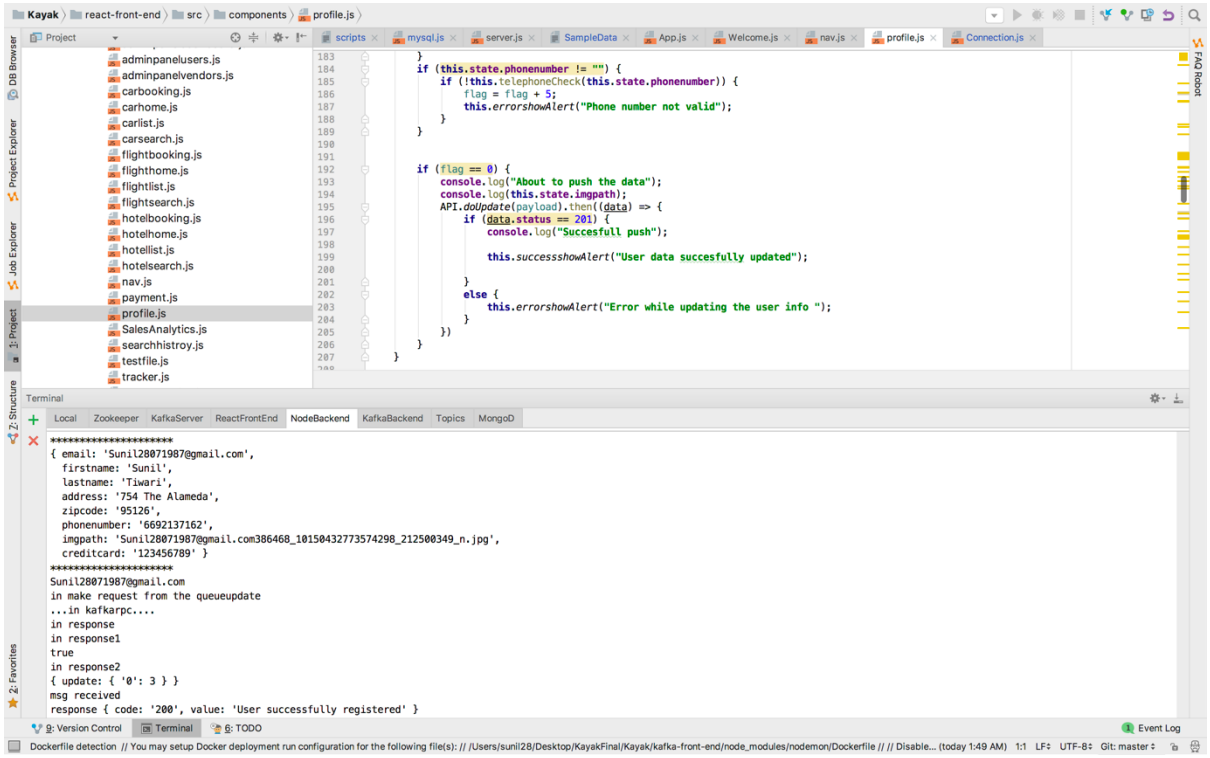
Sign in (Business Logic):



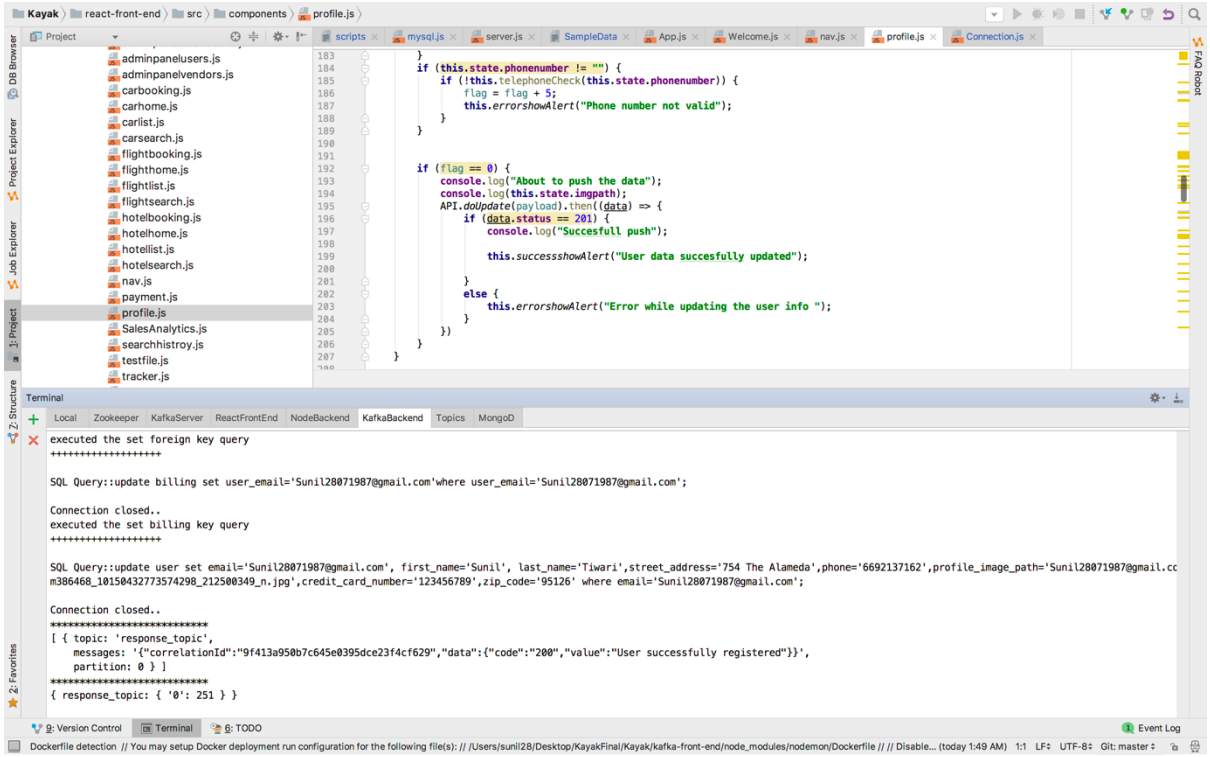
User Profile Page (UI):



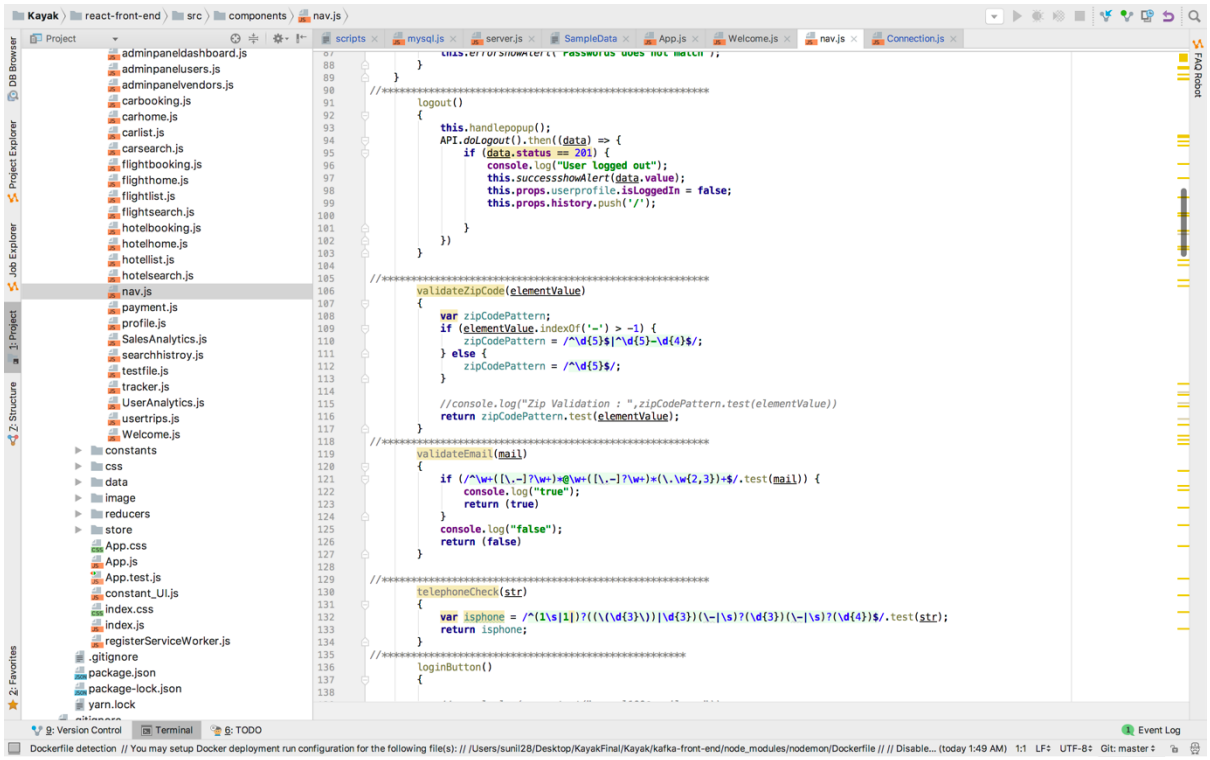
User Profile Page (Node Backend):



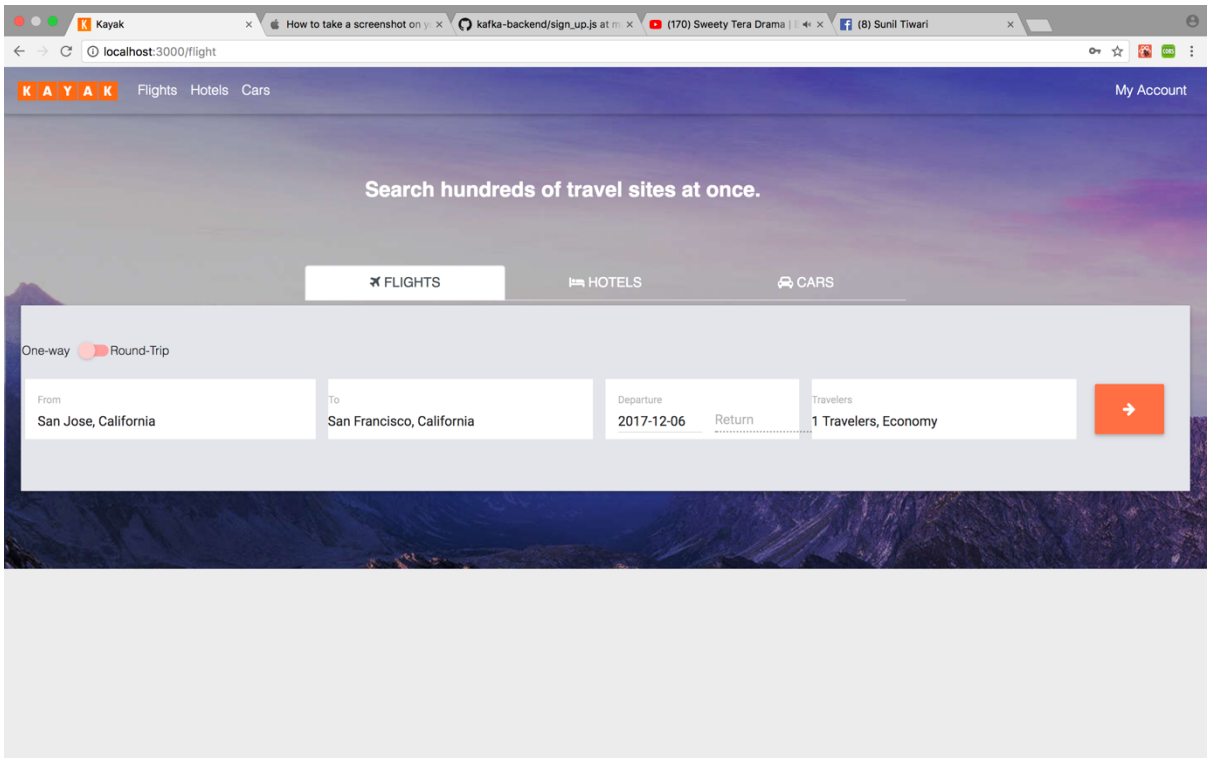
User Profile Page (Kafka Backend):

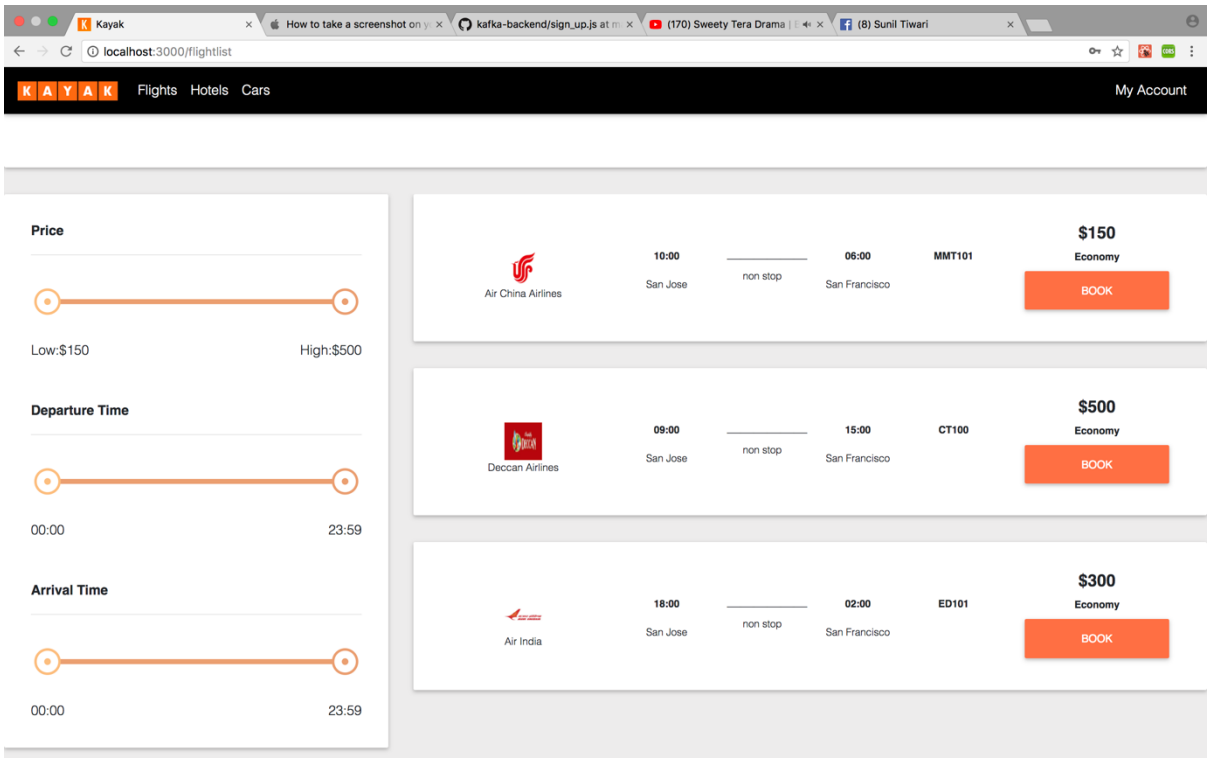


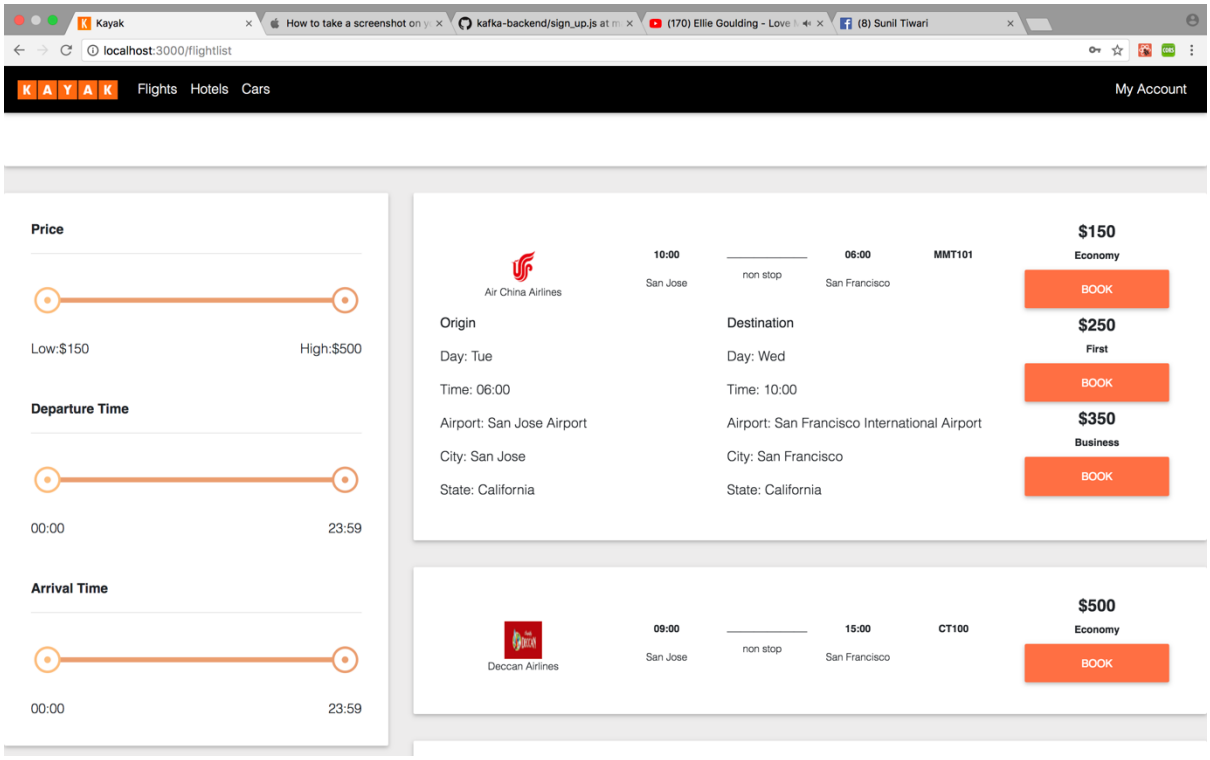
User Profile (Business Logic):

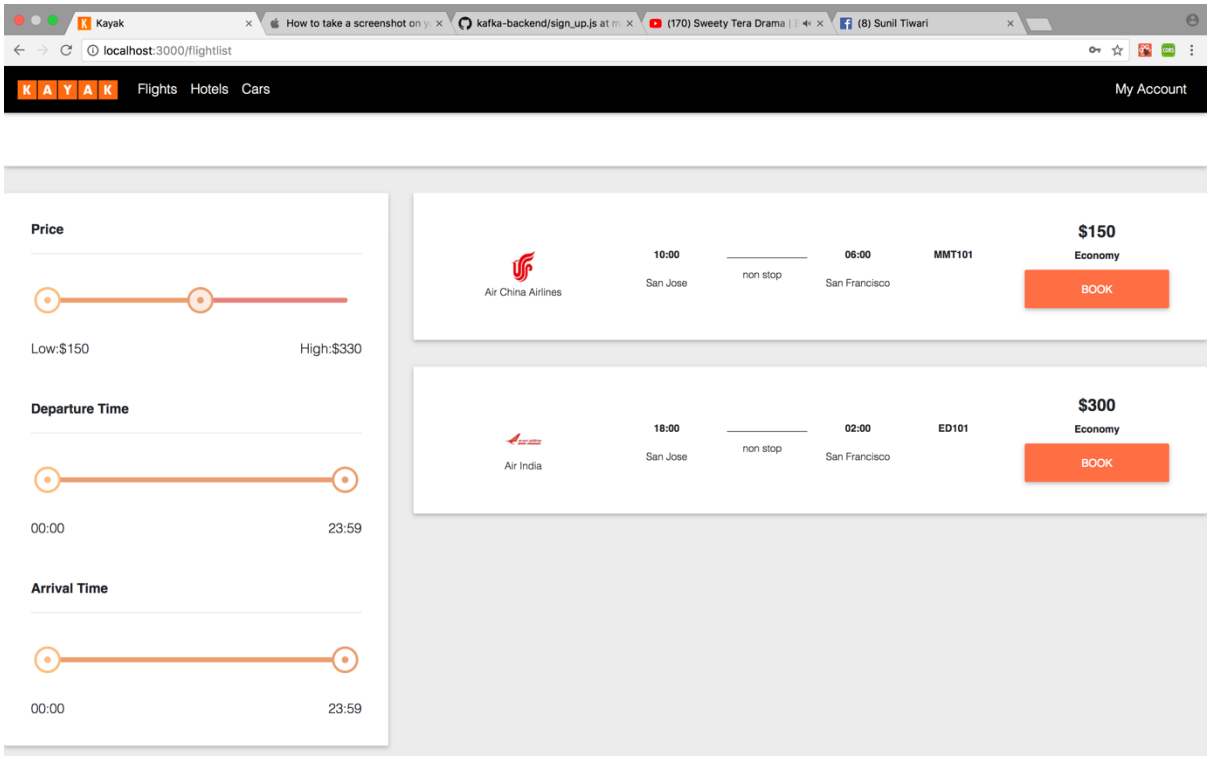


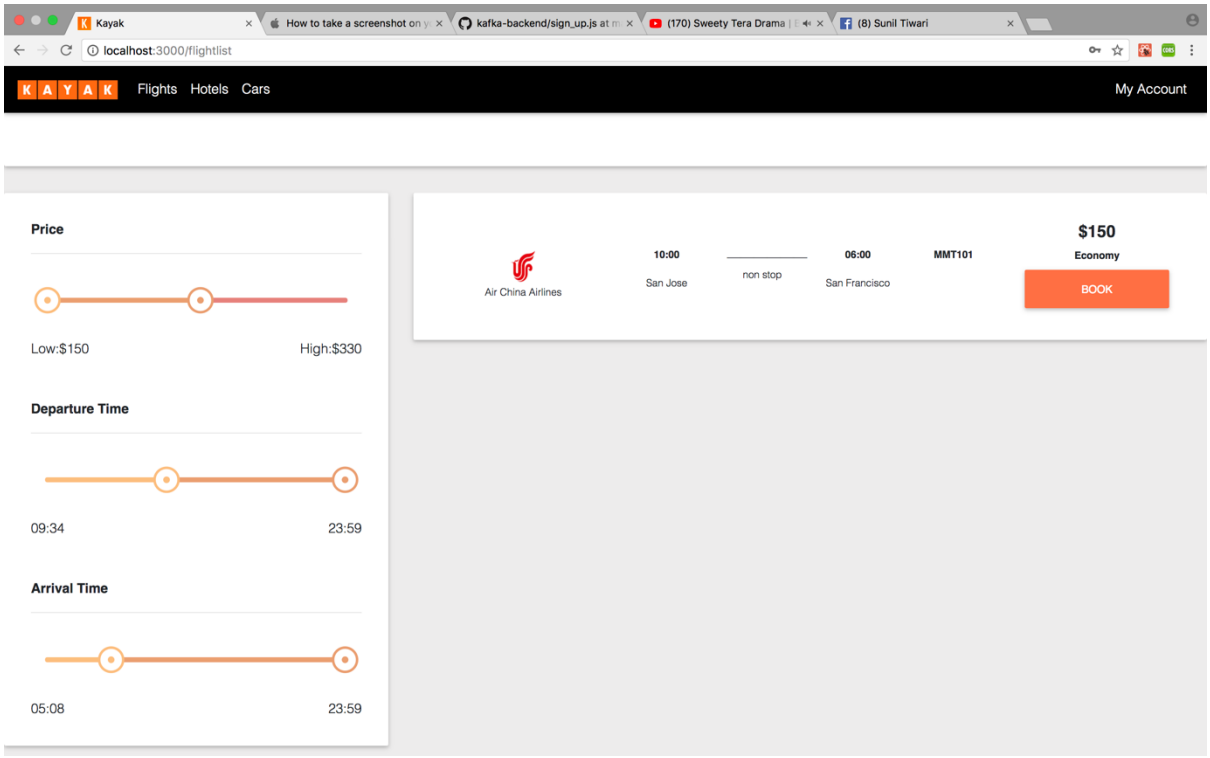
Flight Search Page (UI):



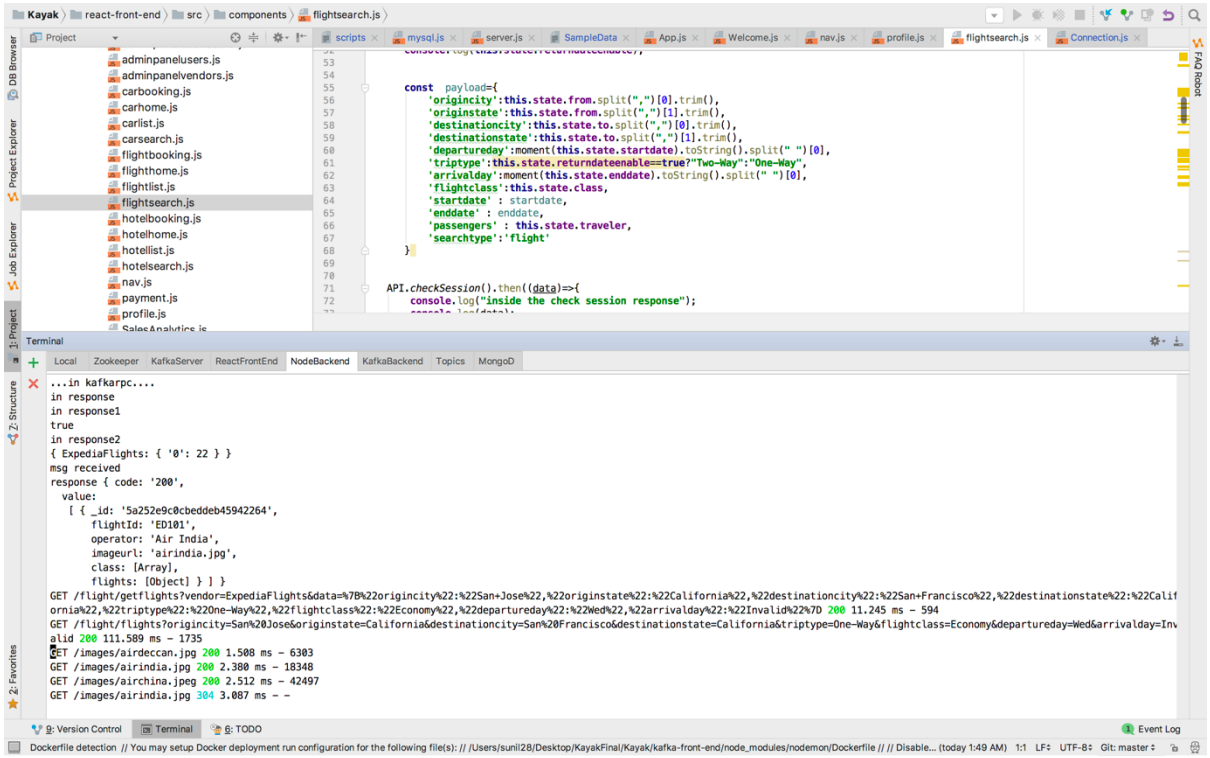



Kayak
How to take a screenshot on y
kafka-backend/sign_up.js at m
(170) Ellie Goulding - Love
(8) Sunil Tiwari
localhost:3000/flightlist
KAYAK
Flights
Hotels
Cars
My Account
Price
Low:$150
High:$500
Departure Time
00:00
23:59
Arrival Time
00:00
23:59
Air China Airlines
10:00
San Jose
non stop
06:00
San Francisco
MMT101
$150
Economy
BOOK
Origin
Day: Tue
Time: 06:00
Airport: San Jose Airport
City: San Jose
State: California
Destination
Day: Wed
Time: 10:00
Airport: San Francisco International Airport
City: San Francisco
State: California
$250
First
BOOK
$350
Business
BOOK
Deccan Airlines
09:00
San Jose
non stop
15:00
San Francisco
CT100
$500
Economy
BOOK

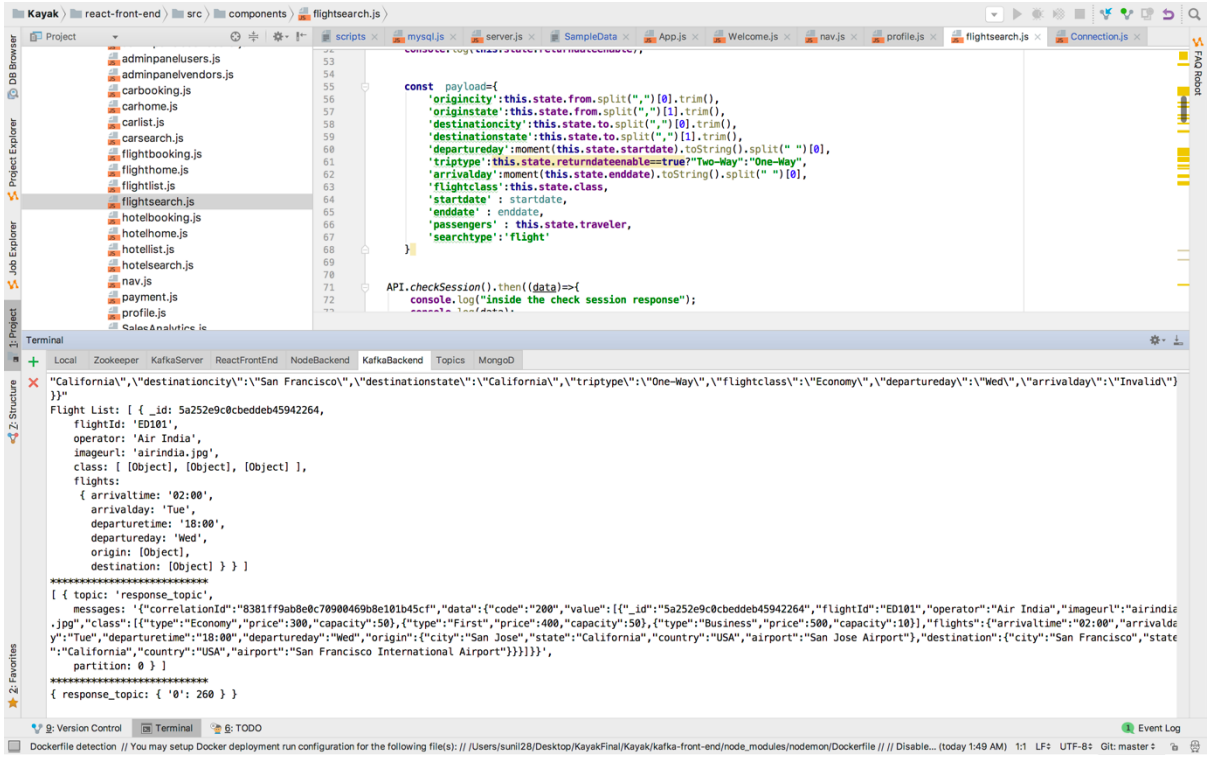

Kayak
How to take a screenshot on y
kafka-backend/sign_up.js at m
(170) Sweety Tera Drama
(8) Sunil Tiwari
localhost:3000/flightlist
KAYAK
Flights
Hotels
Cars
My Account
Price
Low:$150
High:$330
Departure Time
00:00
23:59
Arrival Time
00:00
23:59
Air China Airlines
10:00
San Jose
non stop
06:00
San Francisco
MMT101
$150
Economy
BOOK
Air India
18:00
San Jose
non stop
02:00
San Francisco
ED101
$300
Economy
BOOK



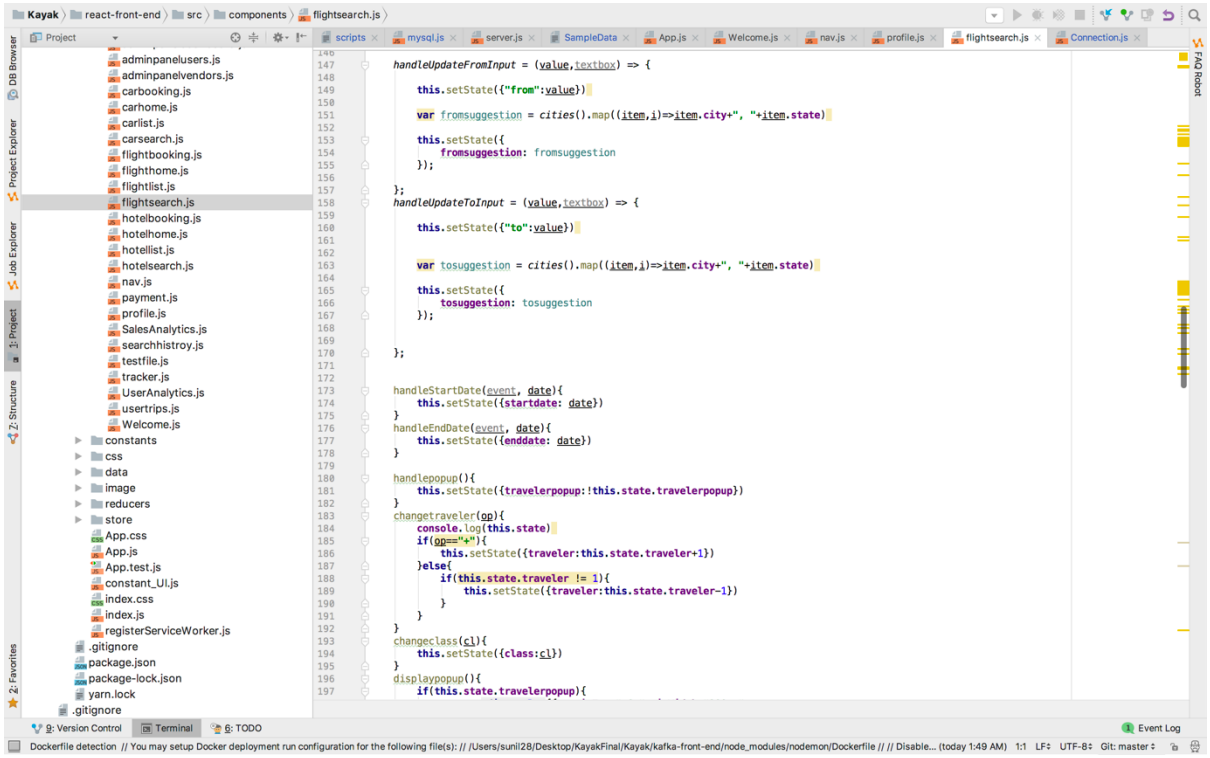
Flight Search (Node Backend):

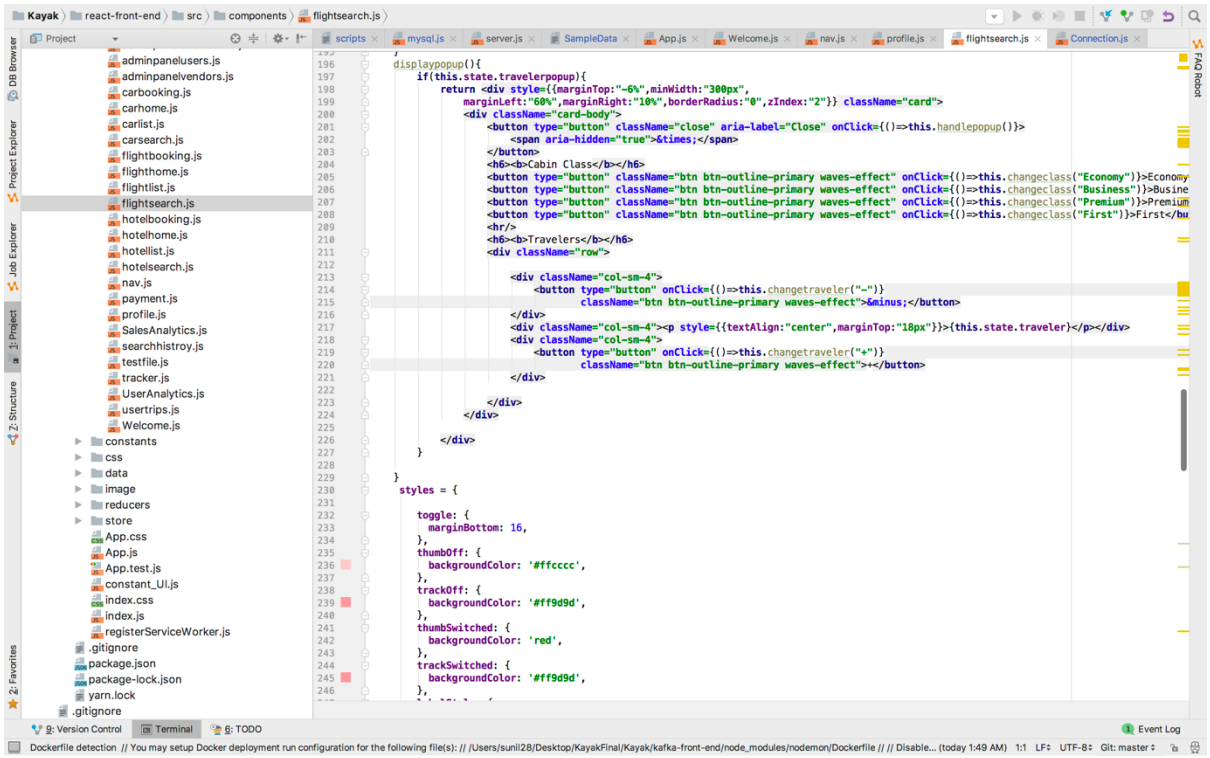


Flight Search (Kafka Backend):



Flight Search (Business Logic):




Kayak › react-front-end › src › components › flightsearch.js
scripts | mysql.js | server.js | SampleData | App.js | Welcome.js | nav.js | profile.js | flightsearch.js | Connection.js
displaypopup(){
if(this.state.travelerpopup){
return <div style={{marginTop:"-6%",minWidth:"300px",
marginLeft:"60%",marginRight:"10%",borderRadius:"0",zIndex:"2"}} className="card">

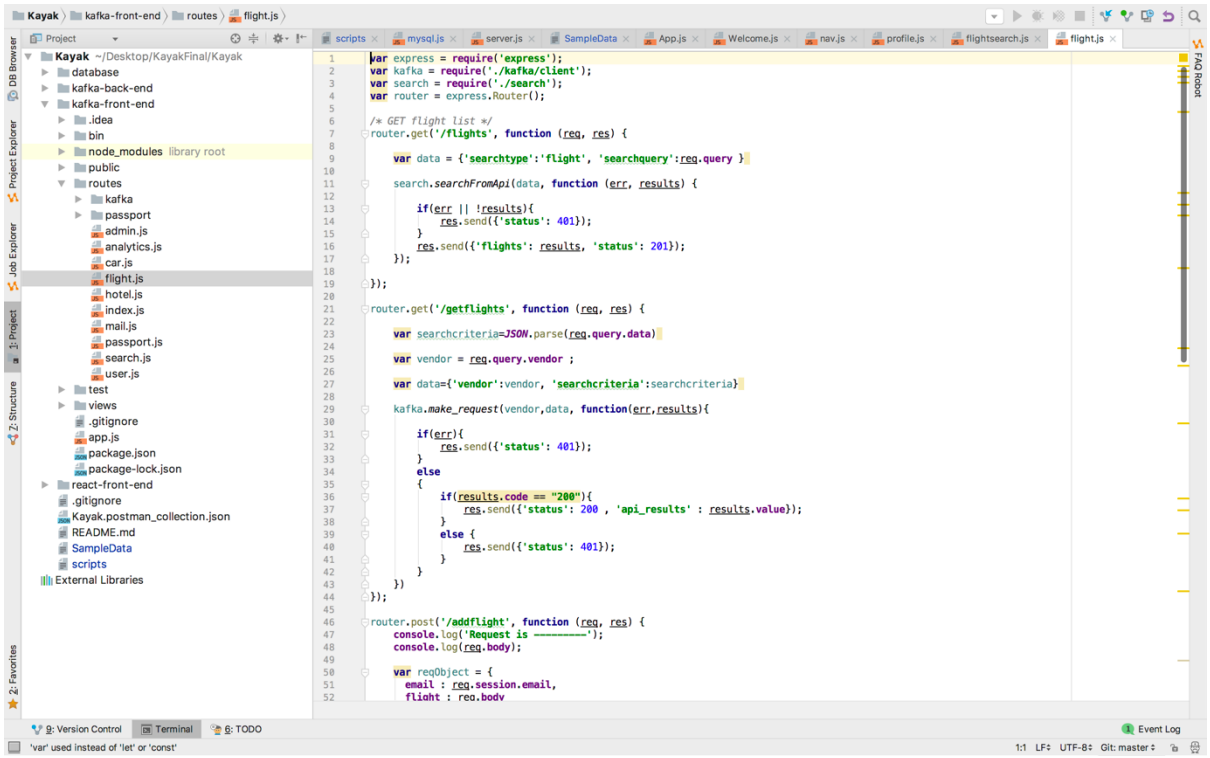
<button type="button" className="close" aria-label="Close" onClick={()=>this.handlepopup()}>
×
</button>
<h6>Cabin Class</h6>
<button type="button" className="btn btn-outline-primary waves-effect" onClick={()=>this.changeclass("Economy")}>Economy
<button type="button" className="btn btn-outline-primary waves-effect" onClick={()=>this.changeclass("Business")}>Busine
<button type="button" className="btn btn-outline-primary waves-effect" onClick={()=>this.changeclass("Premium")}>Premium
<button type="button" className="btn btn-outline-primary waves-effect" onClick={()=>this.changeclass("First")}>First</bu
<hr/>
<h6>Travelers</h6>

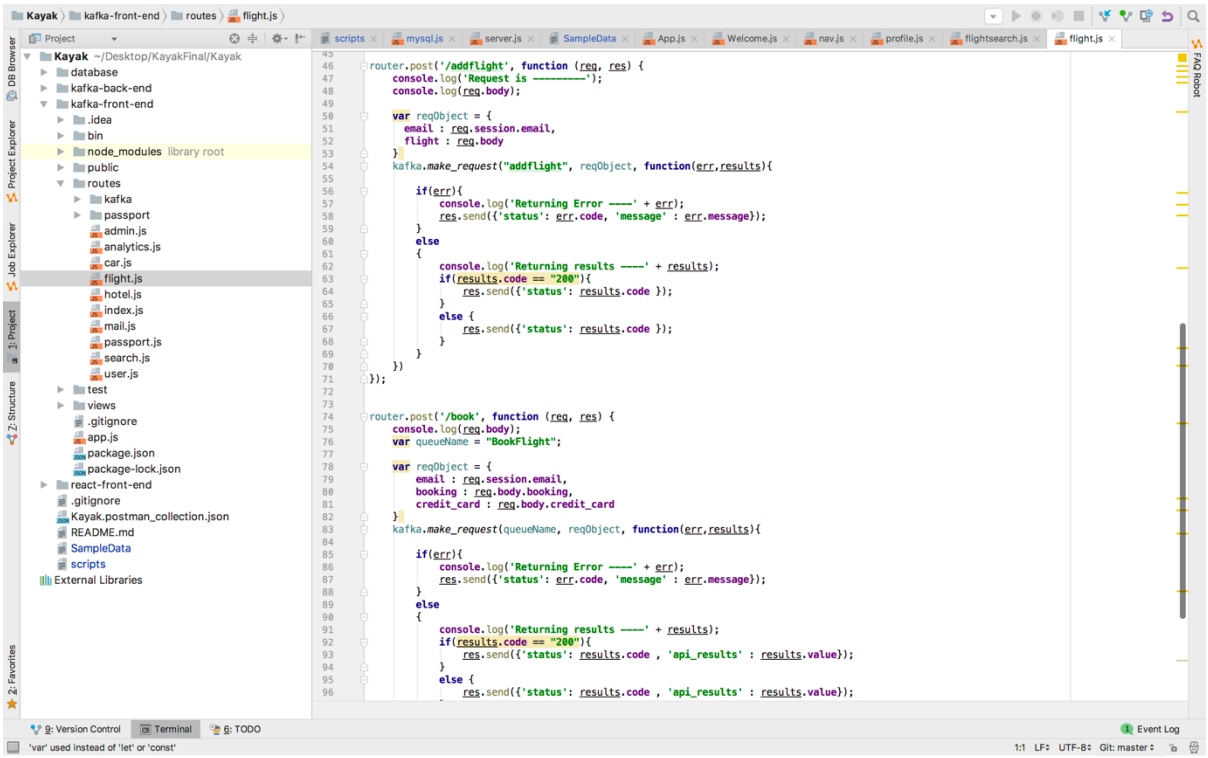
<button type="button" onClick={()=>this.changetraveler("-")}
className="btn btn-outline-primary waves-effect">−</button>

<p style={{textAlign:"center",marginTop:"18px"}}>{this.state.traveler}</p>

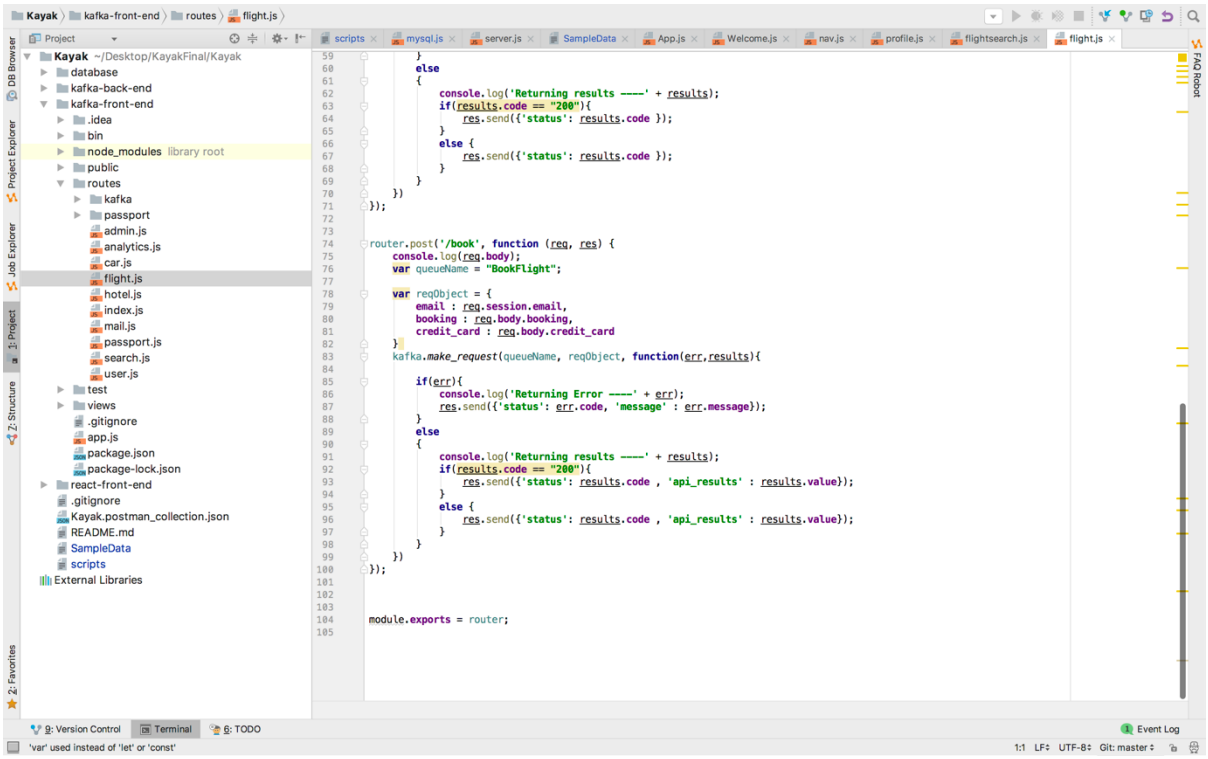
<button type="button" onClick={()=>this.changetraveler("+")}
className="btn btn-outline-primary waves-effect">+</button>

styles = {
toggle: {
marginBottom: 16,
},
thumbOff: {
backgroundColor: '#ffcccc',
},
trackOff: {
backgroundColor: '#ff9d9d',
},
thumbSwitched: {
backgroundColor: 'red',
},
trackSwitched: {
backgroundColor: '#ff9d9d',
},
Project
adminpanelusers.js
adminpanelvendors.js
carbooking.js
carhome.js
carlist.js
carsearch.js
flightbooking.js
flighthome.js
flightlist.js
flightsearch.js
hotelbooking.js
hotelhome.js
hotellist.js
hotelsearch.js
nav.js
payment.js
profile.js
SalesAnalytics.js
searchhistroy.js
testfile.js
tracker.js
UserAnalytics.js
usertrips.js
Welcome.js
constants
css
data
image
reducers
store
App.css
App.js
App.test.js
constant_UI.js
index.css
index.js
registerServiceWorker.js
.gitignore
package.json
package-lock.json
yarn.lock
.gitignore
9: Version Control
Terminal
6: TODO
Event Log
Dockerfile detection // You may setup Docker deployment run configuration for the following file(s): // /Users/sunil28/Desktop/KayakFinal/Kayak/kafka-front-end/node_modules/nodemon/Dockerfile // // Disable... (today 1:49 AM) 1:1 LF UTF-8 Git: master


Kayak › kafka-front-end › routes › flight.js
scripts | mysql.js | server.js | SampleData | App.js | Welcome.js | nav.js | profile.js | flightsearch.js | flight.js
var express = require('express');
var kafka = require('./kafka/client');
var search = require('./search');
var router = express.Router();
/* GET flight list */
router.get('/flights', function (req, res) {
var data = {'searchtype':'flight', 'searchquery':req.query }
search.searchFromApi(data, function (err, results) {
if(err || !results){
res.send({'status': 401});
}
res.send({'flights': results, 'status': 201});
});
});
router.get('/getflights', function (req, res) {
var searchcriteria=JSON.parse(req.query.data)
var vendor = req.query.vendor ;
var data={'vendor':vendor, 'searchcriteria':searchcriteria}
kafka.make_request(vendor,data, function(err,results){
if(err){
res.send({'status': 401});
}
else
{
if(results.code == "200"){
res.send({'status': 200 , 'api_results' : results.value});
}
else {
res.send({'status': 401});
}
}
})
});
router.post('/addflight', function (req, res) {
console.log('Request is ---------');
console.log(req.body);
var reqObject = {
email : req.session.email,
flight : req.body
Project
Kayak ~/Desktop/KayakFinal/Kayak
database
kafka-back-end
kafka-front-end
.idea
bin
node_modules library root
public
routes
kafka
passport
admin.js
analytics.js
car.js
flight.js
hotel.js
index.js
mail.js
passport.js
search.js
user.js
test
views
.gitignore
app.js
package.json
package-lock.json
react-front-end
.gitignore
Kayak.postman_collection.json
README.md
SampleData
scripts
External Libraries
9: Version Control
Terminal
6: TODO
Event Log
'var' used instead of 'let' or 'const'
1:1 LF UTF-8 Git: master

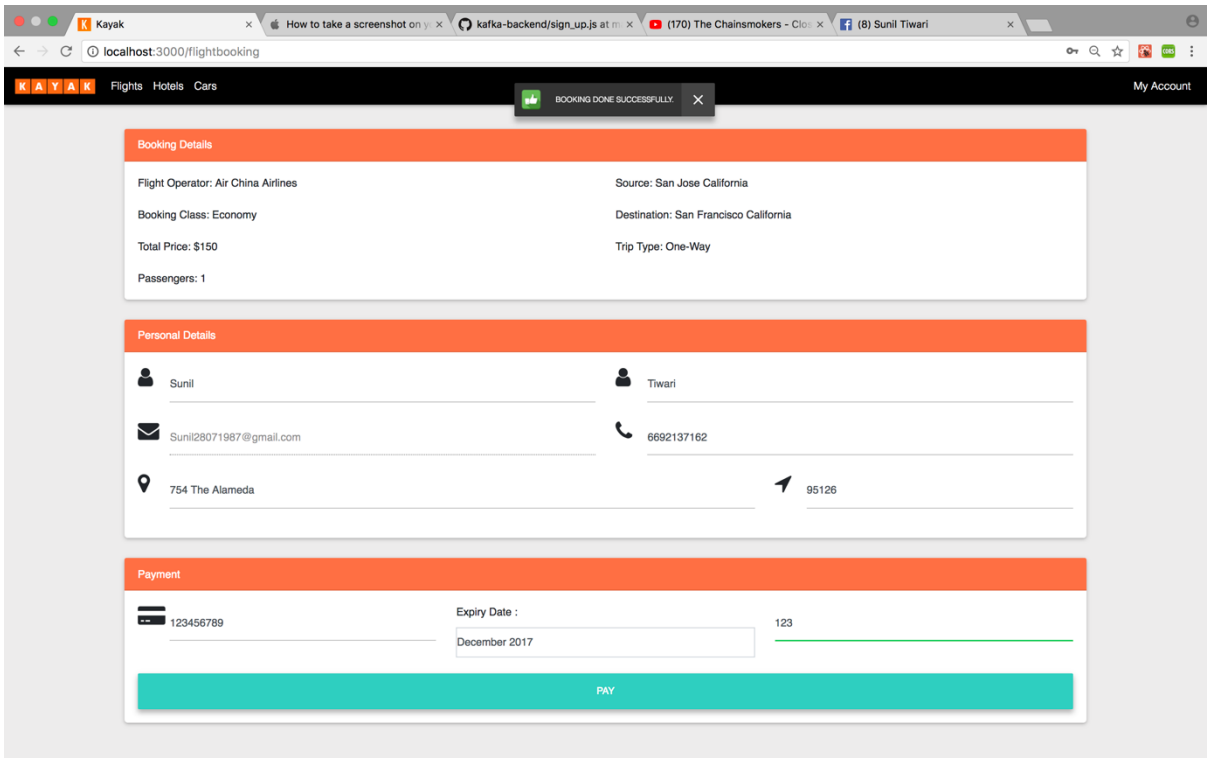


```
router.post('/addflight', function (req, res) {
    console.log('Request is ---------');
    console.log(req.body);

    var reqObject = {
      email : req.session.email,
      flight : req.body
    }
    kafka.make_request("addflight", reqObject, function(err,results){

        if(err){
            console.log('Returning Error ----' + err);
            res.send({'status': err.code, 'message' : err.message});
        }
        else
        {
            console.log('Returning results ----' + results);
            if(results.code == "200"){
                res.send({'status': results.code });
            }
            else {
                res.send({'status': results.code });
            }
        }
    })
});


router.post('/book', function (req, res) {
    console.log(req.body);
    var queueName = "BookFlight";

    var reqObject = {
        email : req.session.email,
        booking : req.body.booking,
        credit_card : req.body.credit_card
    }
    kafka.make_request(queueName, reqObject, function(err,results){

        if(err){
            console.log('Returning Error ----' + err);
            res.send({'status': err.code, 'message' : err.message});
        }
        else
        {
            console.log('Returning results ----' + results);
            if(results.code == "200"){
                res.send({'status': results.code , 'api_results' : results.value});
            }
            else {
                res.send({'status': results.code , 'api_results' : results.value});
```

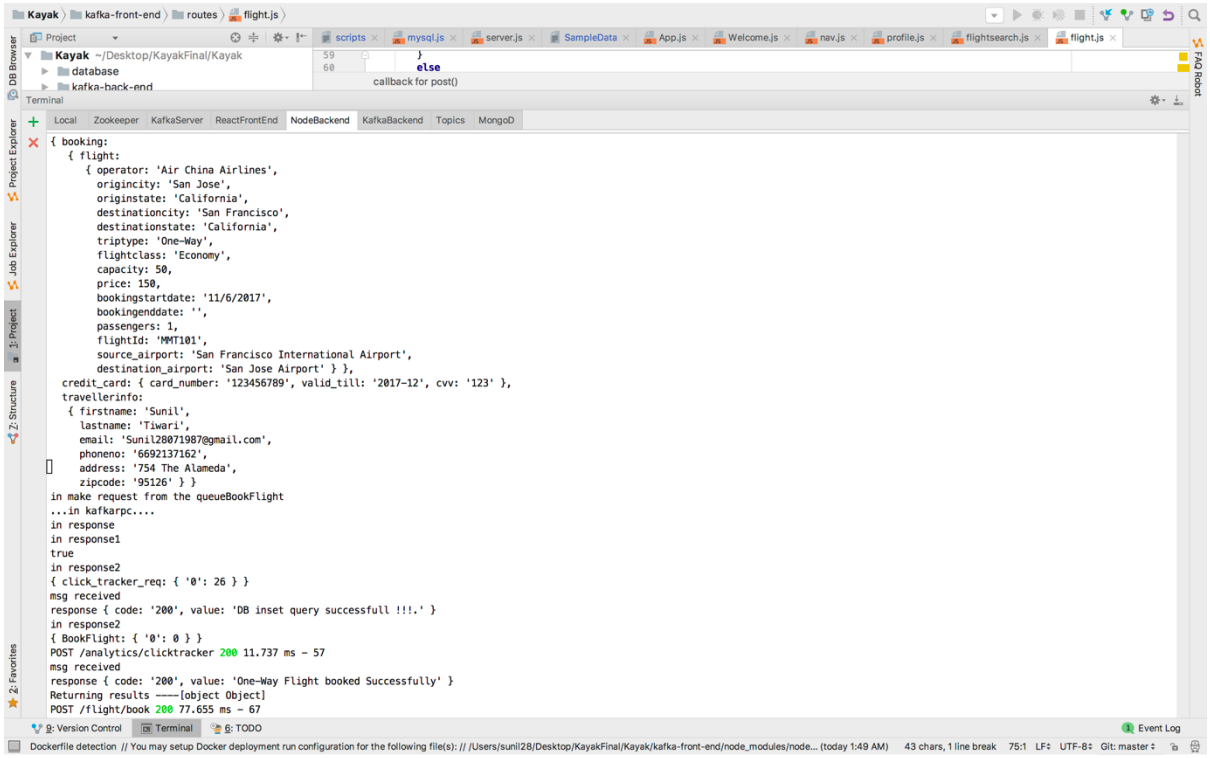


```
        }
        else
        {
            console.log('Returning results ----' + results);
            if(results.code == "200"){
                res.send({'status': results.code });
            }
            else {
                res.send({'status': results.code });
            }
        }
    })
});


router.post('/book', function (req, res) {
    console.log(req.body);
    var queueName = "BookFlight";

    var reqObject = {
        email : req.session.email,
        booking : req.body.booking,
        credit_card : req.body.credit_card
    }
    kafka.make_request(queueName, reqObject, function(err,results){

        if(err){
            console.log('Returning Error ----' + err);
            res.send({'status': err.code, 'message' : err.message});
        }
        else
        {
            console.log('Returning results ----' + results);
            if(results.code == "200"){
                res.send({'status': results.code , 'api_results' : results.value});
            }
            else {
                res.send({'status': results.code , 'api_results' : results.value});
            }
        }
    })
});


module.exports = router;
```

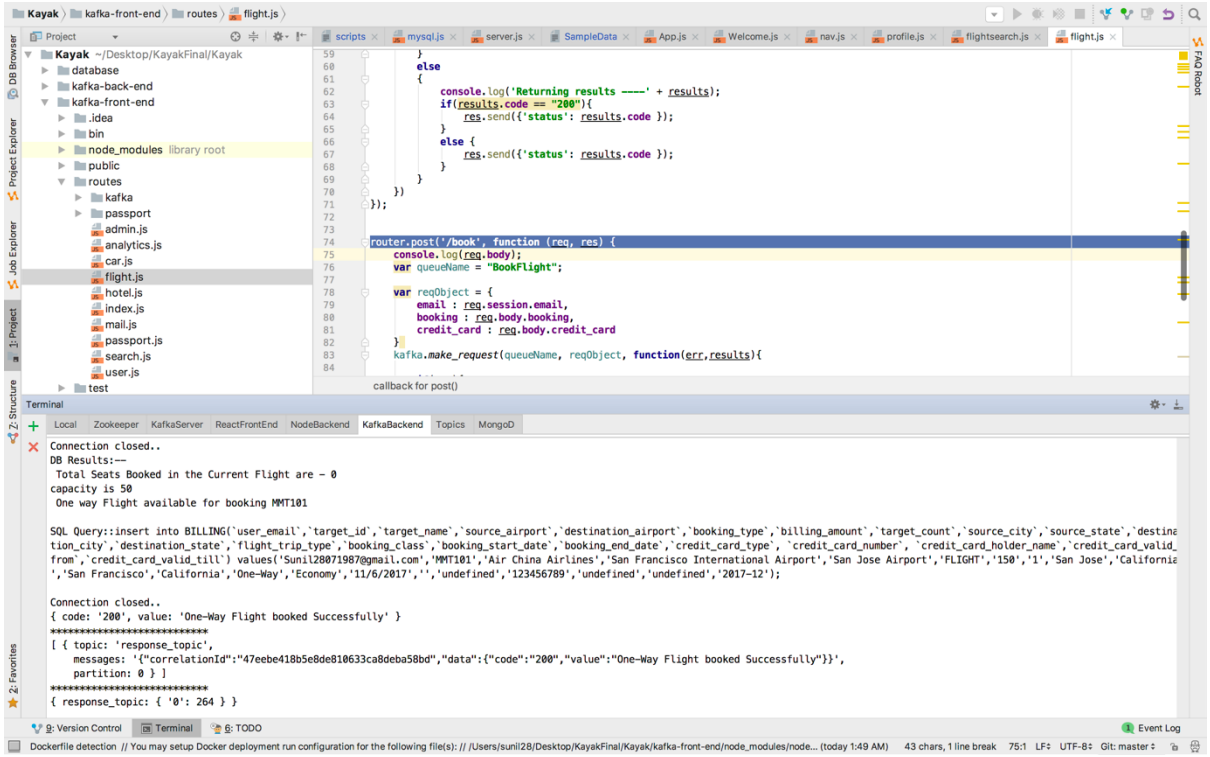
Flight Booking Page (UI):



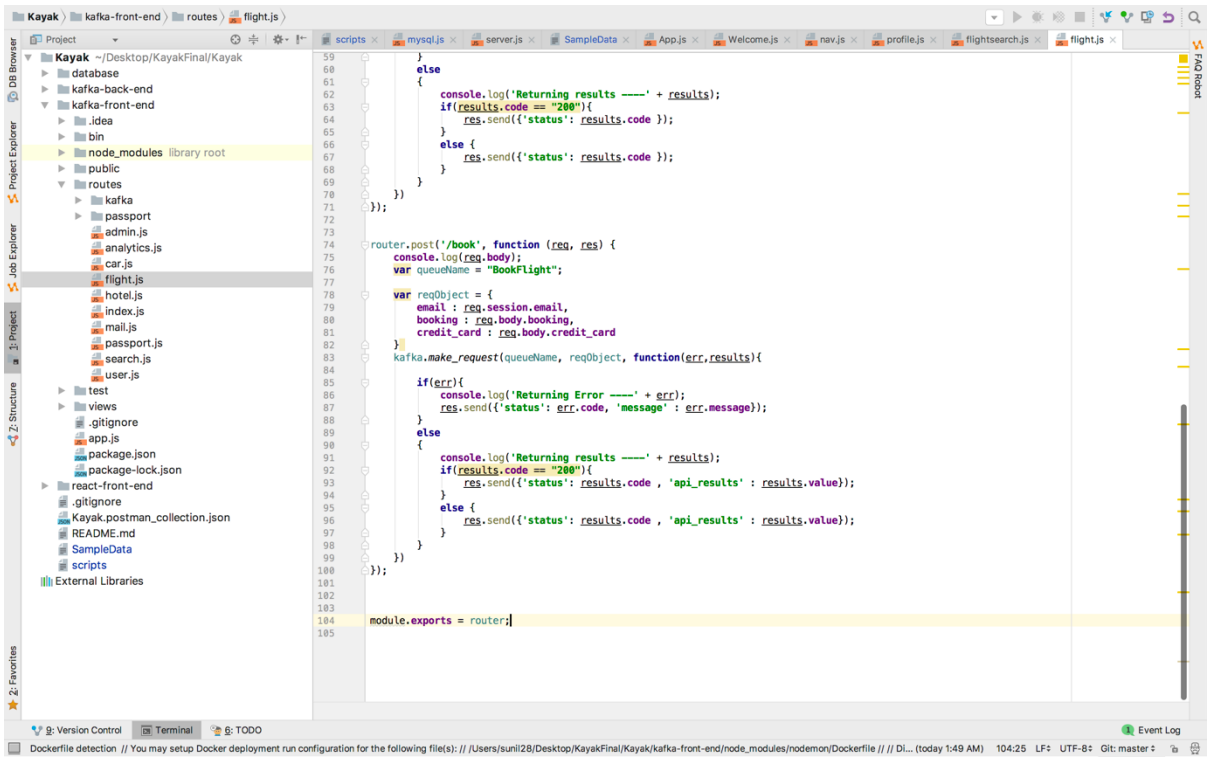
Flight Booking Page (Node Backend):



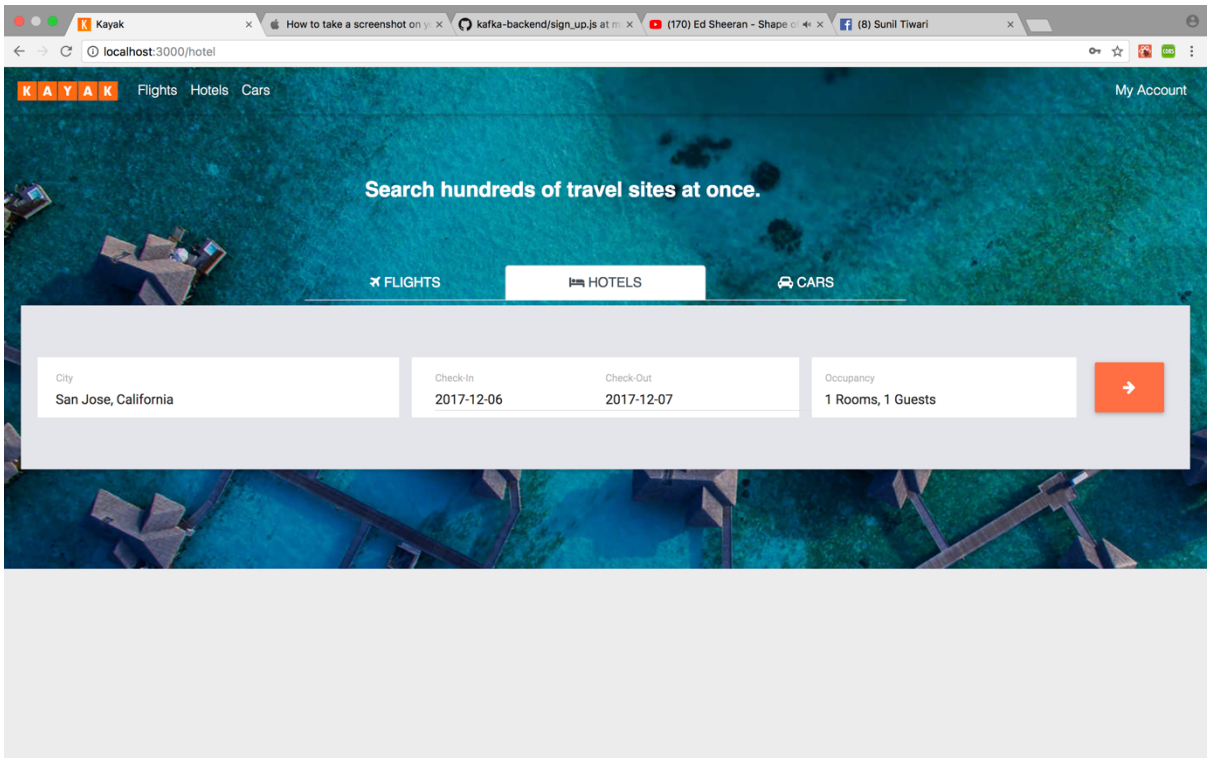
Flight Booking Page (Kafka Backend):

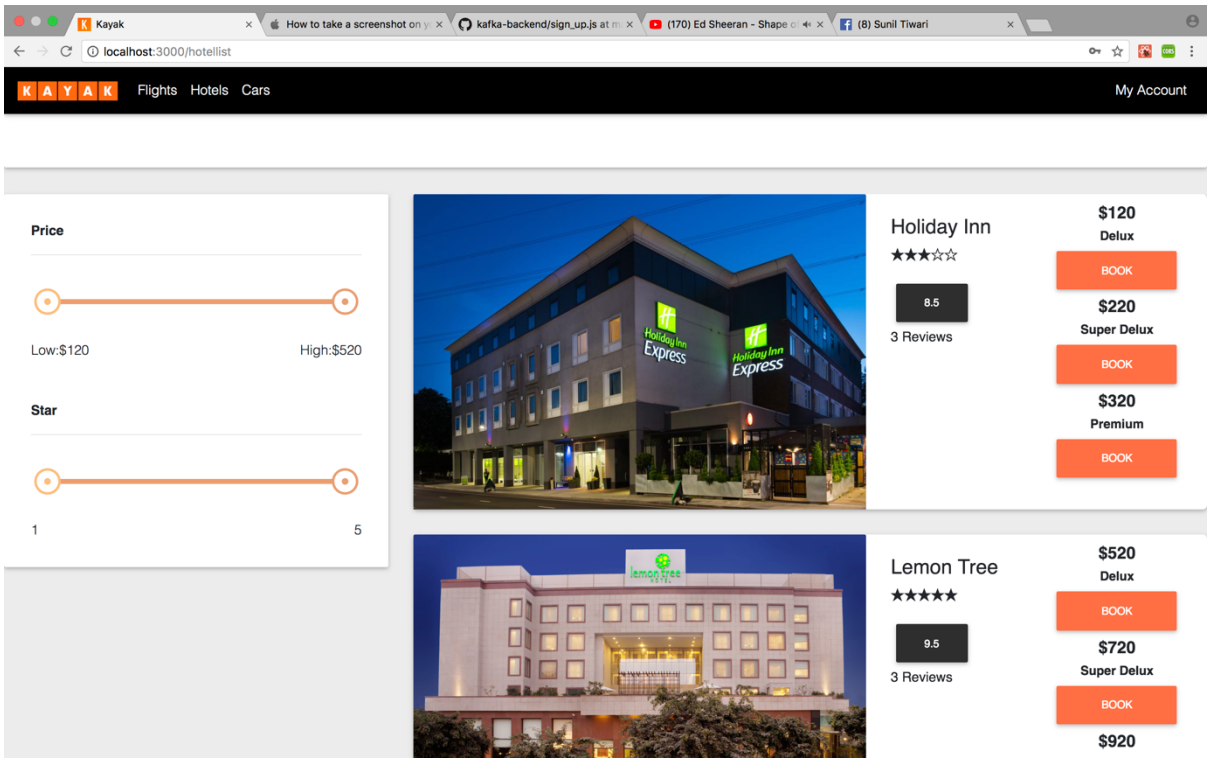


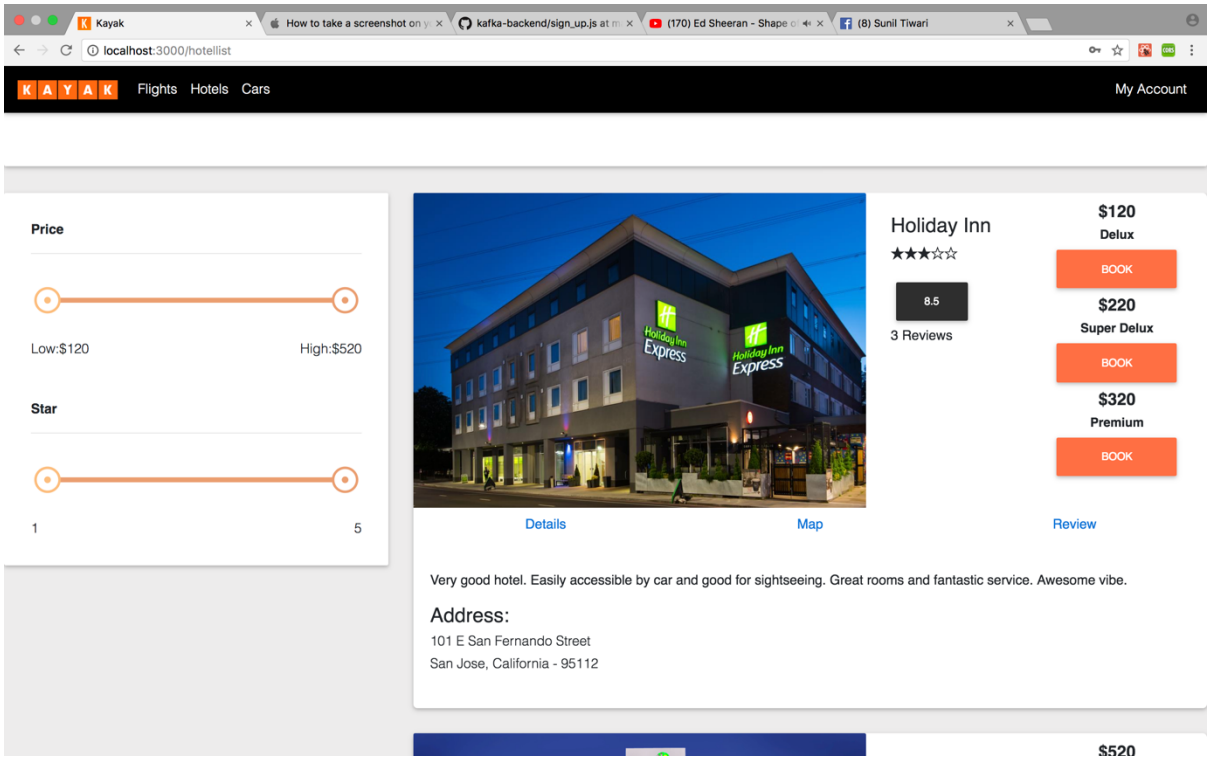
Flight Booking (Business Logic):

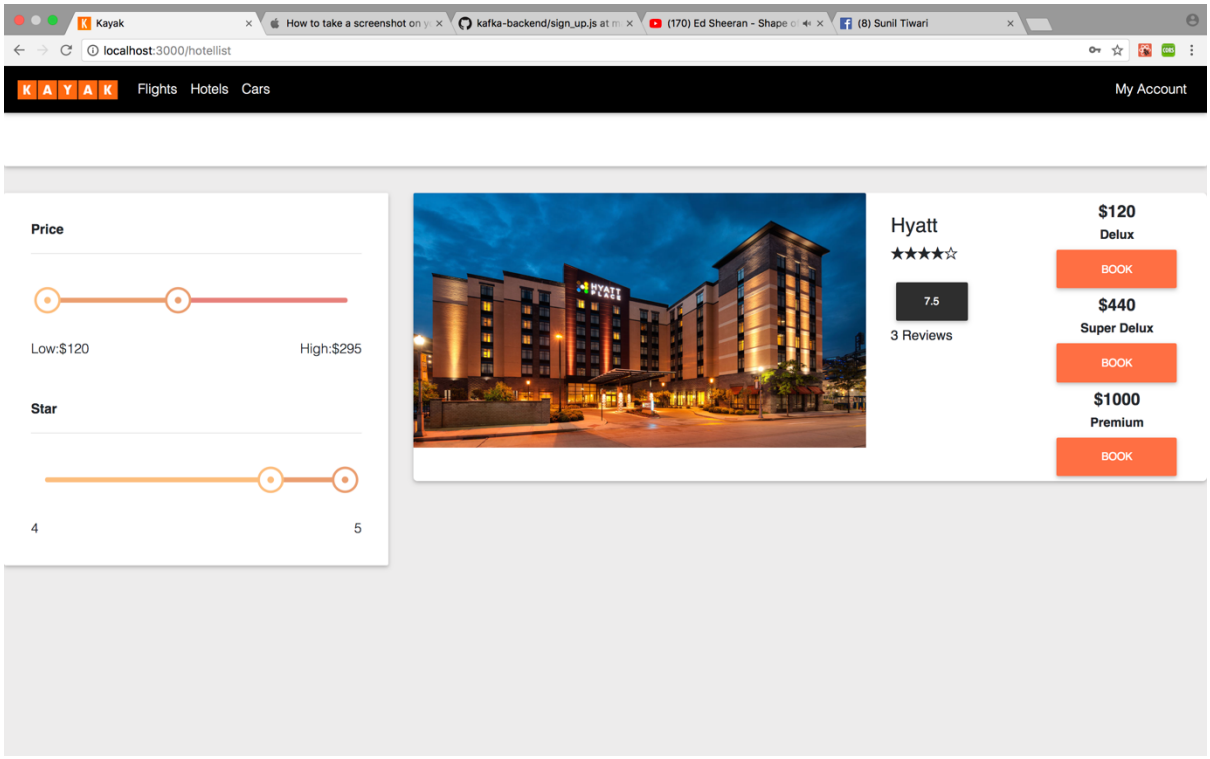


Hotel Search Page (UI):

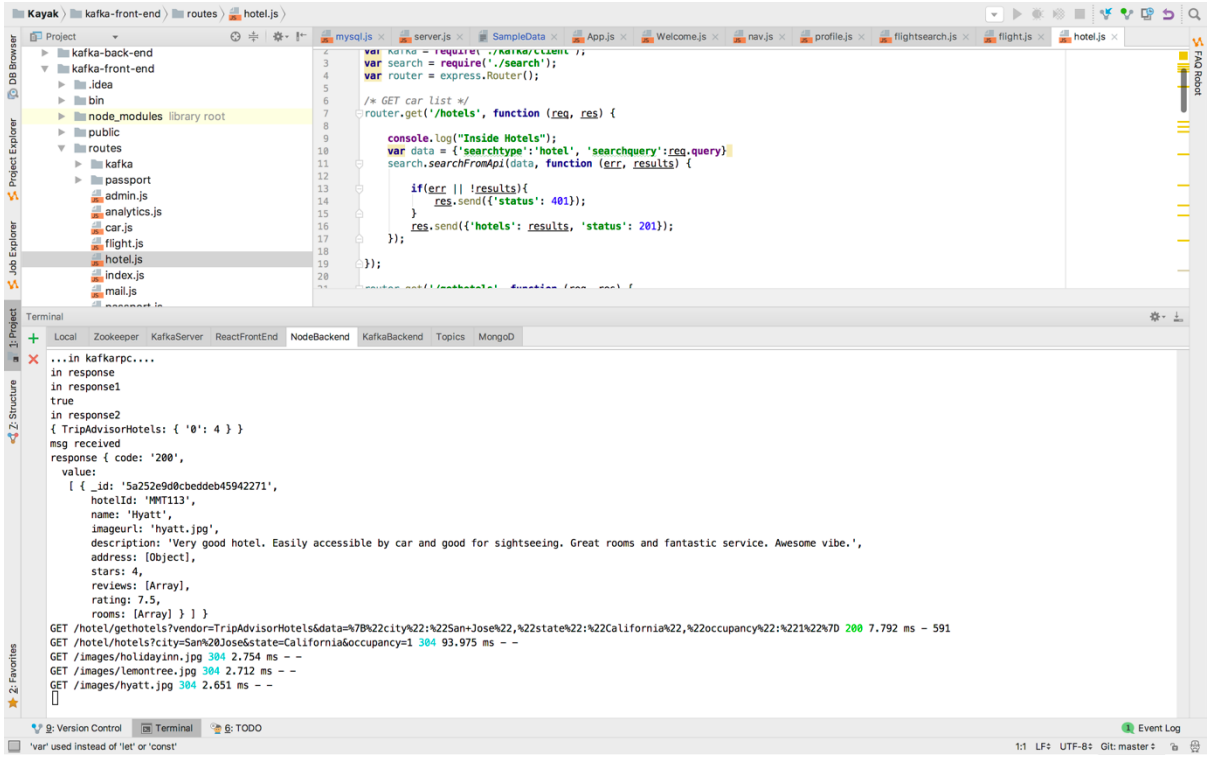




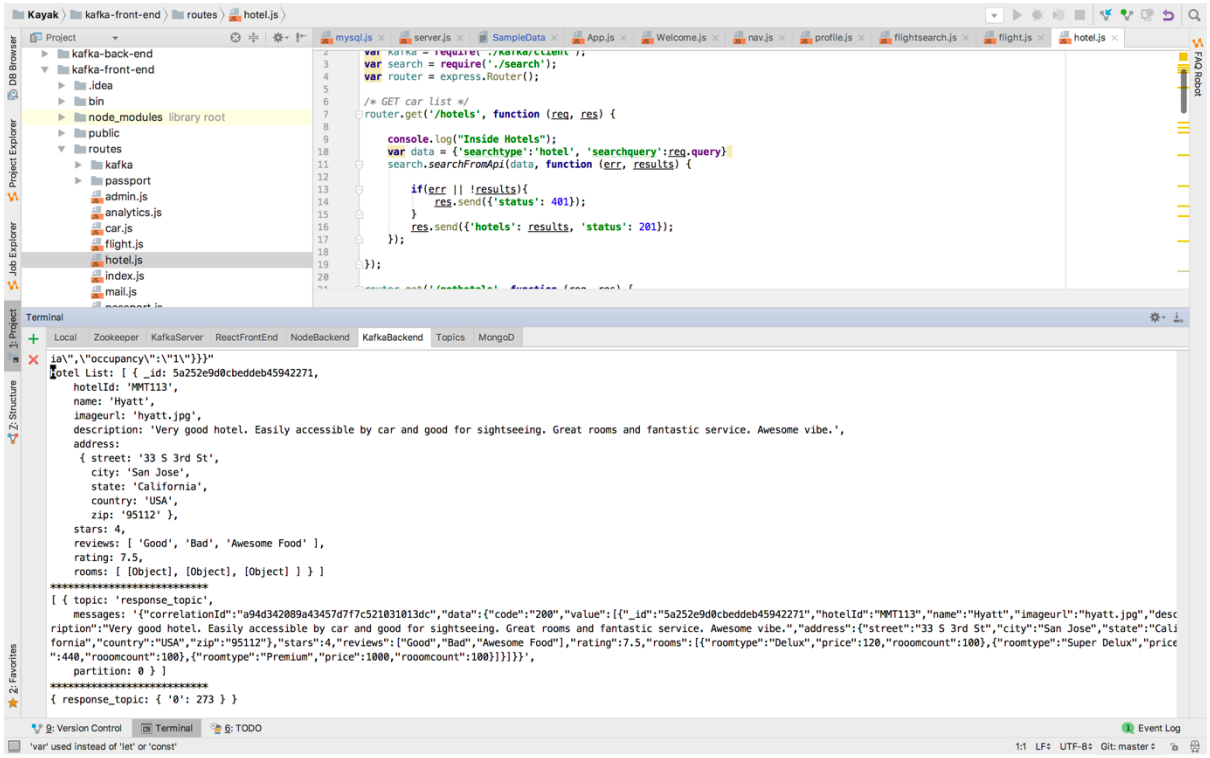

Kayak
How to take a screenshot on y
kafka-backend/sign_up.js at m
(170) Ed Sheeran - Shape o
(8) Sunil Tiwari
localhost:3000/hotellist
KAYAK
Flights
Hotels
Cars
My Account
Price
Low:$120
High:$520
Star
1
5
Holiday Inn
Express
Holiday Inn
Express
Holiday Inn
★★★☆☆
8.5
3 Reviews
$120
Delux
BOOK
$220
Super Delux
BOOK
$320
Premium
BOOK
Details
Map
Review
Very good hotel. Easily accessible by car and good for sightseeing. Great rooms and fantastic service. Awesome vibe.
Address:
101 E San Fernando Street
San Jose, California - 95112
$520


Kayak
How to take a screenshot on y
kafka-backend/sign_up.js at m
(170) Ed Sheeran - Shape o
(8) Sunil Tiwari
localhost:3000/hotellist
KAYAK
Flights
Hotels
Cars
My Account
Price
Low:$120
High:$295
Star
4
5
HYATT
PLACE
Hyatt
★★★★☆
7.5
3 Reviews
$120
Delux
BOOK
$440
Super Delux
BOOK
$1000
Premium
BOOK

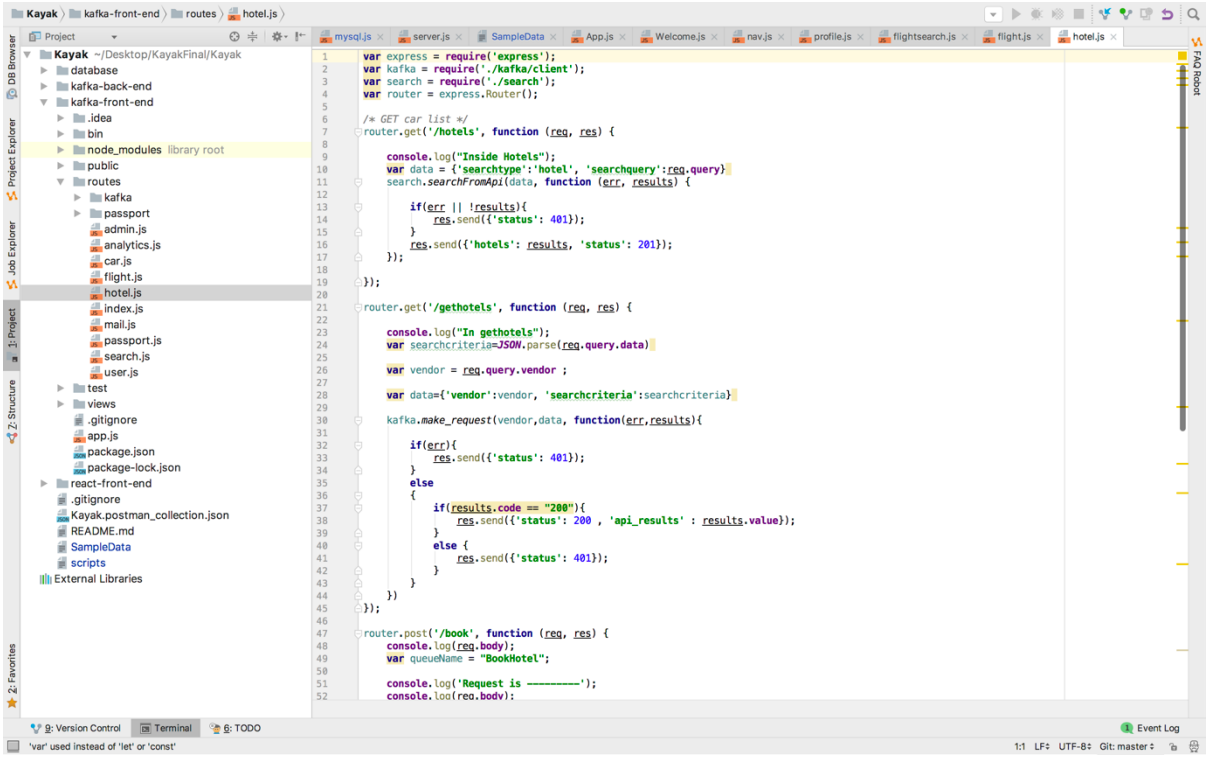
Hotel Search Page (Node Backend):



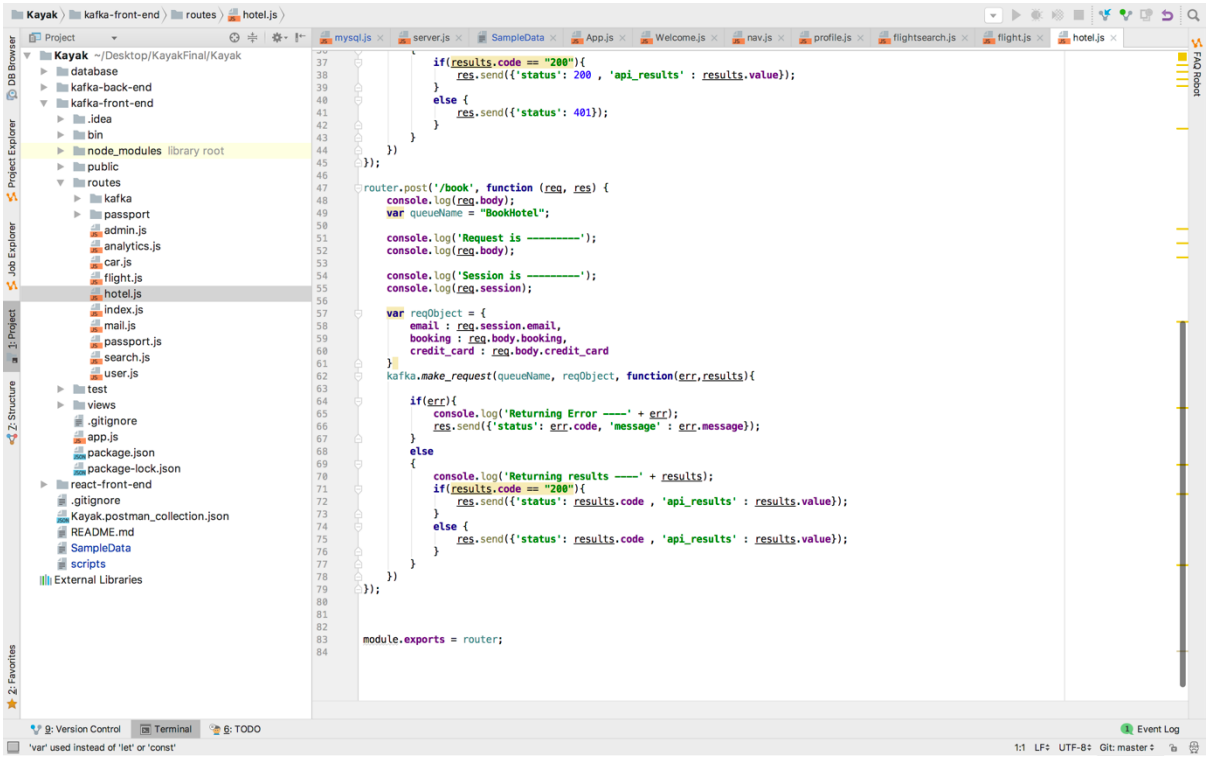
Hotel Search Page (Kafka Backend):



Hotel Search (Business Logic):

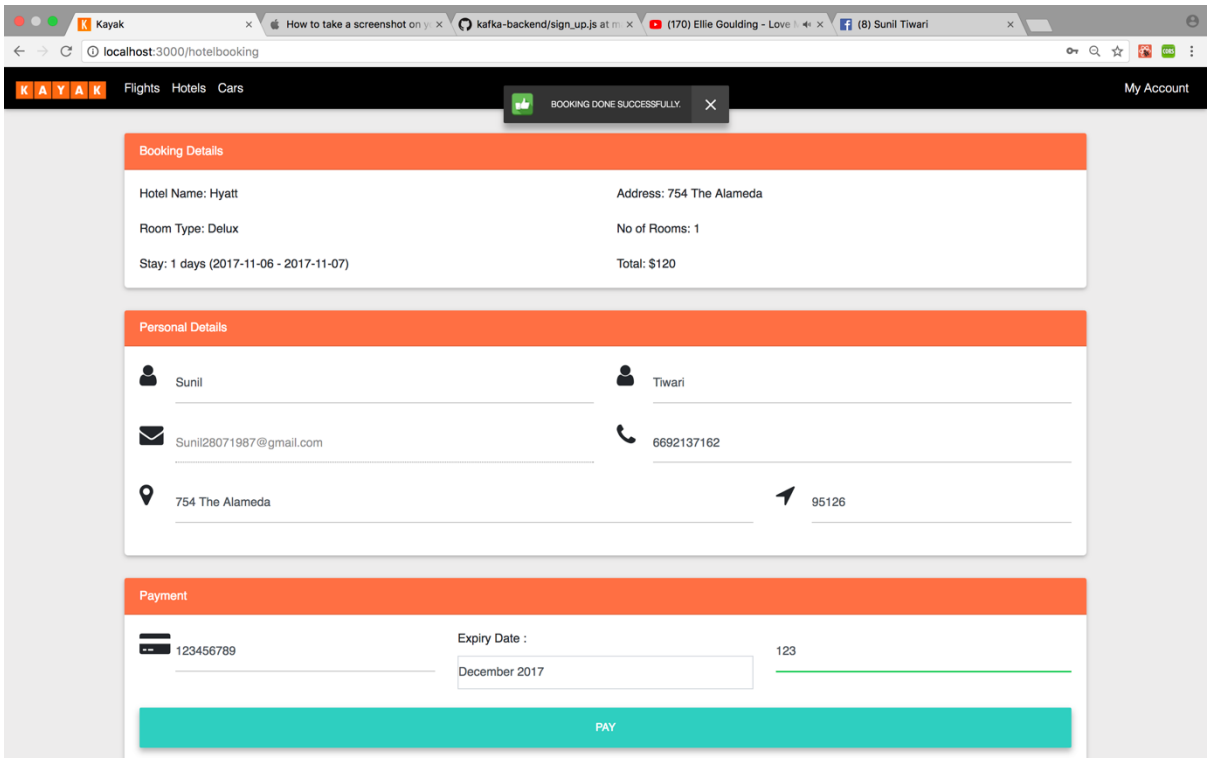


```
var express = require('express');
var kafka = require('./kafka/client');
var search = require('./search');
var router = express.Router();

/* GET car list */
router.get('/hotels', function (req, res) {

    console.log("Inside Hotels");
    var data = {'searchtype':'hotel', 'searchquery':req.query}
    search.searchFromApi(data, function (err, results) {

        if(err || !results){
            res.send({'status': 401});
        }
        res.send({'hotels': results, 'status': 201});
    });

});

router.get('/gethotels', function (req, res) {

    console.log("In gethotels");
    var searchcriteria=JSON.parse(req.query.data)

    var vendor = req.query.vendor ;

    var data={'vendor':vendor, 'searchcriteria':searchcriteria}

    kafka.make_request(vendor,data, function(err,results){

        if(err){
            res.send({'status': 401});
        }
        else
        {
            if(results.code == "200"){
                res.send({'status': 200 , 'api_results' : results.value});
            }
            else {
                res.send({'status': 401});
            }
        }
    })
});

router.post('/book', function (req, res) {
    console.log(req.body);
    var queueName = "BookHotel";

    console.log('Request is ---------');
    console.log(req.body);
```

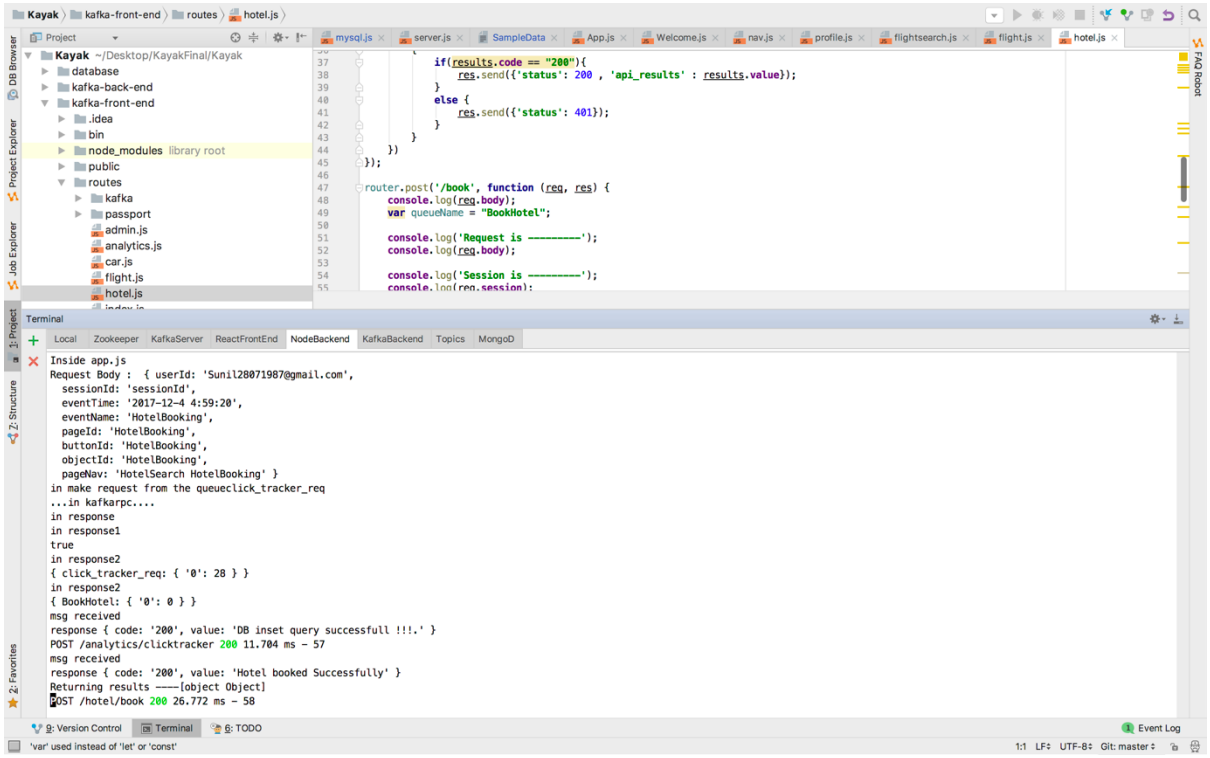


```
            if(results.code == "200"){
                res.send({'status': 200 , 'api_results' : results.value});
            }
            else {
                res.send({'status': 401});
            }
        }
    })
});

router.post('/book', function (req, res) {
    console.log(req.body);
    var queueName = "BookHotel";

    console.log('Request is ---------');
    console.log(req.body);

    console.log('Session is ---------');
    console.log(req.session);

    var reqObject = {
        email : req.session.email,
        booking : req.body.booking,
        credit_card : req.body.credit_card
    }
    kafka.make_request(queueName, reqObject, function(err,results){

        if(err){
            console.log('Returning Error ----' + err);
            res.send({'status': err.code, 'message' : err.message});
        }
        else
        {
            console.log('Returning results ----' + results);
            if(results.code == "200"){
                res.send({'status': results.code , 'api_results' : results.value});
            }
            else {
                res.send({'status': results.code , 'api_results' : results.value});
            }
        }
    })
});


module.exports = router;
```

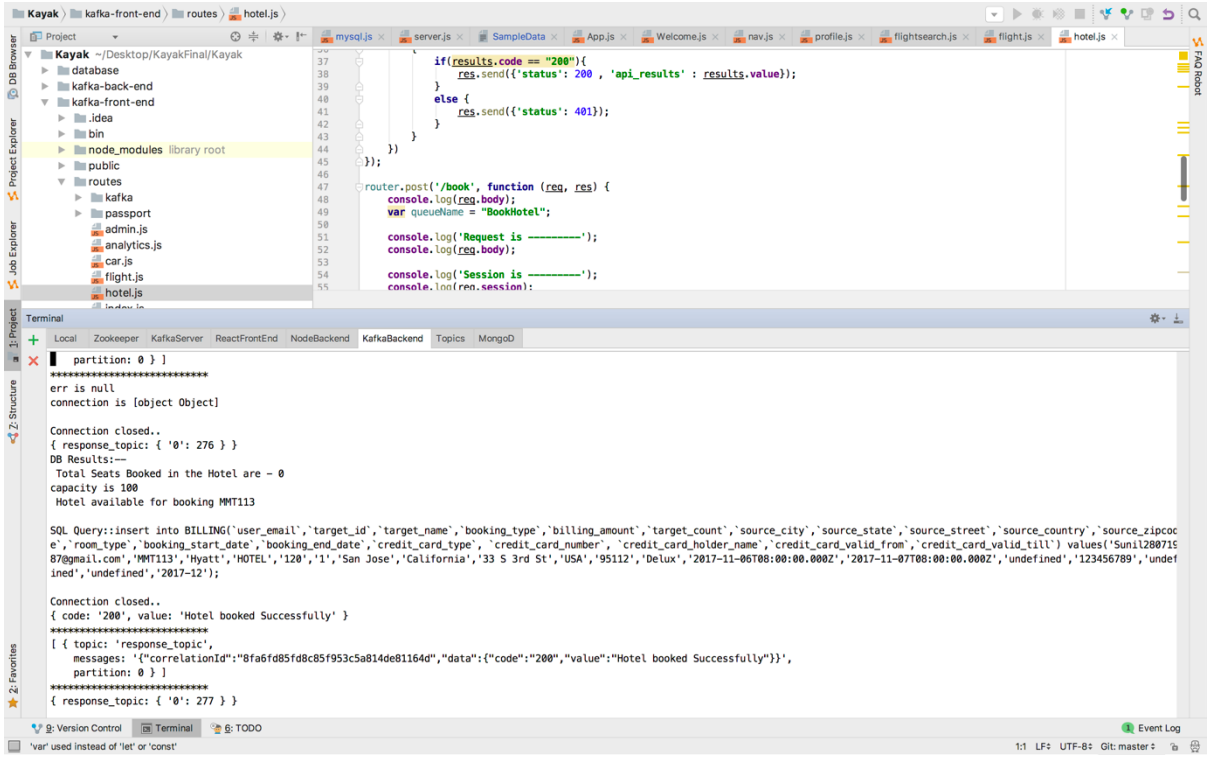
Hotel Booking Page (UI):



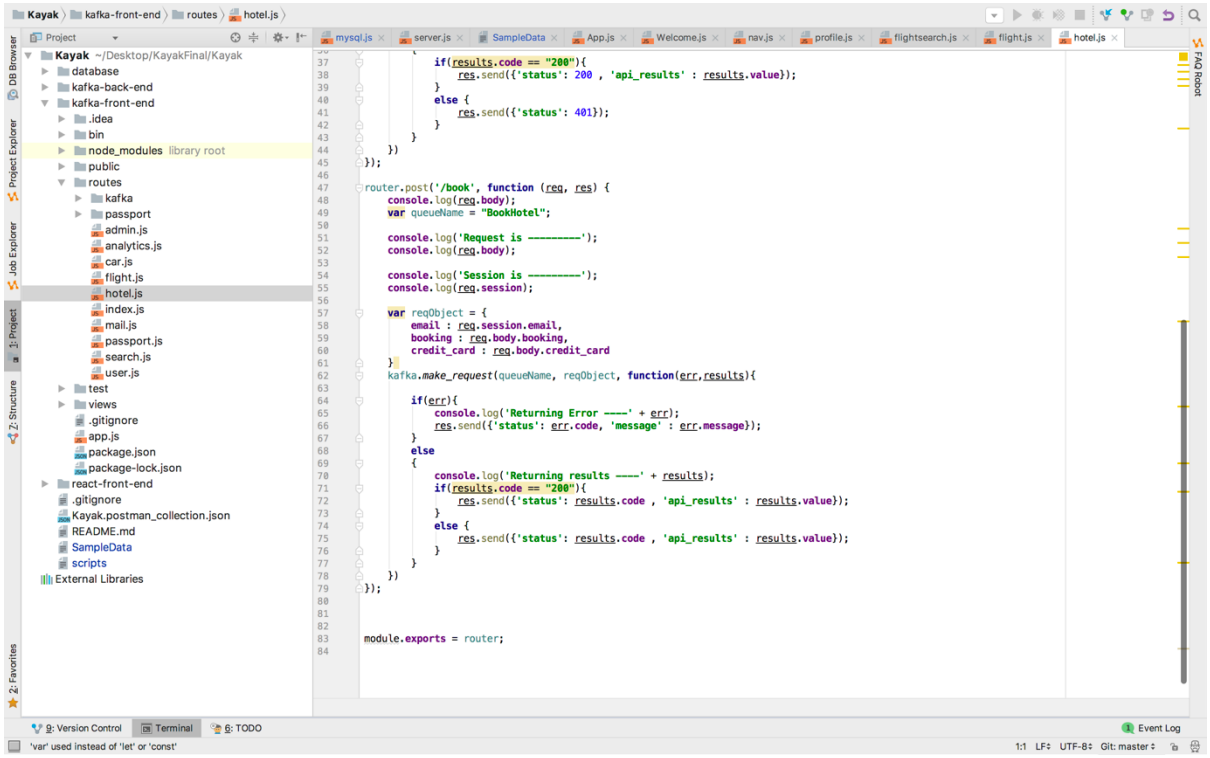
Hotel Booking Page (Node Backend):



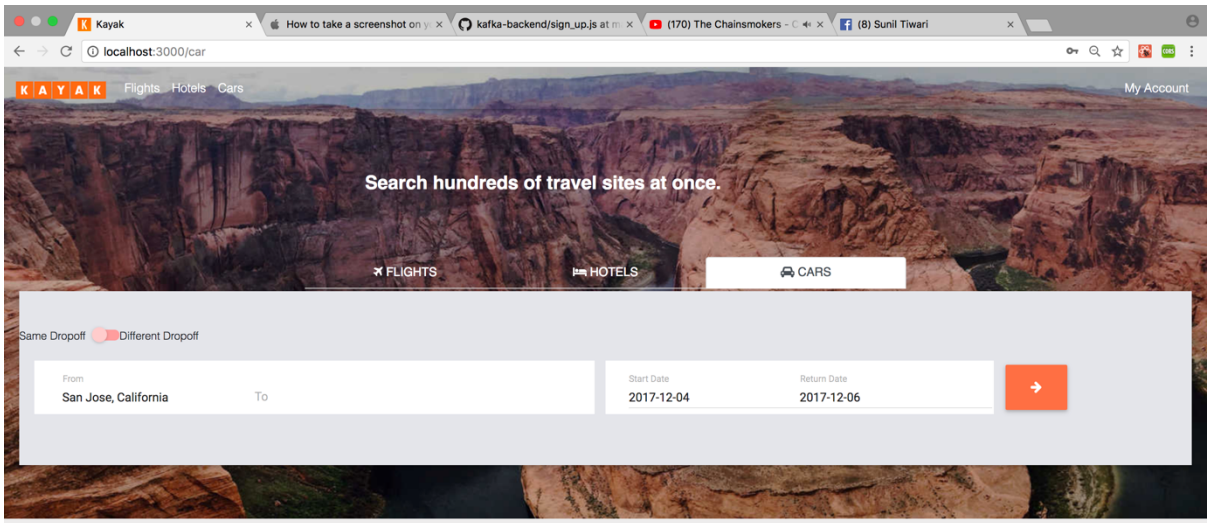
Hotel Booking Page (Kafka Backend):

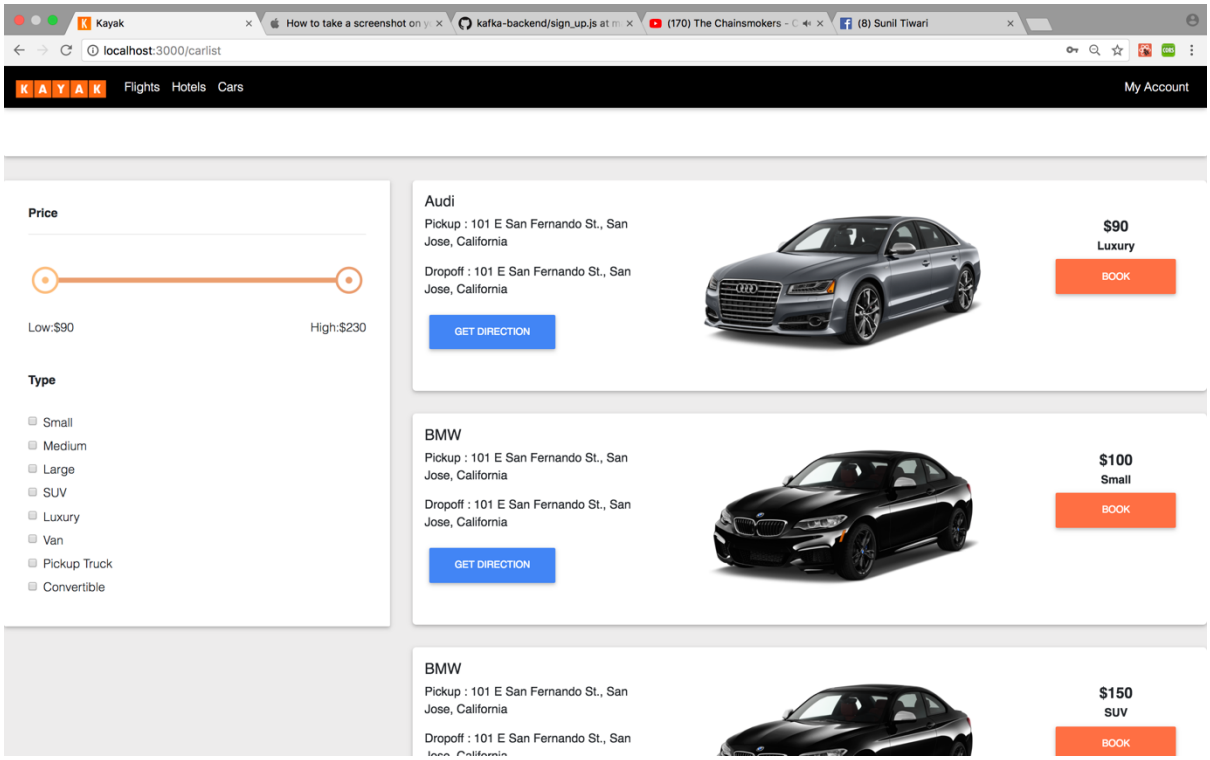


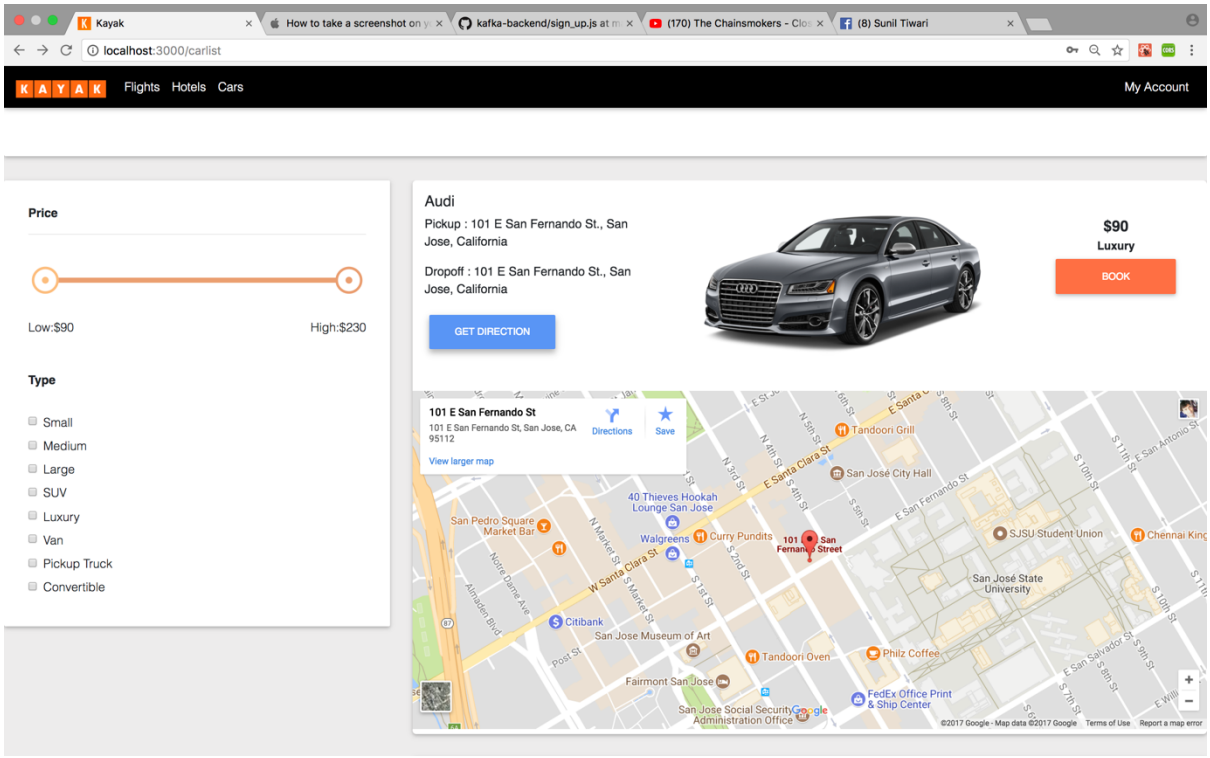
Hotel Booking (Business Logic):

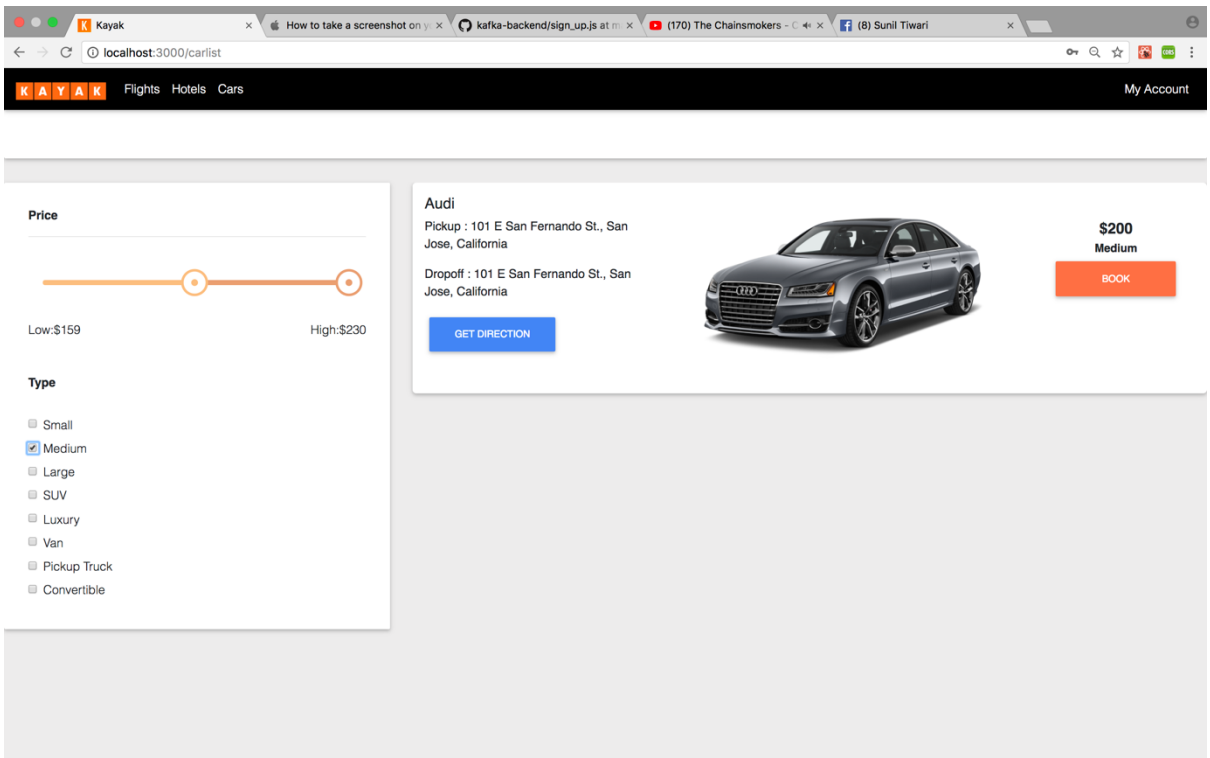


Car Search Page (UI):

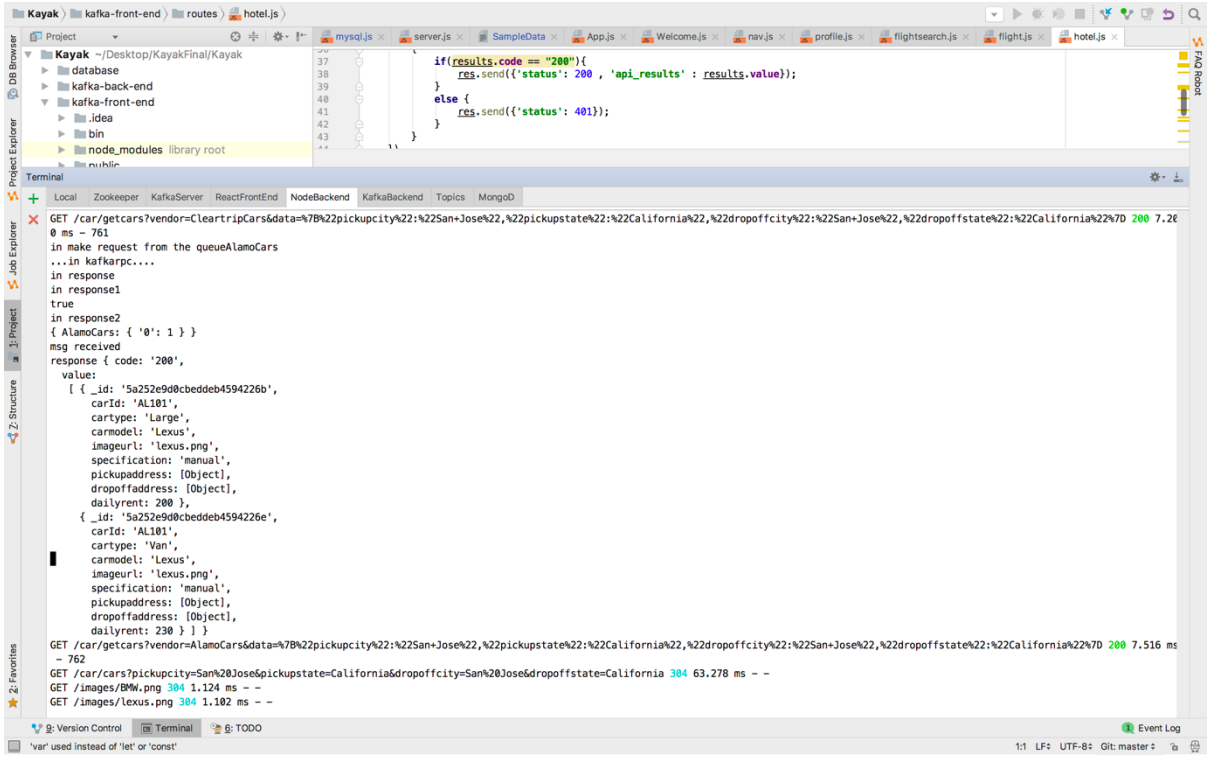




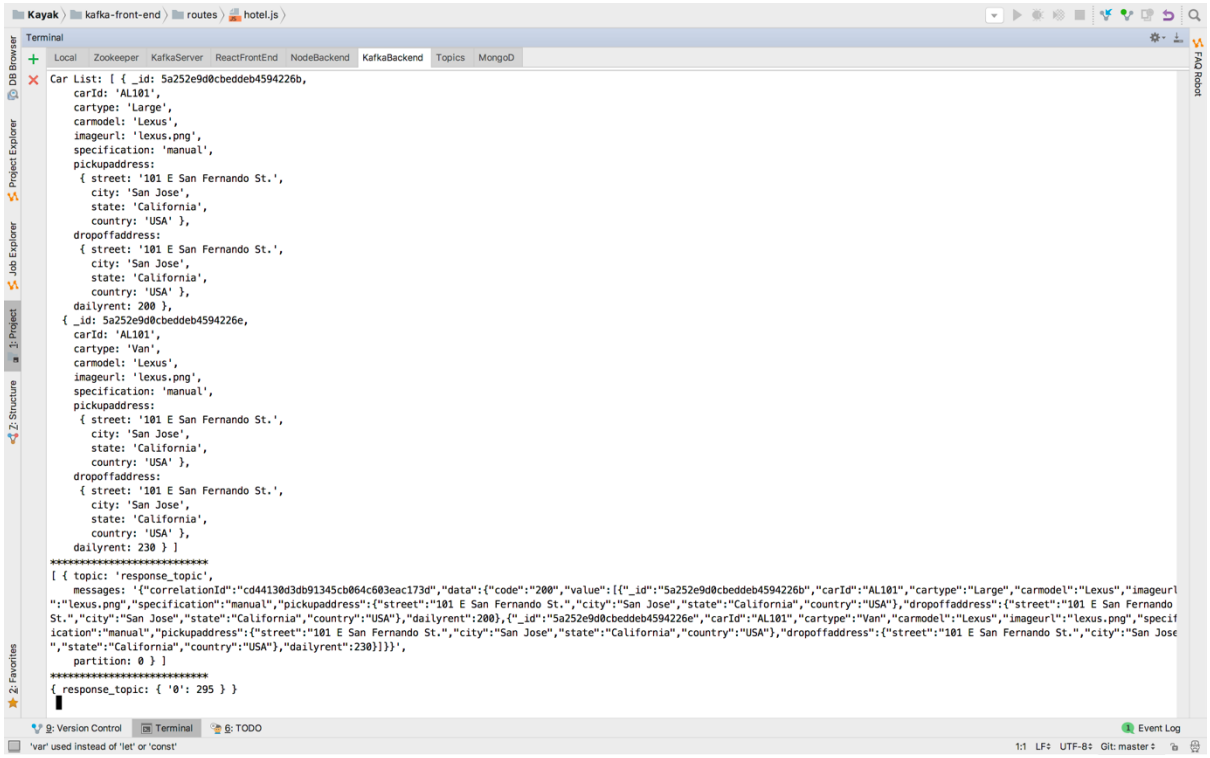

Kayak
How to take a screenshot on y
kafka-backend/sign_up.js at m
(170) The Chainsmokers - Clos
(8) Sunil Tiwari
localhost:3000/carlist
KAYAK
Flights
Hotels
Cars
My Account
Price
Low:$90
High:$230
Type
Small
Medium
Large
SUV
Luxury
Van
Pickup Truck
Convertible
Audi
Pickup : 101 E San Fernando St., San Jose, California
Dropoff : 101 E San Fernando St., San Jose, California
GET DIRECTION
$90
Luxury
BOOK
101 E San Fernando St
101 E San Fernando St, San Jose, CA 95112
Directions
Save
View larger map
Tandoori Grill
San José City Hall
40 Thieves Hookah Lounge San Jose
San Pedro Square Market Bar
Walgreens
Curry Pundits
SJSU Student Union
San José State University
Citibank
San Jose Museum of Art
Tandoori Oven
Philz Coffee
Fairmont San Jose
FedEx Office Print & Ship Center
San Jose Social Security Administration Office
©2017 Google - Map data ©2017 Google
Terms of Use
Report a map error


Kayak
How to take a screenshot on y
kafka-backend/sign_up.js at m
(170) The Chainsmokers - C
(8) Sunil Tiwari
localhost:3000/carlist
KAYAK
Flights
Hotels
Cars
My Account
Price
Low:$159
High:$230
Type
Small
Medium
Large
SUV
Luxury
Van
Pickup Truck
Convertible
Audi
Pickup : 101 E San Fernando St., San Jose, California
Dropoff : 101 E San Fernando St., San Jose, California
GET DIRECTION
$200
Medium
BOOK

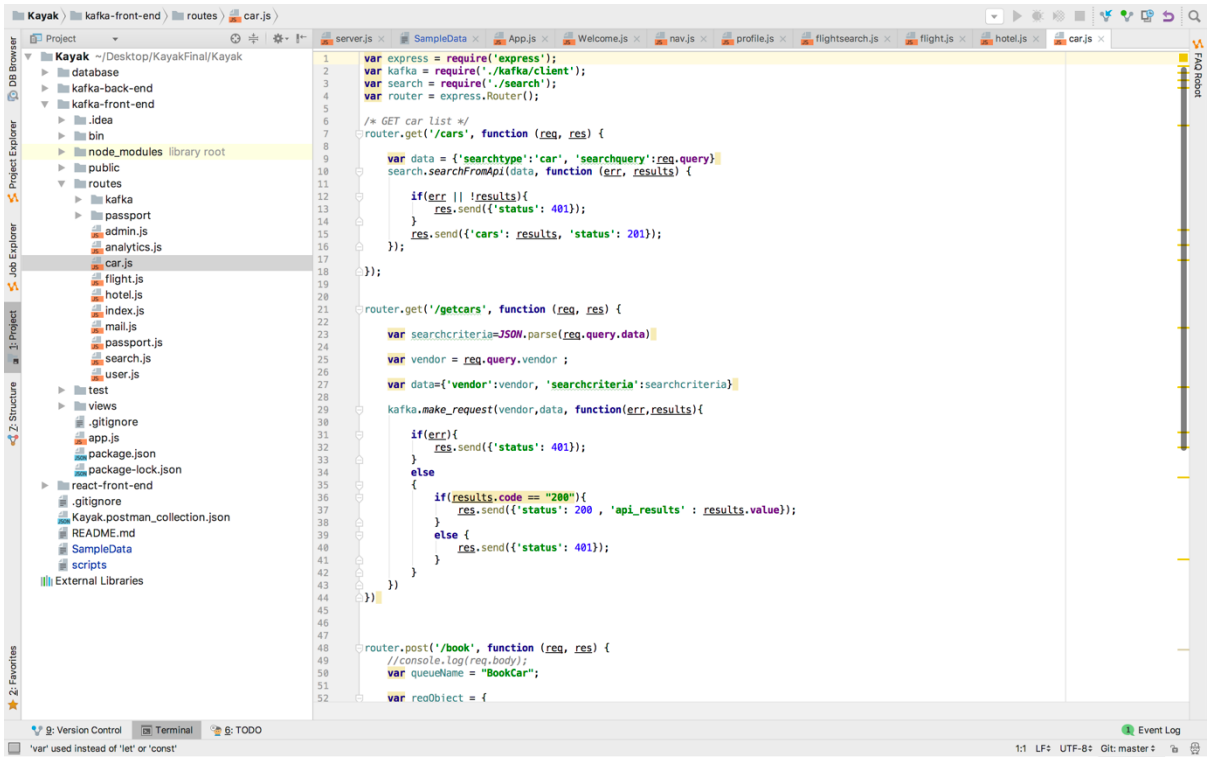
Car Search Page (Node Backend):



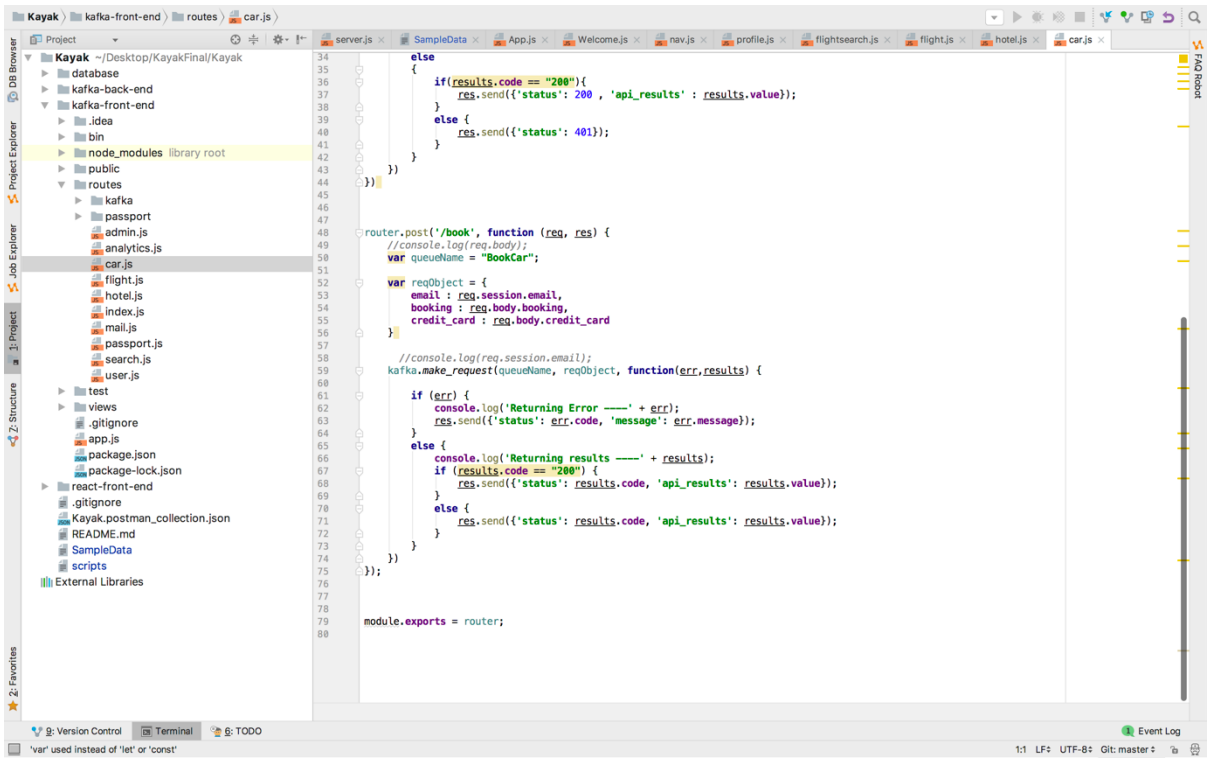
Car Search Page (Kafka Backend):



Car Search (Business Logic):

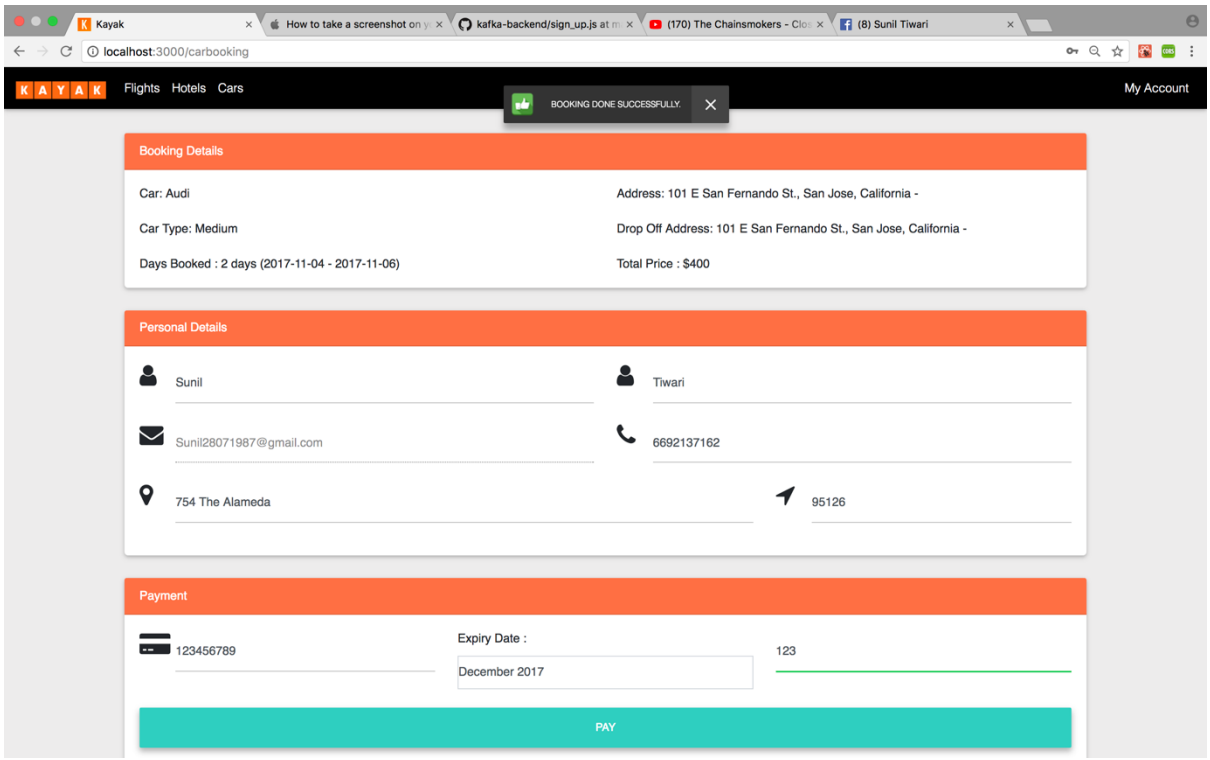


```
var express = require('express');
var kafka = require('./kafka/client');
var search = require('./search');
var router = express.Router();

/* GET car list */
router.get('/cars', function (req, res) {

    var data = {'searchtype':'car', 'searchquery':req.query}
    search.searchFromApi(data, function (err, results) {

        if(err || !results){
            res.send({'status': 401});
        }
        res.send({'cars': results, 'status': 201});
    });

});


router.get('/getcars', function (req, res) {

    var searchcriteria=JSON.parse(req.query.data)

    var vendor = req.query.vendor ;

    var data={'vendor':vendor, 'searchcriteria':searchcriteria}

    kafka.make_request(vendor,data, function(err,results){

        if(err){
            res.send({'status': 401});
        }
        else
        {
            if(results.code == "200"){
                res.send({'status': 200 , 'api_results' : results.value});
            }
            else {
                res.send({'status': 401});
            }
        }
    })
})



router.post('/book', function (req, res) {
    //console.log(req.body);
    var queueName = "BookCar";

    var reqObject = {
```

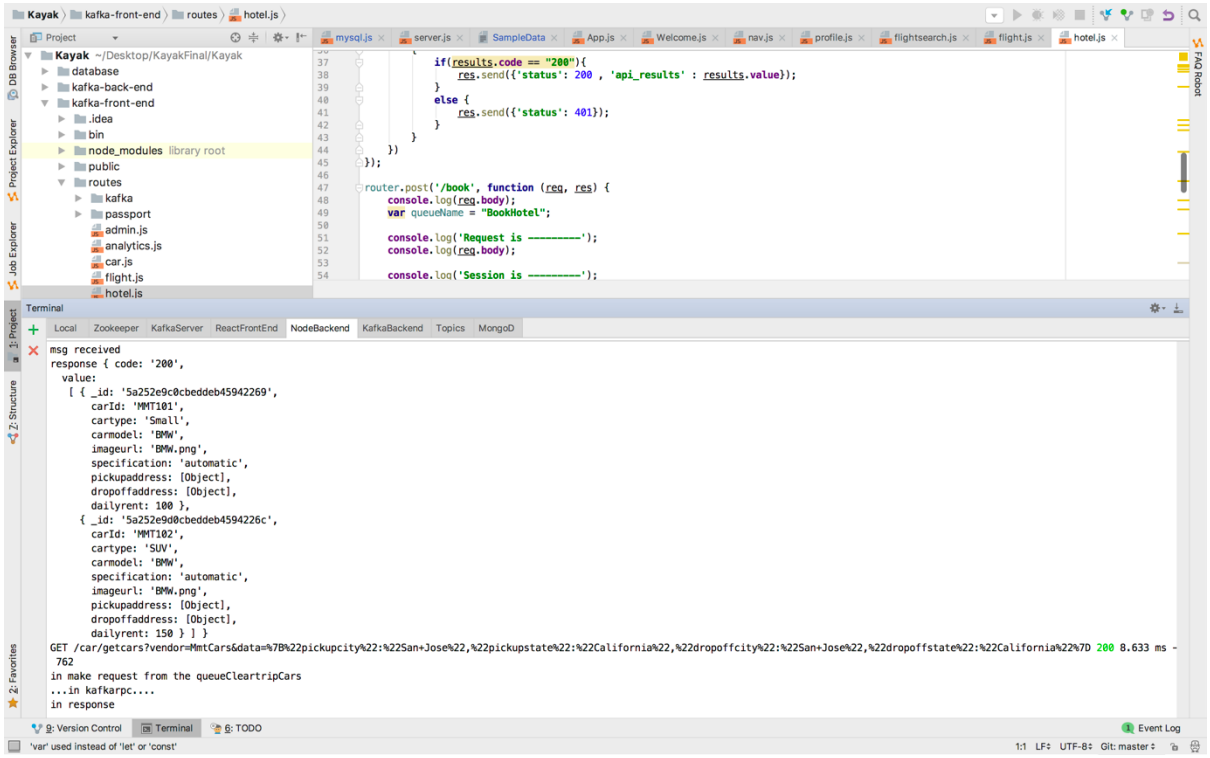


```
        else
        {
            if(results.code == "200"){
                res.send({'status': 200 , 'api_results' : results.value});
            }
            else {
                res.send({'status': 401});
            }
        }
    })
})



router.post('/book', function (req, res) {
    //console.log(req.body);
    var queueName = "BookCar";

    var reqObject = {
        email : req.session.email,
        booking : req.body.booking,
        credit_card : req.body.credit_card
    }

      //console.log(req.session.email);
    kafka.make_request(queueName, reqObject, function(err,results) {

        if (err) {
            console.log('Returning Error ----' + err);
            res.send({'status': err.code, 'message': err.message});
        }
        else {
            console.log('Returning results ----' + results);
            if (results.code == "200") {
                res.send({'status': results.code, 'api_results': results.value});
            }
            else {
                res.send({'status': results.code, 'api_results': results.value});
            }
        }
    })
});


module.exports = router;
```

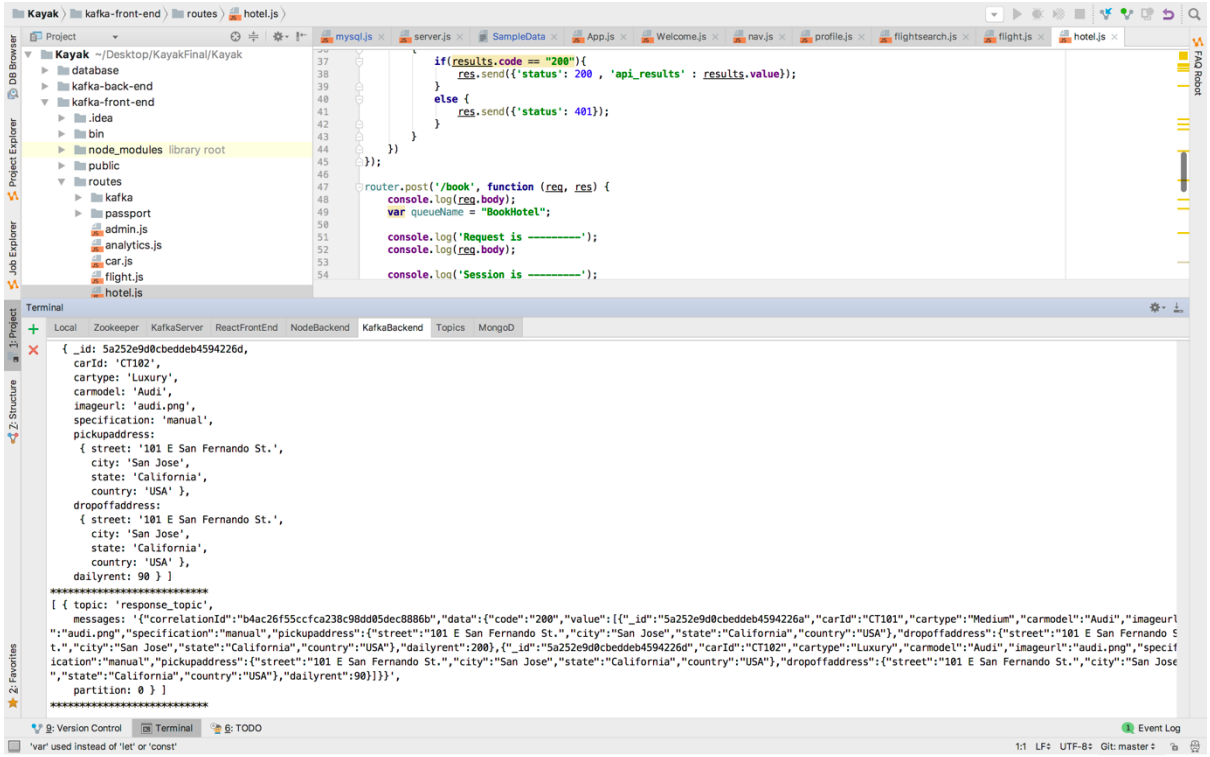
Car Booking Page (UI):



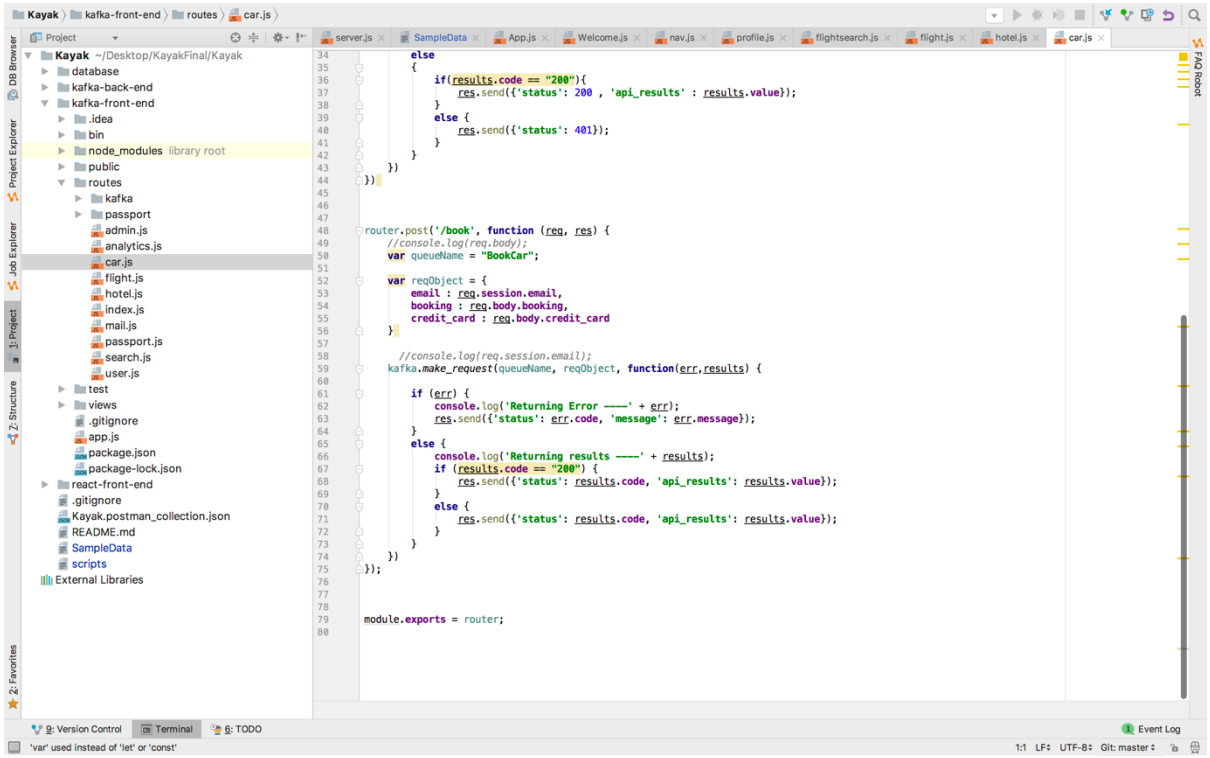
Car Booking Page (Node Backend):



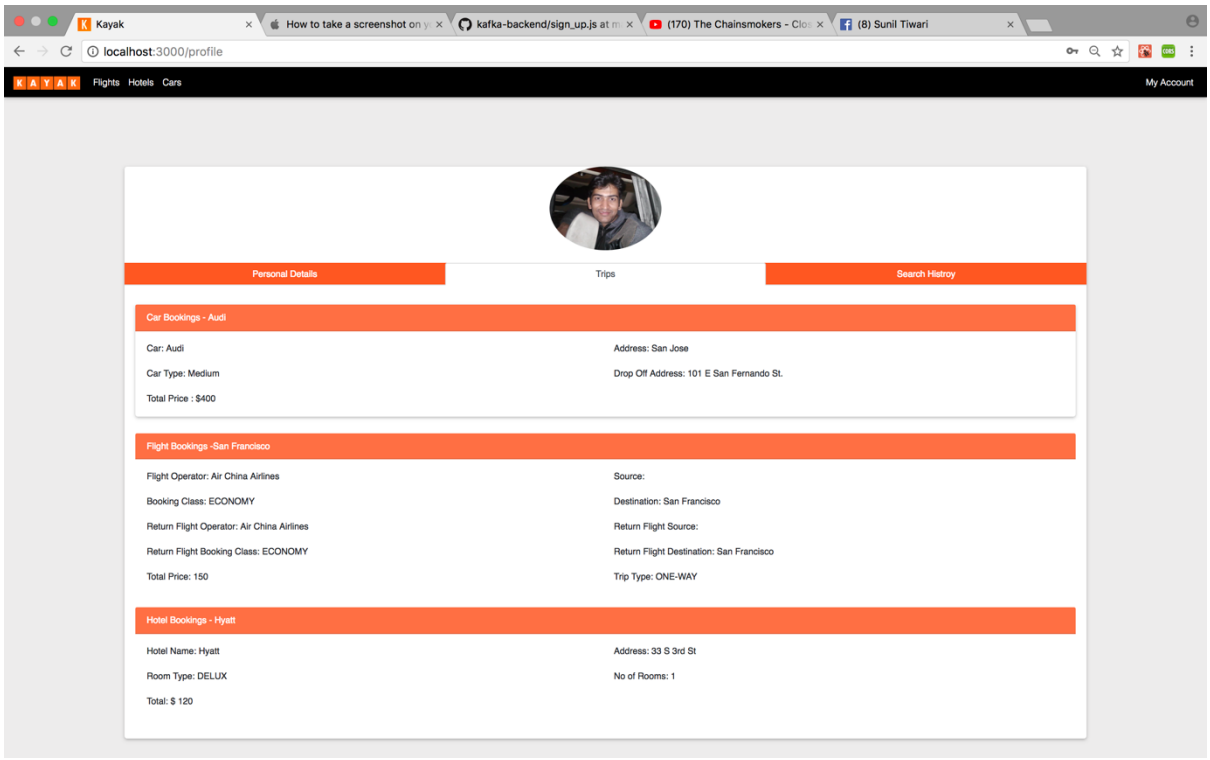
Car Booking Page (Kafka Backend):



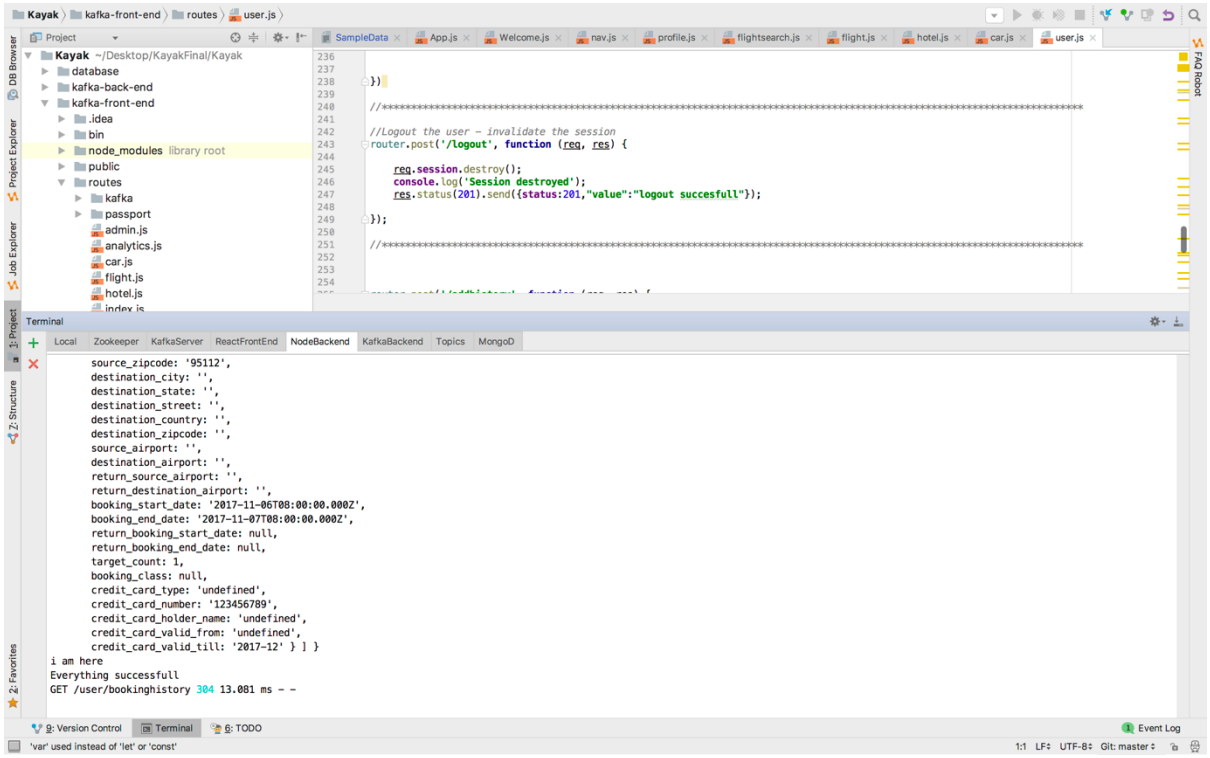
Car Booking (Business Logic):



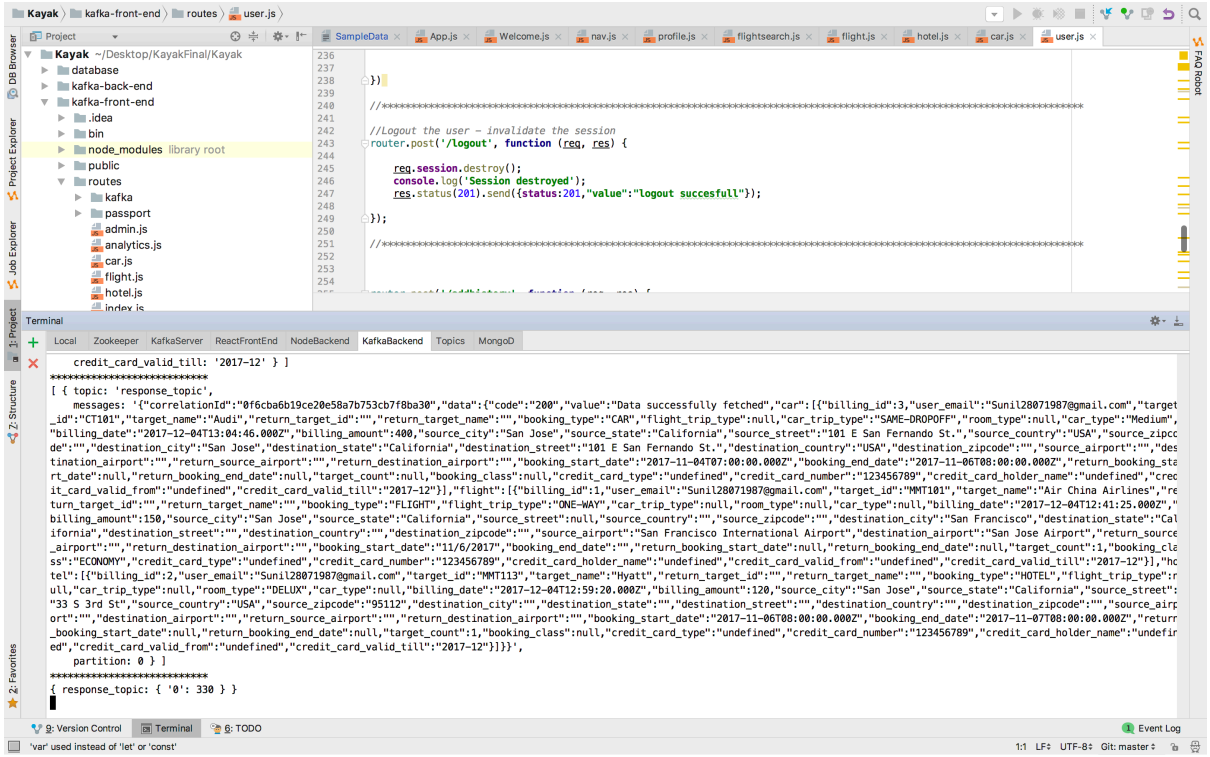
User Booking History Page (UI):



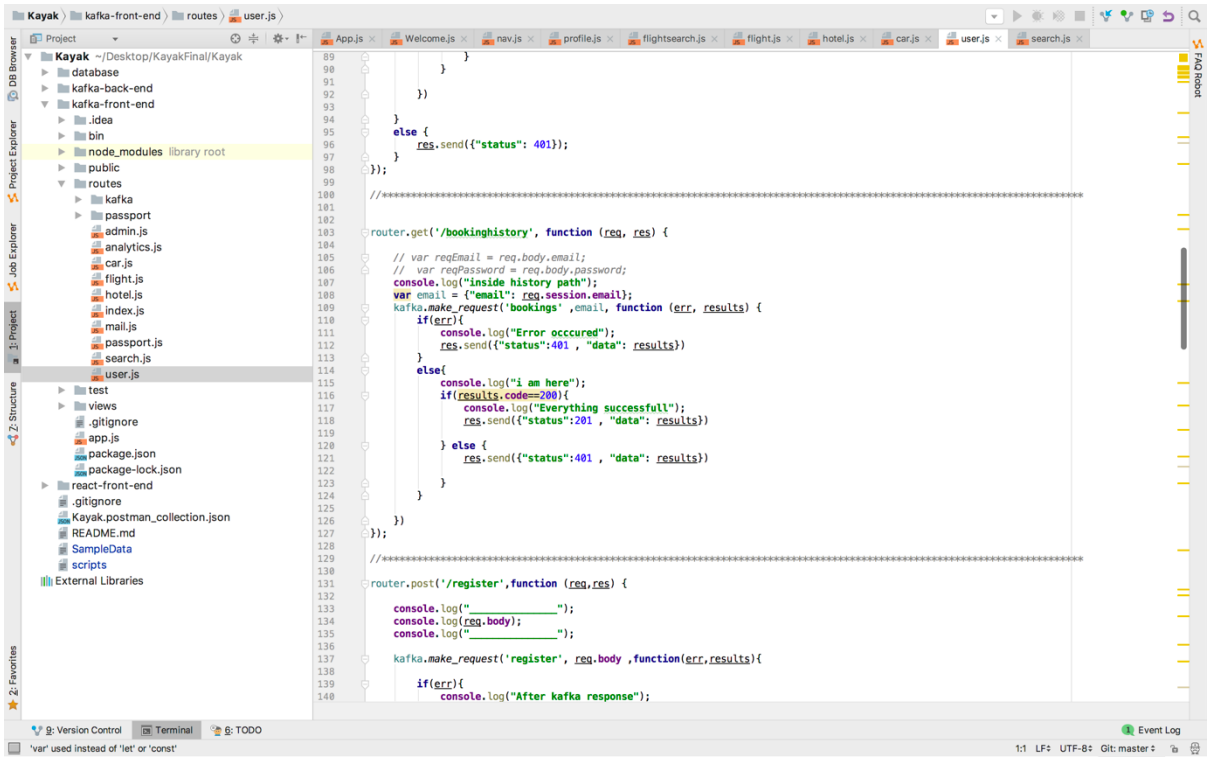
User Booking History Page (Node Backend):



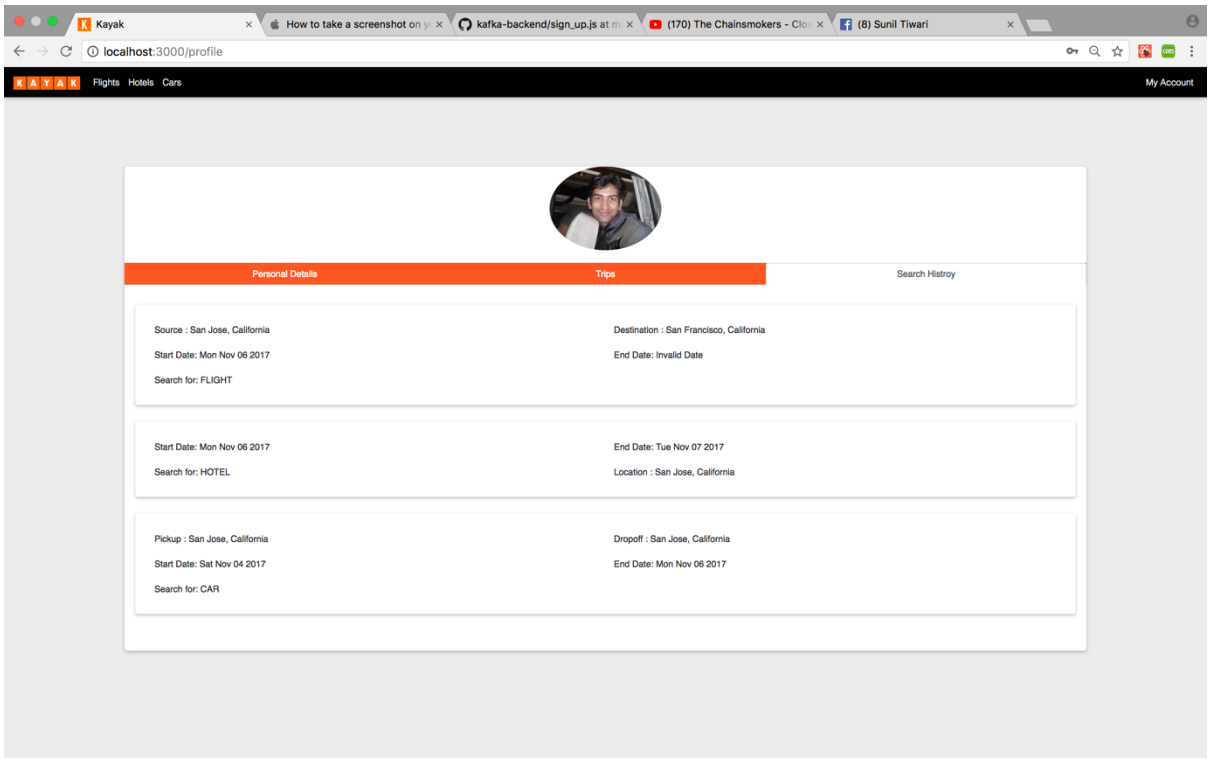
User Booking History Page (Kafka Backend):



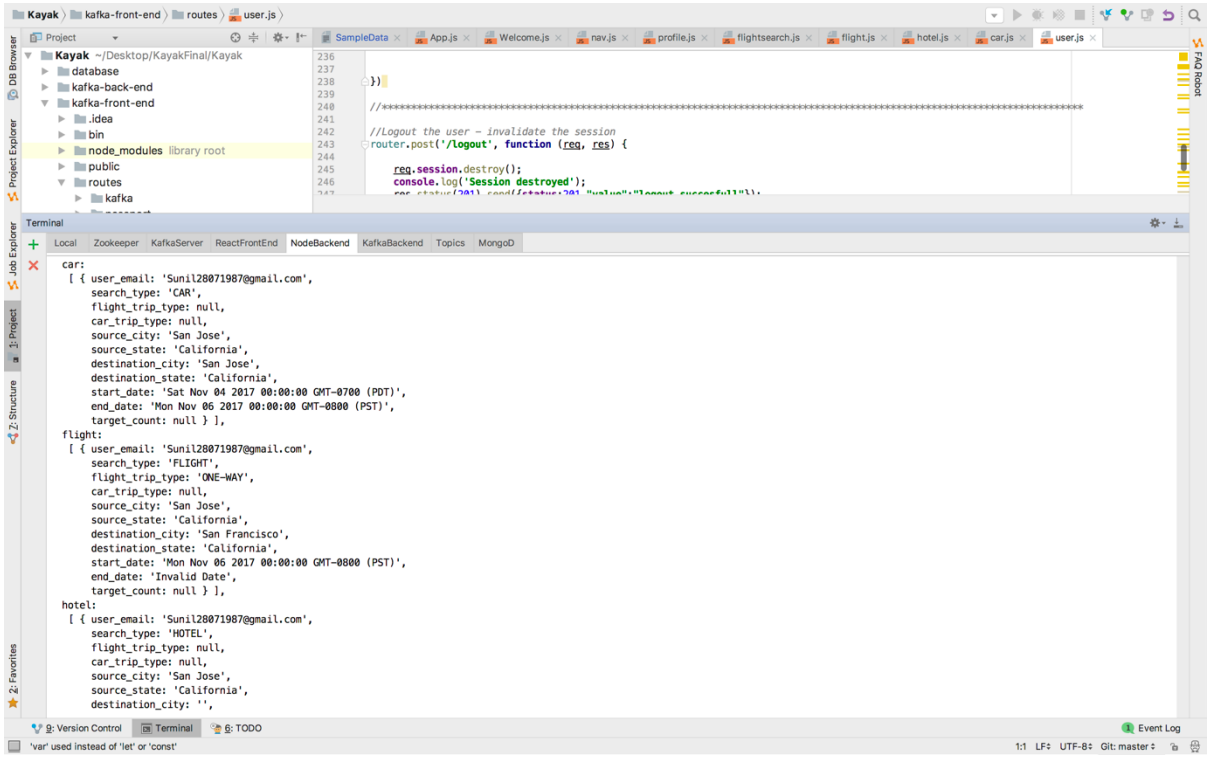
User Booking History Page (Business Logic):



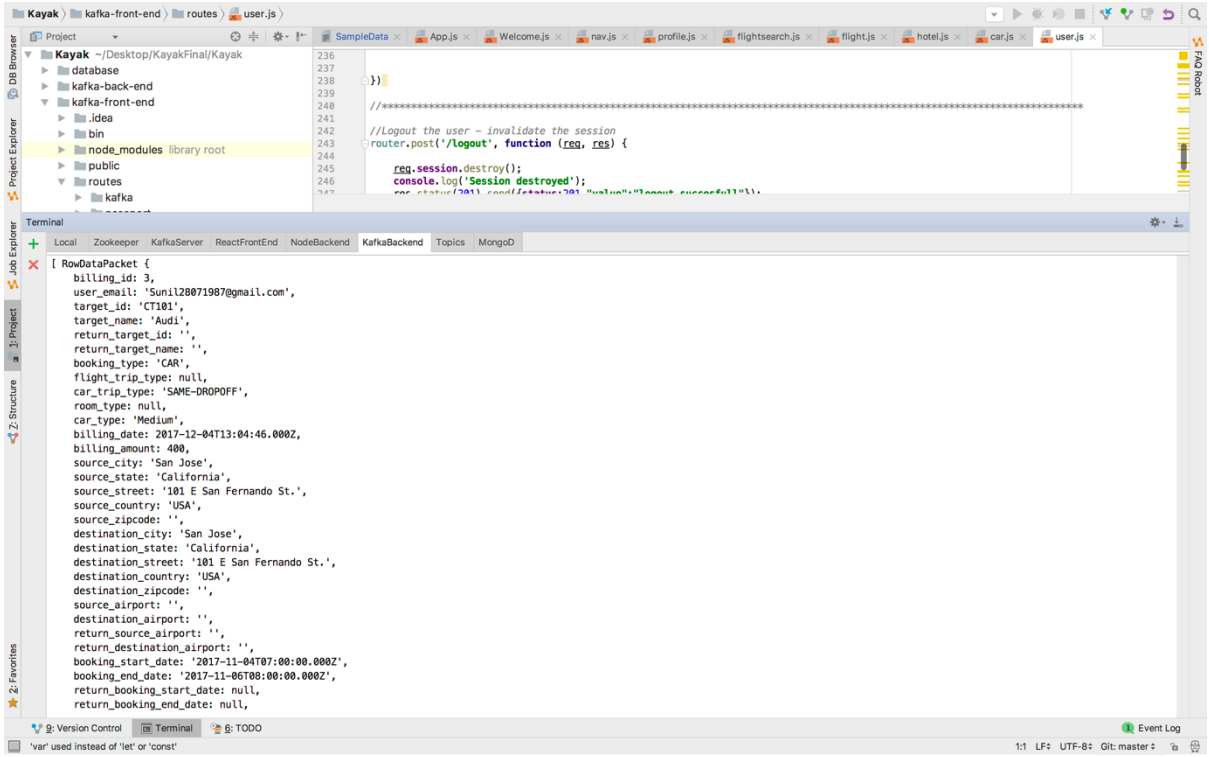
User Search History Page (UI):



User Search History Page (Node Backend):

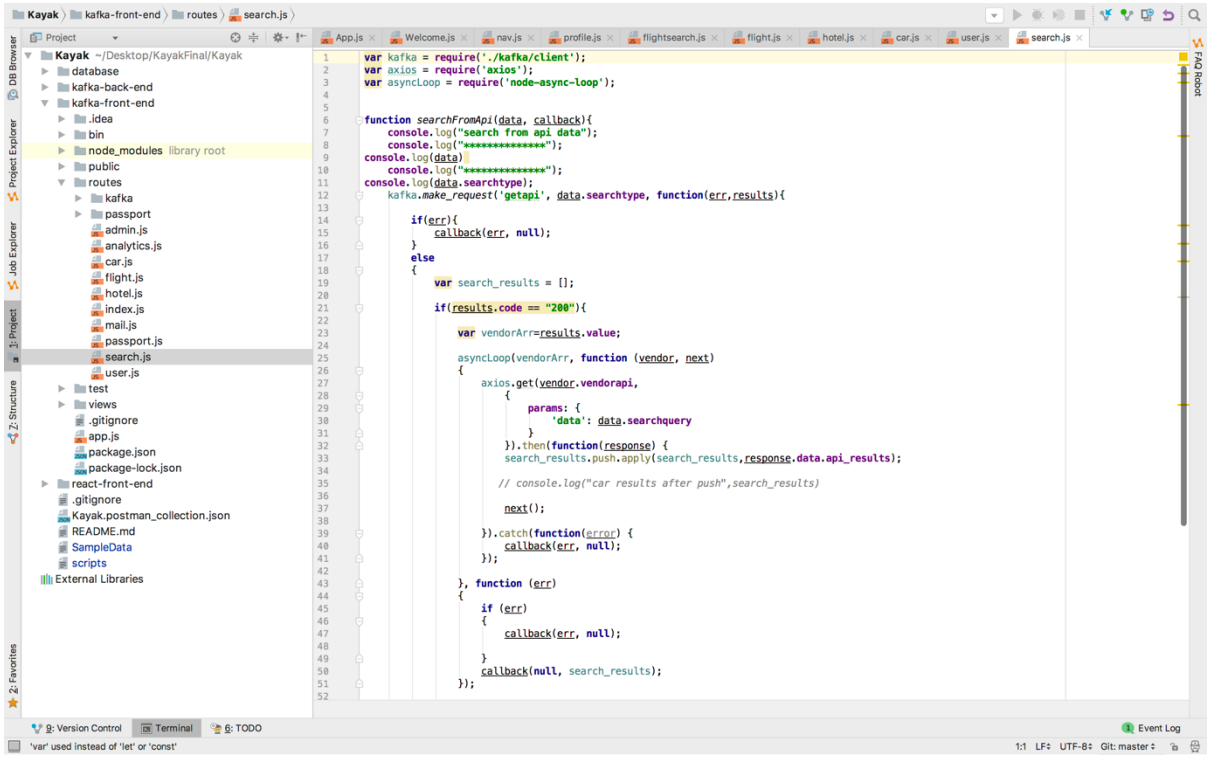


User Search History Page (Kafka Backend):

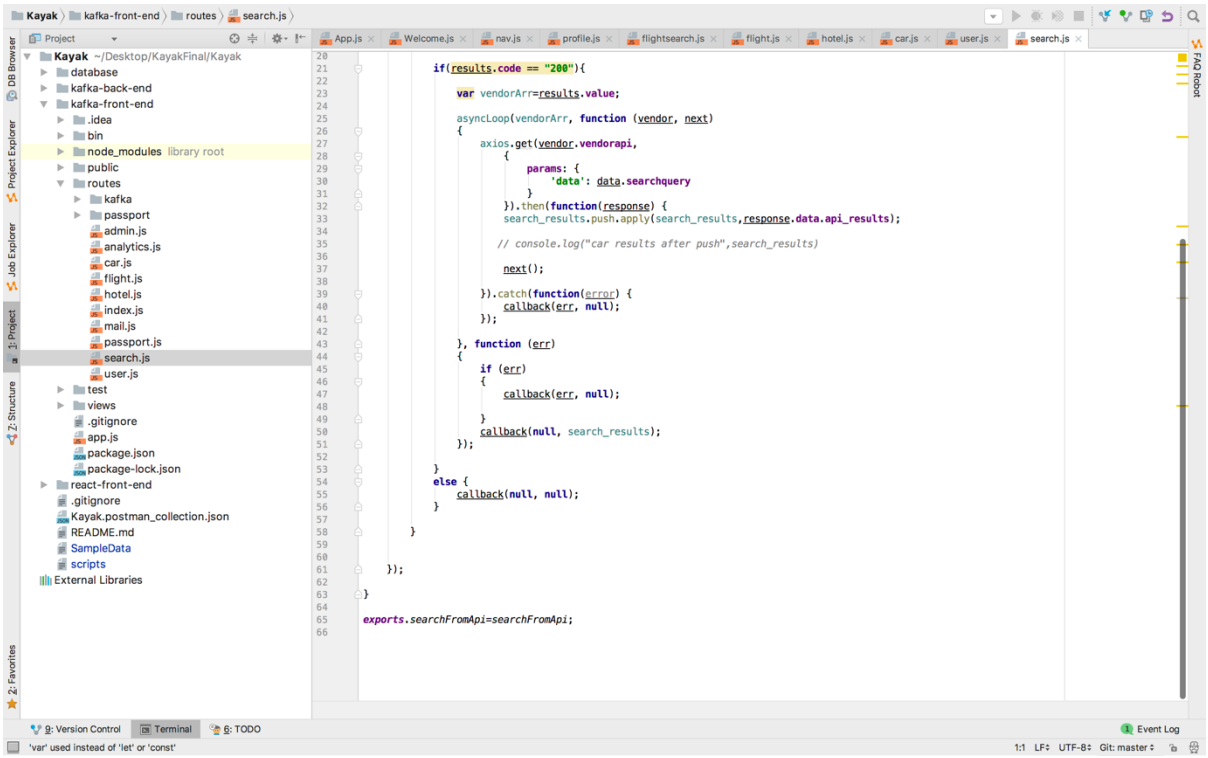


```
[ RowDataPacket {
    billing_id: 3,
    user_email: 'Sunil28071987@gmail.com',
    target_id: 'CT101',
    target_name: 'Audi',
    return_target_id: '',
    return_target_name: '',
    booking_type: 'CAR',
    flight_trip_type: null,
    car_trip_type: 'SAME-DROPOFF',
    room_type: null,
    car_type: 'Medium',
    billing_date: 2017-12-04T13:04:46.000Z,
    billing_amount: 400,
    source_city: 'San Jose',
    source_state: 'California',
    source_street: '101 E San Fernando St.',
    source_country: 'USA',
    source_zipcode: '',
    destination_city: 'San Jose',
    destination_state: 'California',
    destination_street: '101 E San Fernando St.',
    destination_country: 'USA',
    destination_zipcode: '',
    source_airport: '',
    destination_airport: '',
    return_source_airport: '',
    return_destination_airport: '',
    booking_start_date: '2017-11-04T07:00:00.000Z',
    booking_end_date: '2017-11-06T08:00:00.000Z',
    return_booking_start_date: null,
    return_booking_end_date: null,
```

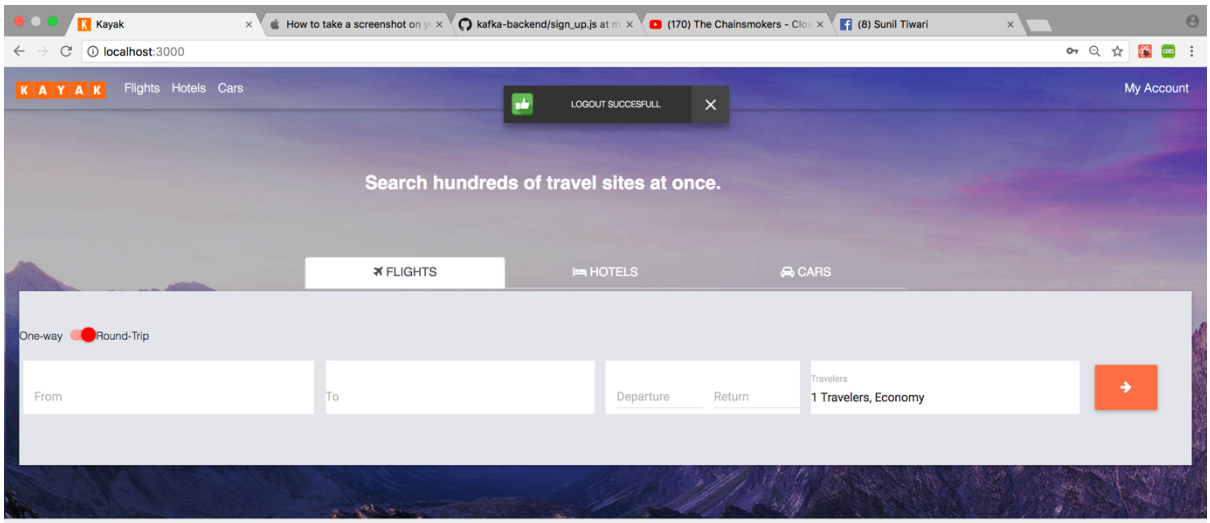
User Search History Page (Business Logic):



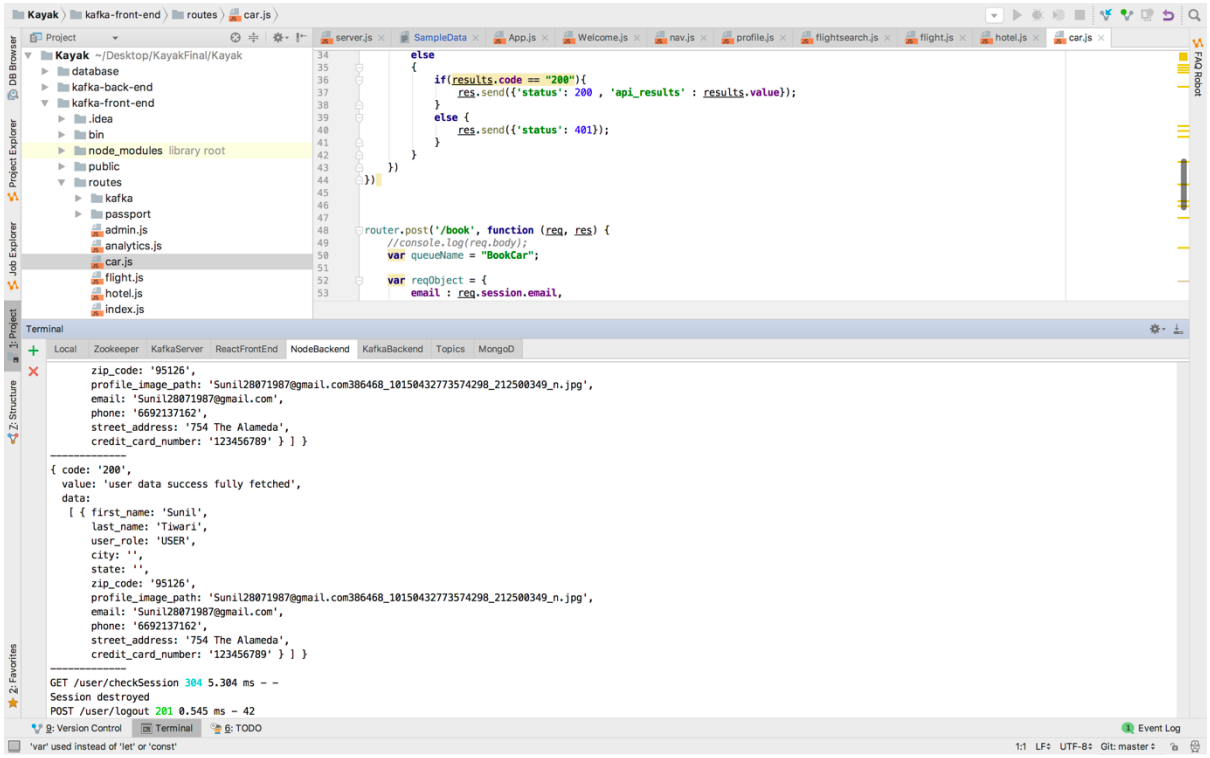
```
var kafka = require('./kafka/client');
var axios = require('axios');
var asyncLoop = require('node-async-loop');


function searchFromApi(data, callback){
    console.log("search from api data");
    console.log("**************");
console.log(data)
    console.log("**************");
console.log(data.searchtype);
    kafka.make_request('getapi', data.searchtype, function(err,results){

        if(err){
            callback(err, null);
        }
        else
        {
            var search_results = [];

            if(results.code == "200"){

                var vendorArr=results.value;

                asyncLoop(vendorArr, function (vendor, next)
                {
                    axios.get(vendor.vendorapi,
                        {
                            params: {
                                'data': data.searchquery
                            }
                        }).then(function(response) {
                        search_results.push.apply(search_results,response.data.api_results);

                        // console.log("car results after push",search_results)

                        next();

                    }).catch(function(error) {
                        callback(err, null);
                    });

                }, function (err)
                {
                    if (err)
                    {
                        callback(err, null);

                    }
                    callback(null, search_results);
                });
```


Kayak 〉 kafka-front-end 〉 routes 〉 search.js 〉
Project
Kayak ~/Desktop/KayakFinal/Kayak
database
kafka-back-end
kafka-front-end
.idea
bin
node_modules library root
public
routes
kafka
passport
admin.js
analytics.js
car.js
flight.js
hotel.js
index.js
mail.js
passport.js
search.js
user.js
test
views
.gitignore
app.js
package.json
package-lock.json
react-front-end
.gitignore
Kayak.postman_collection.json
README.md
SampleData
scripts
External Libraries
App.js
Welcome.js
nav.js
profile.js
flightsearch.js
flight.js
hotel.js
car.js
user.js
search.js
if(results.code == "200"){
var vendorArr=results.value;
asyncLoop(vendorArr, function (vendor, next)
{
axios.get(vendor.vendorapi,
{
params: {
'data': data.searchquery
}
}).then(function(response) {
search_results.push.apply(search_results,response.data.api_results);
// console.log("car results after push",search_results)
next();
}).catch(function(error) {
callback(err, null);
});
}, function (err)
{
if (err)
{
callback(err, null);
}
callback(null, search_results);
});
}
else {
callback(null, null);
}
}
});
}
exports.searchFromApi=searchFromApi;
9: Version Control
Terminal
6: TODO
Event Log
'var' used instead of 'let' or 'const'
1:1 LF UTF-8 Git: master

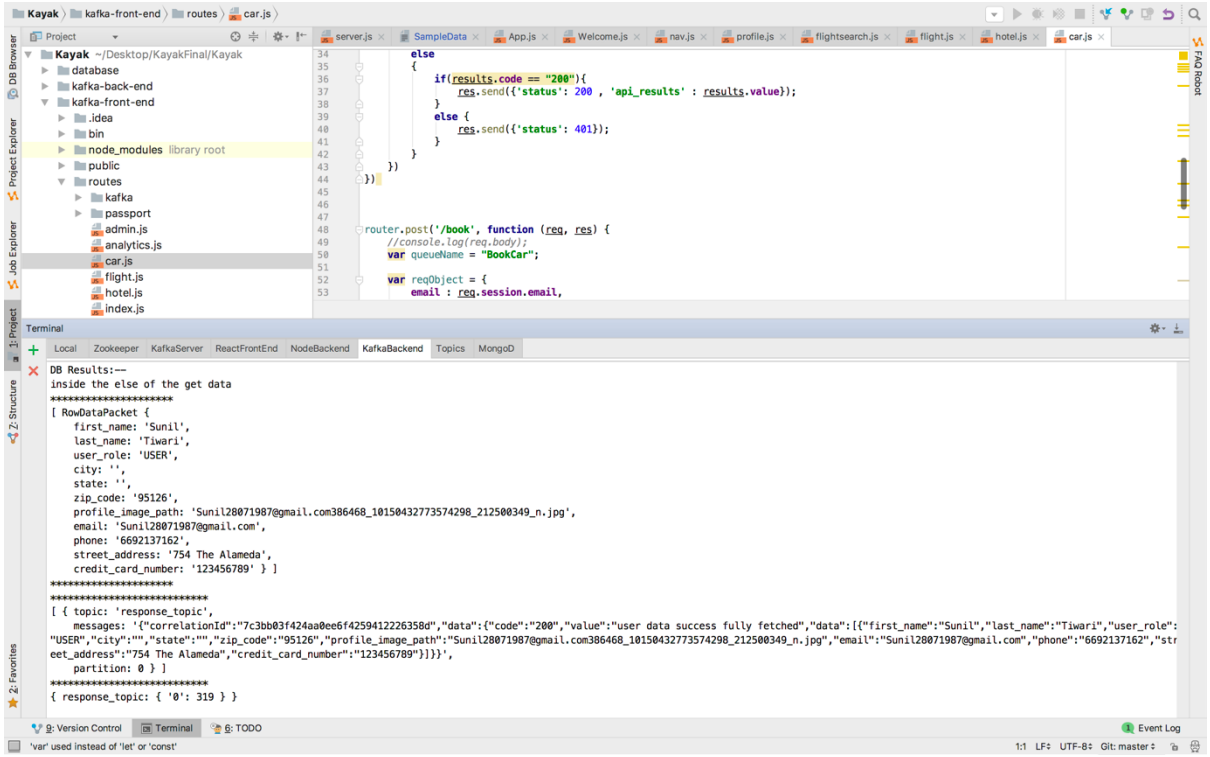
Sign out Page (UI):



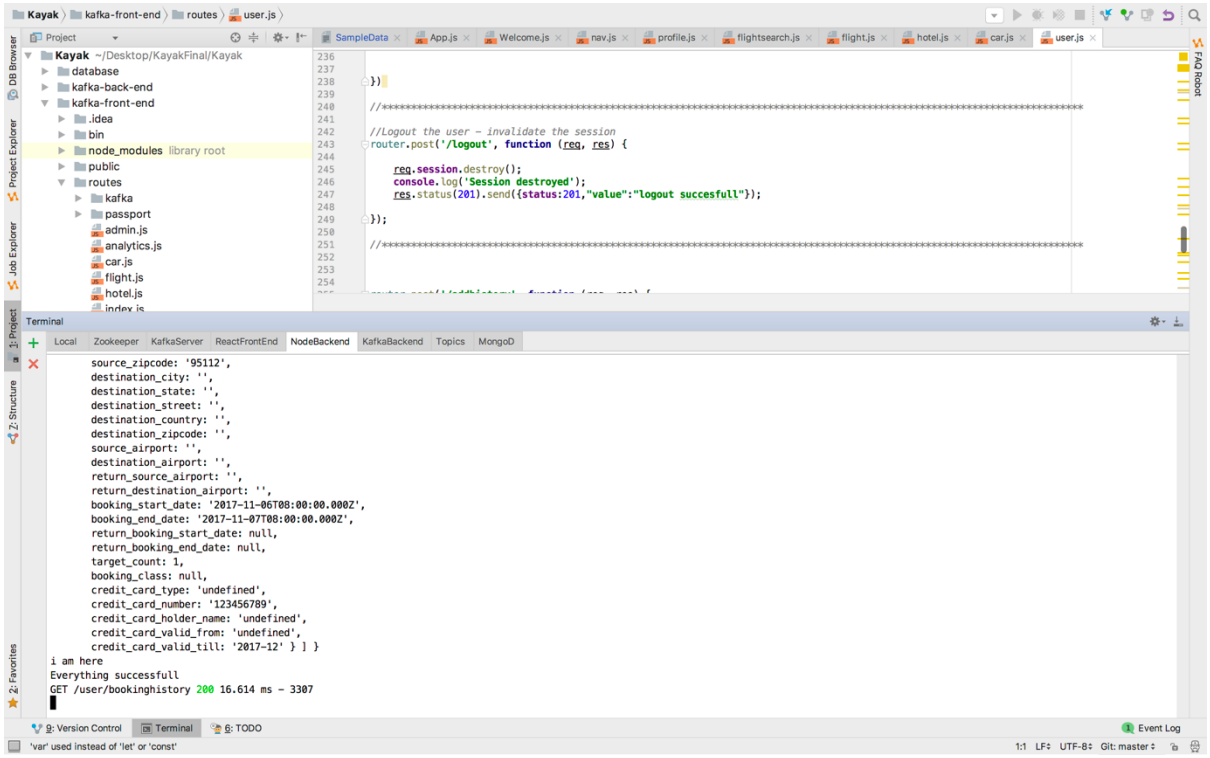
Sign out Page (Node Backend):



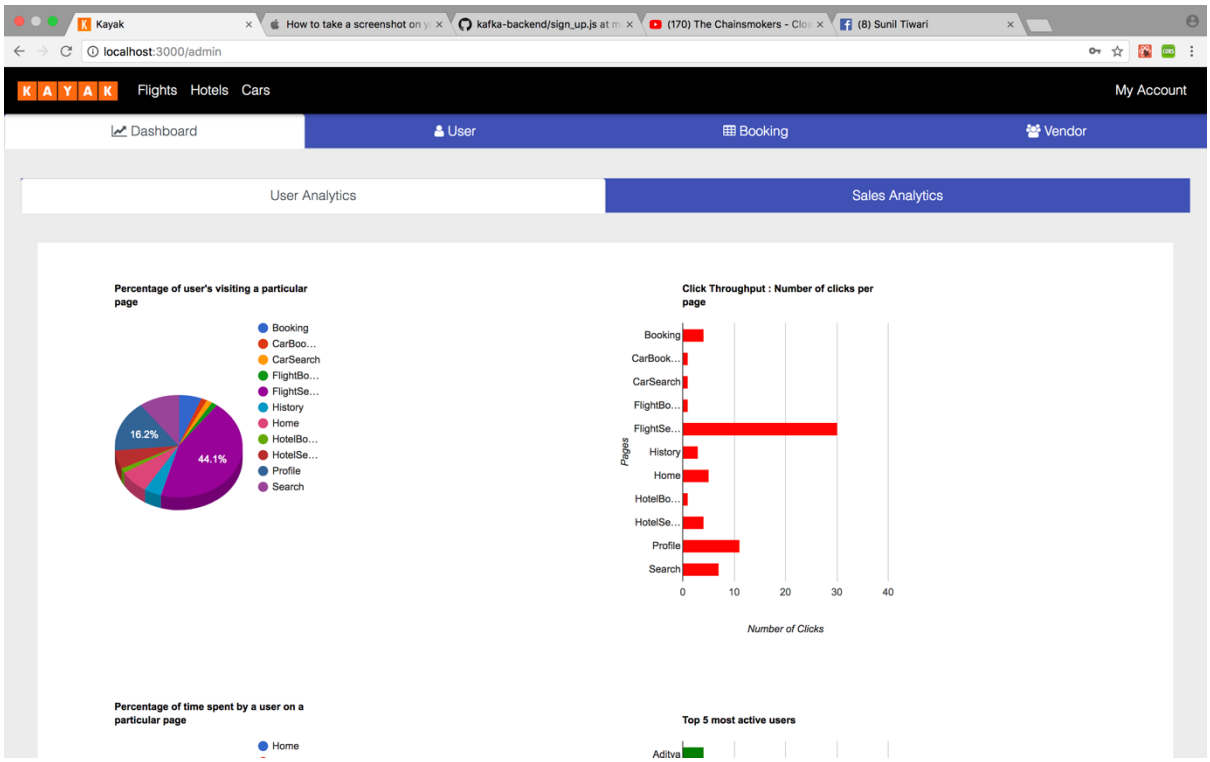
Sign out Page (Kafka Backend):

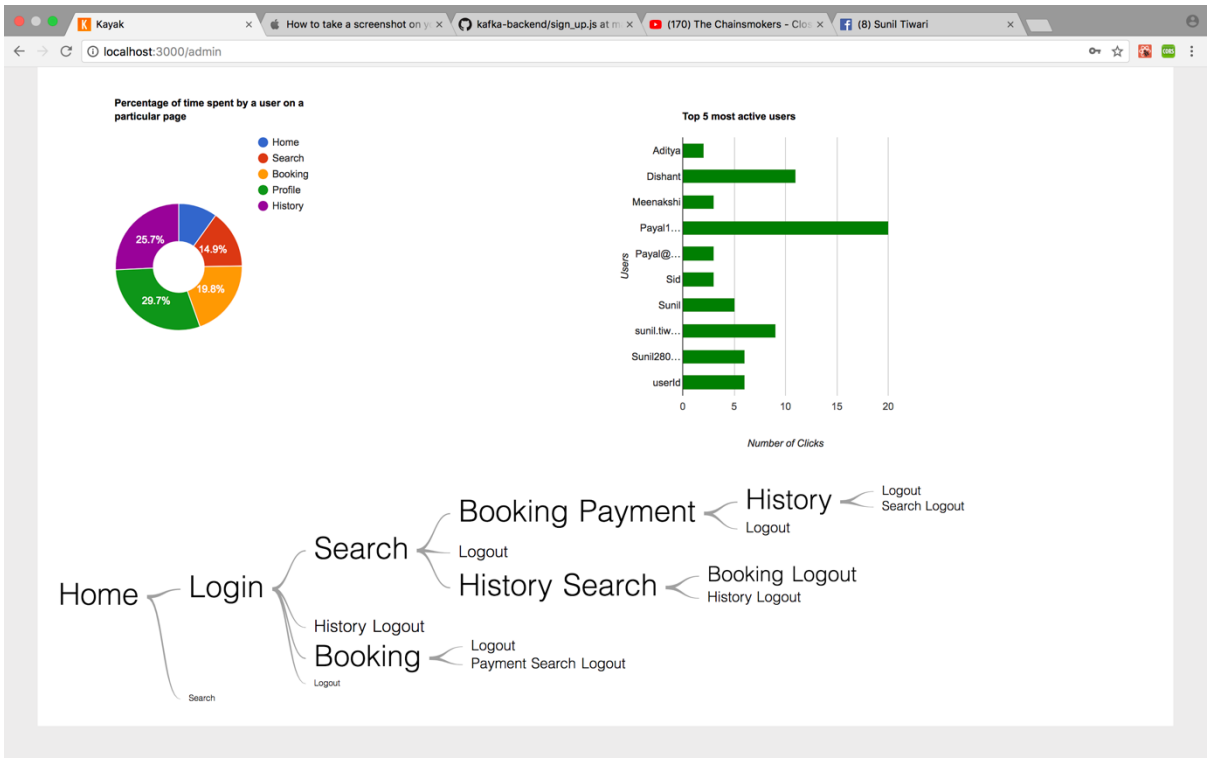


Sign out Page (Business Logic):

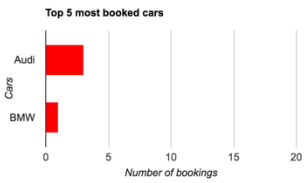


Admin Dashboard Page (UI):





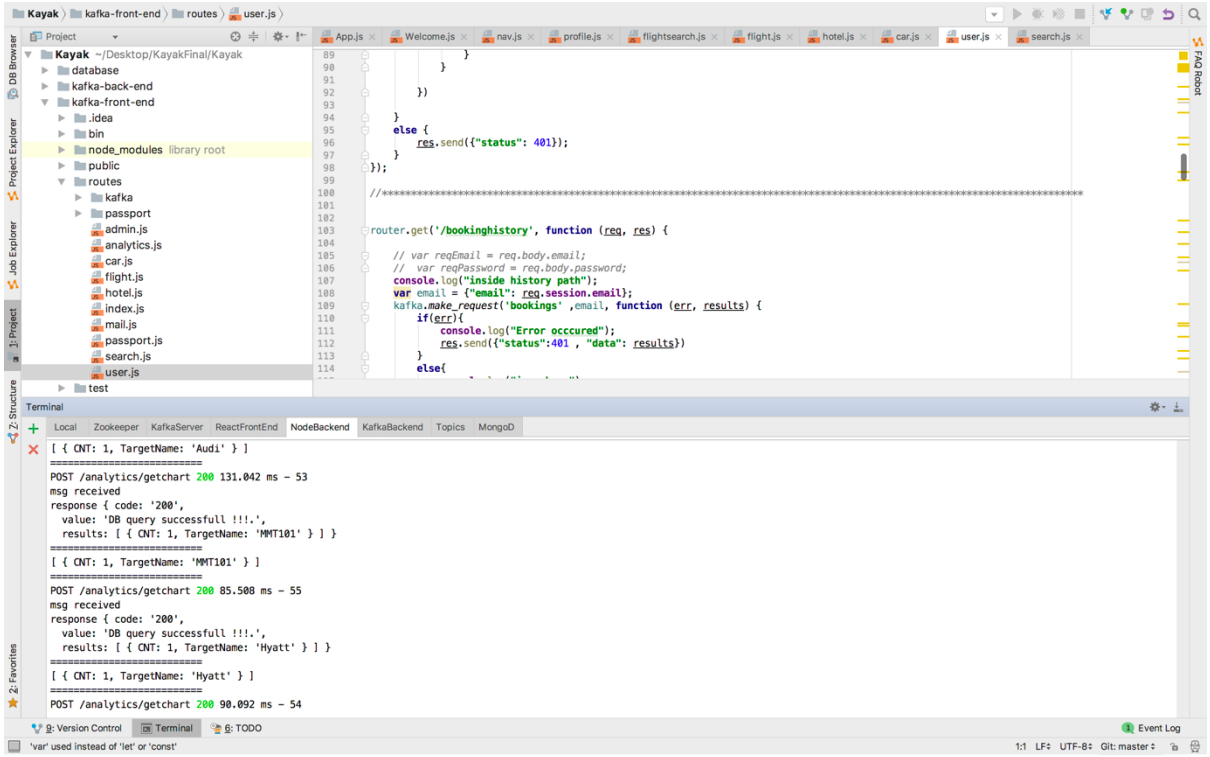

Top 5 most booked items
.emon Tree
ina Airlines
Holiday Inn
Audi
BMW
Items
0
5
10
15
20
Number of bookings


Top 5 most booked cars
Audi
BMW
Cars
0
5
10
15
20
Number of bookings

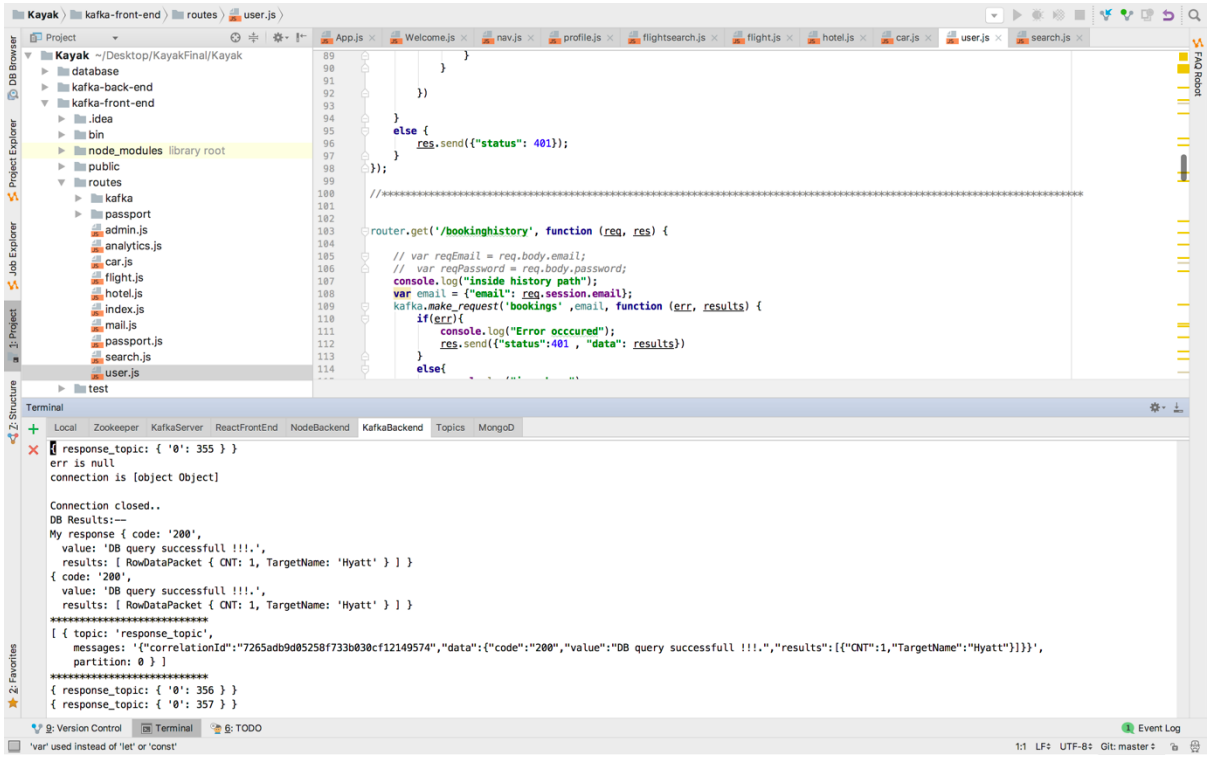

Top 5 most booked flights
MH101
Flights
0
5
10
15
20
Number of bookings


Top 5 most booked hotels
.emon Tree
Holiday Inn
Hotels
0
5
10
15
20
Number of bookings

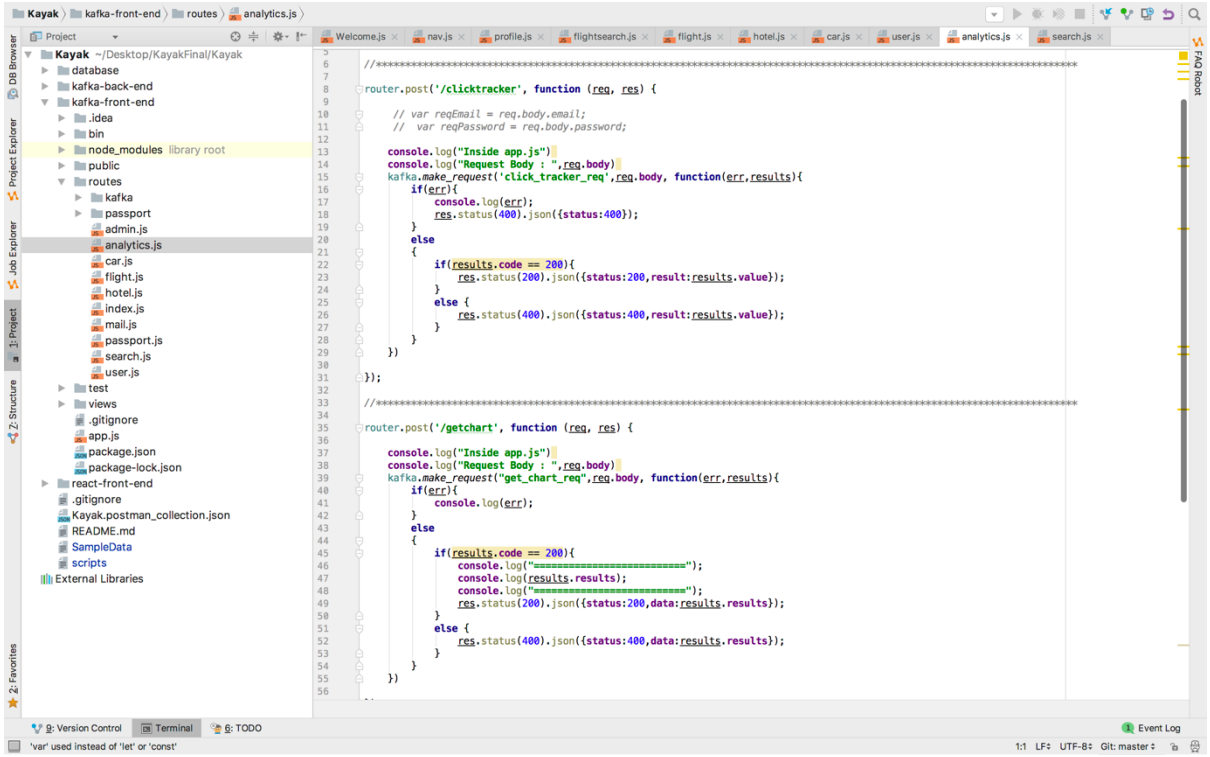
Admin Dashboard Page (Node Backend):



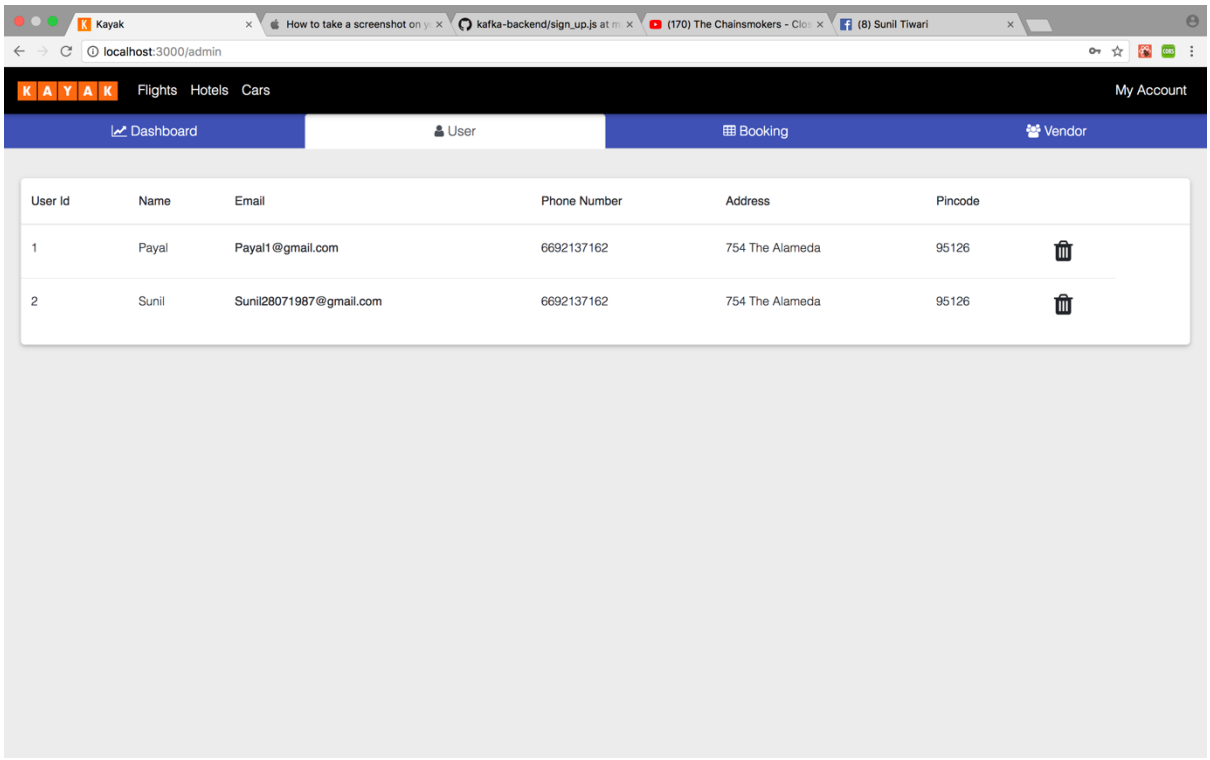
Admin Dashboard Page (Kafka Backend):



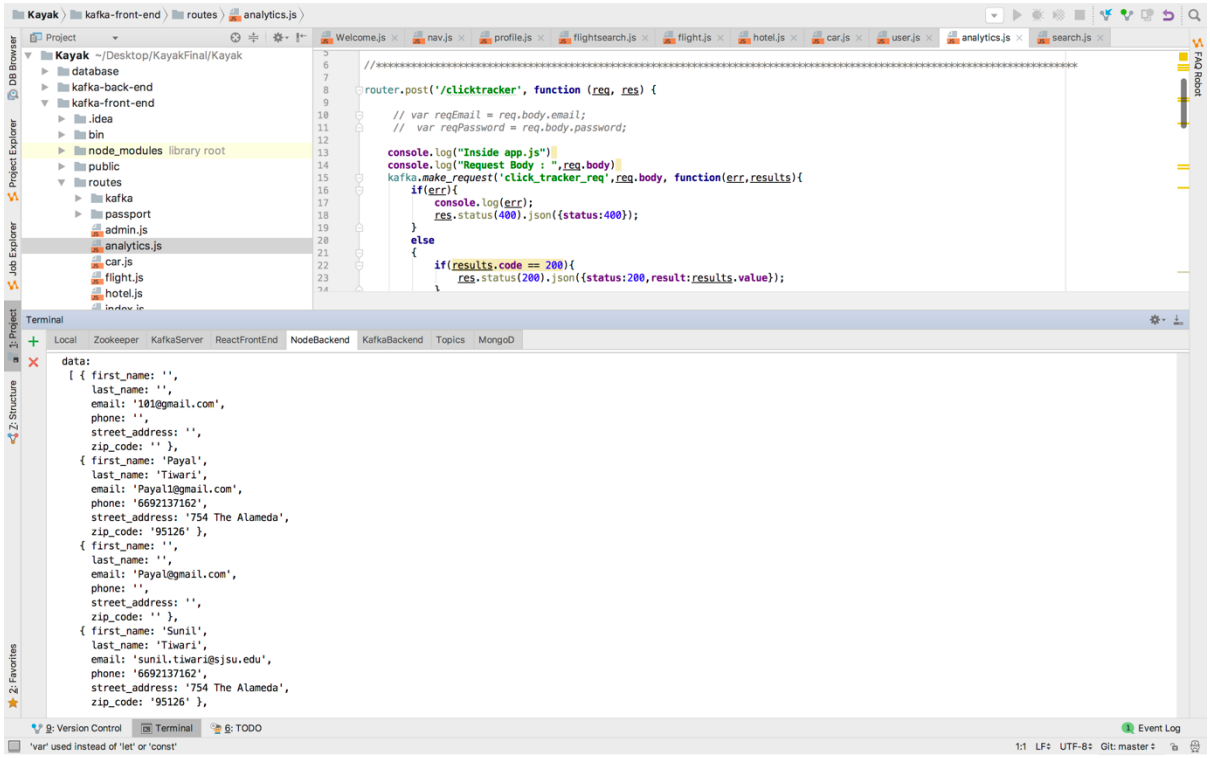
Admin Dashboard Page (Business Logic):



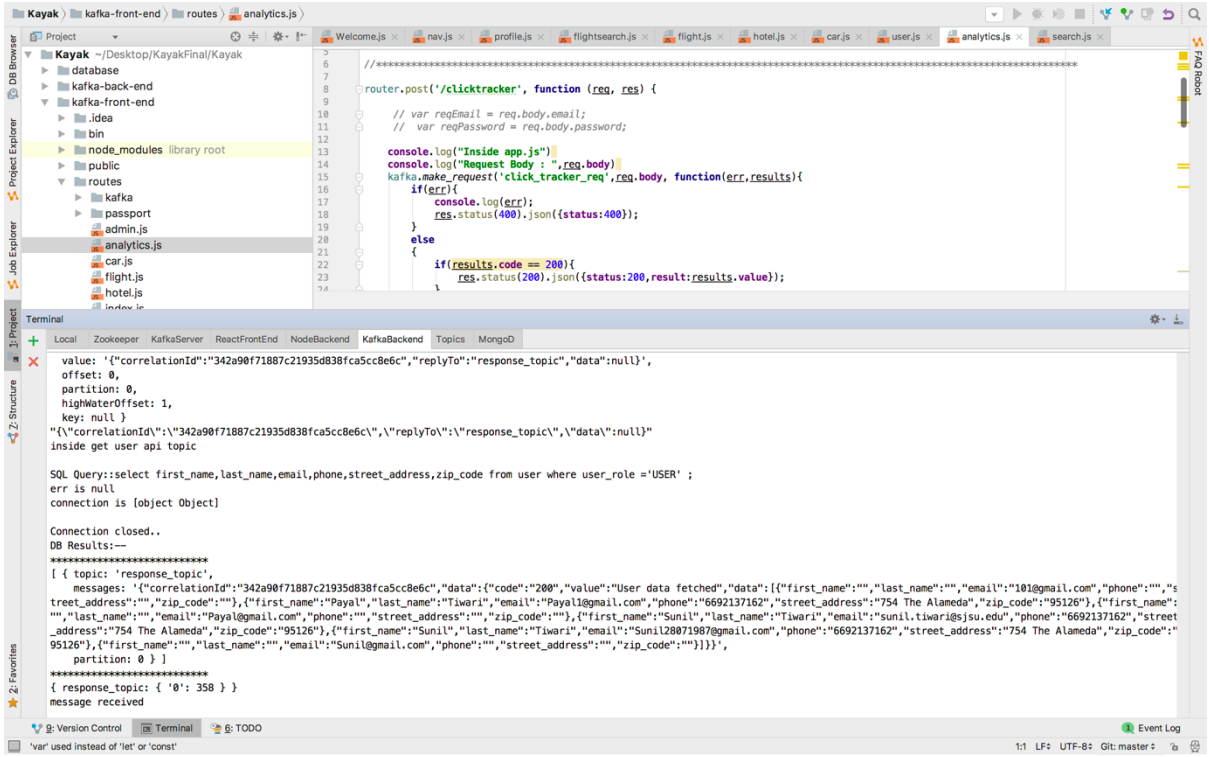
Admin User Page (UI):



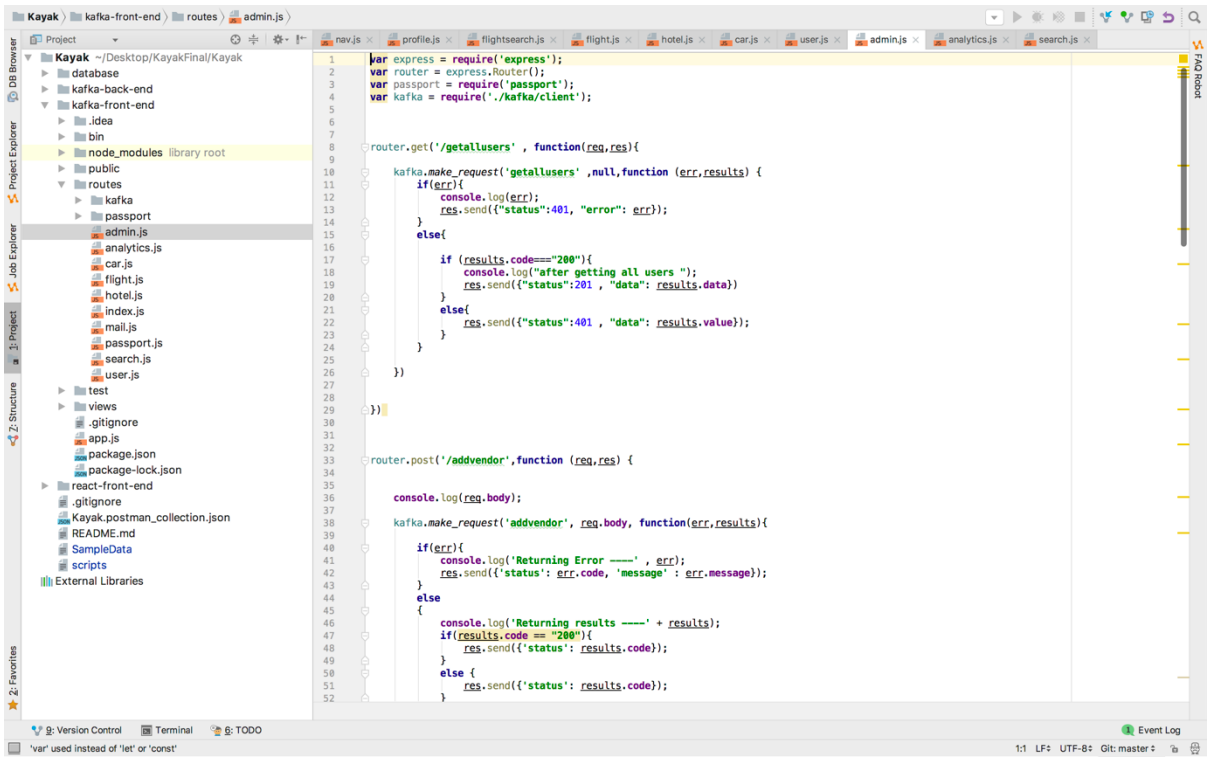
Admin User Page (Node Backend):



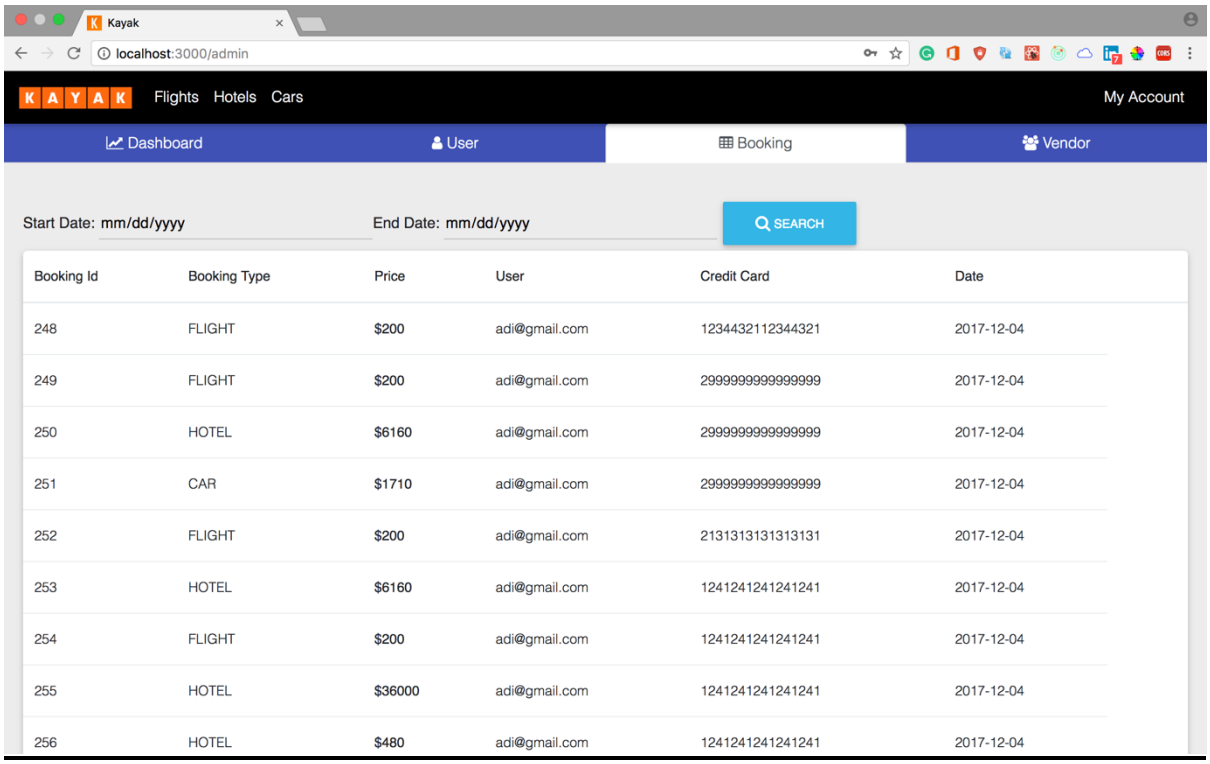
Admin User Page (Kafka Backend):



Admin User Page (Business Logic):

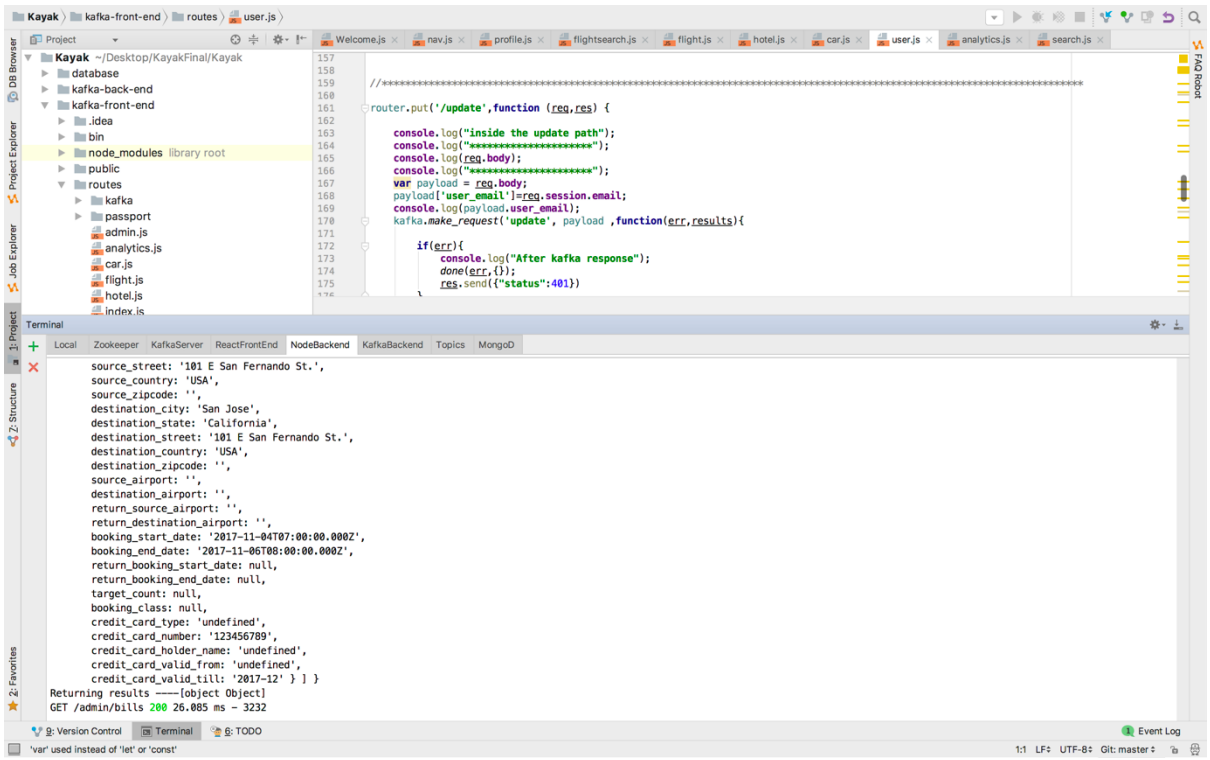


Admin Bookings Page (UI):

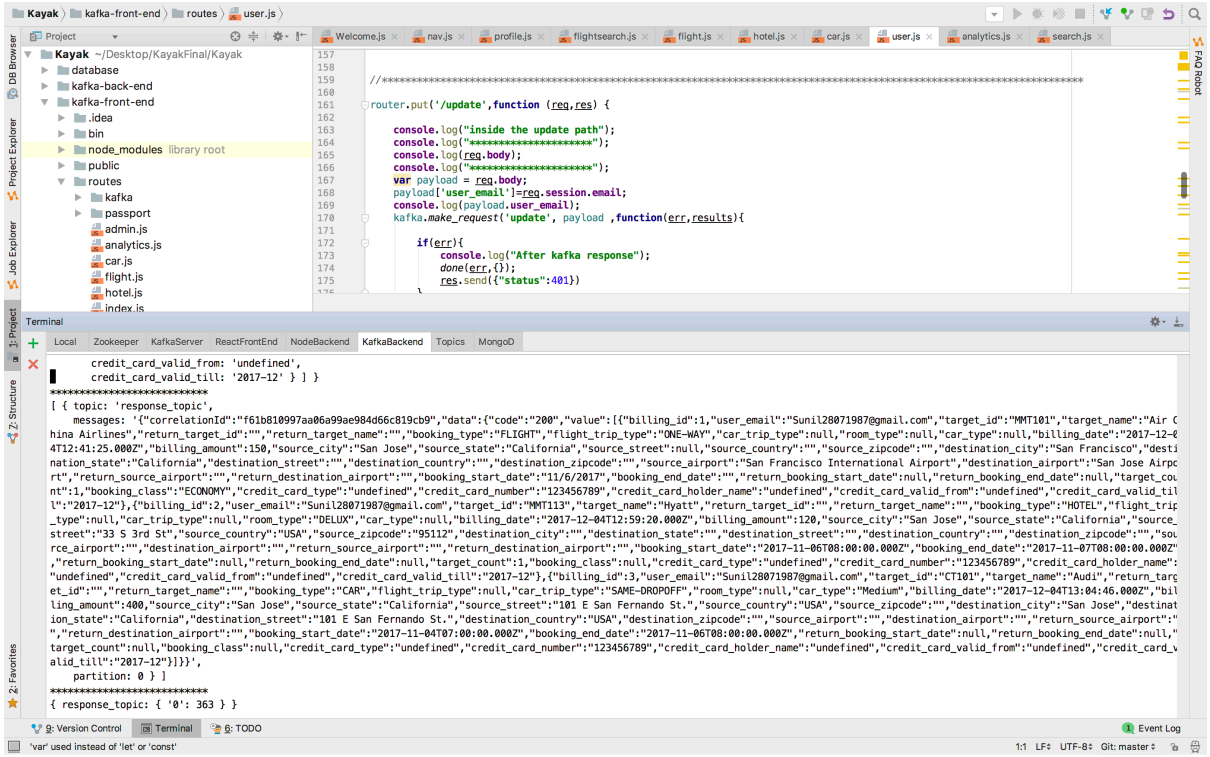


Booking Id	Booking Type	Price	User	Credit Card	Date
248	FLIGHT	$200	adi@gmail.com	1234432112344321	2017-12-04
249	FLIGHT	$200	adi@gmail.com	2999999999999999	2017-12-04
250	HOTEL	$6160	adi@gmail.com	2999999999999999	2017-12-04
251	CAR	$1710	adi@gmail.com	2999999999999999	2017-12-04
252	FLIGHT	$200	adi@gmail.com	2131313131313131	2017-12-04
253	HOTEL	$6160	adi@gmail.com	1241241241241241	2017-12-04
254	FLIGHT	$200	adi@gmail.com	1241241241241241	2017-12-04
255	HOTEL	$36000	adi@gmail.com	1241241241241241	2017-12-04
256	HOTEL	$480	adi@gmail.com	1241241241241241	2017-12-04

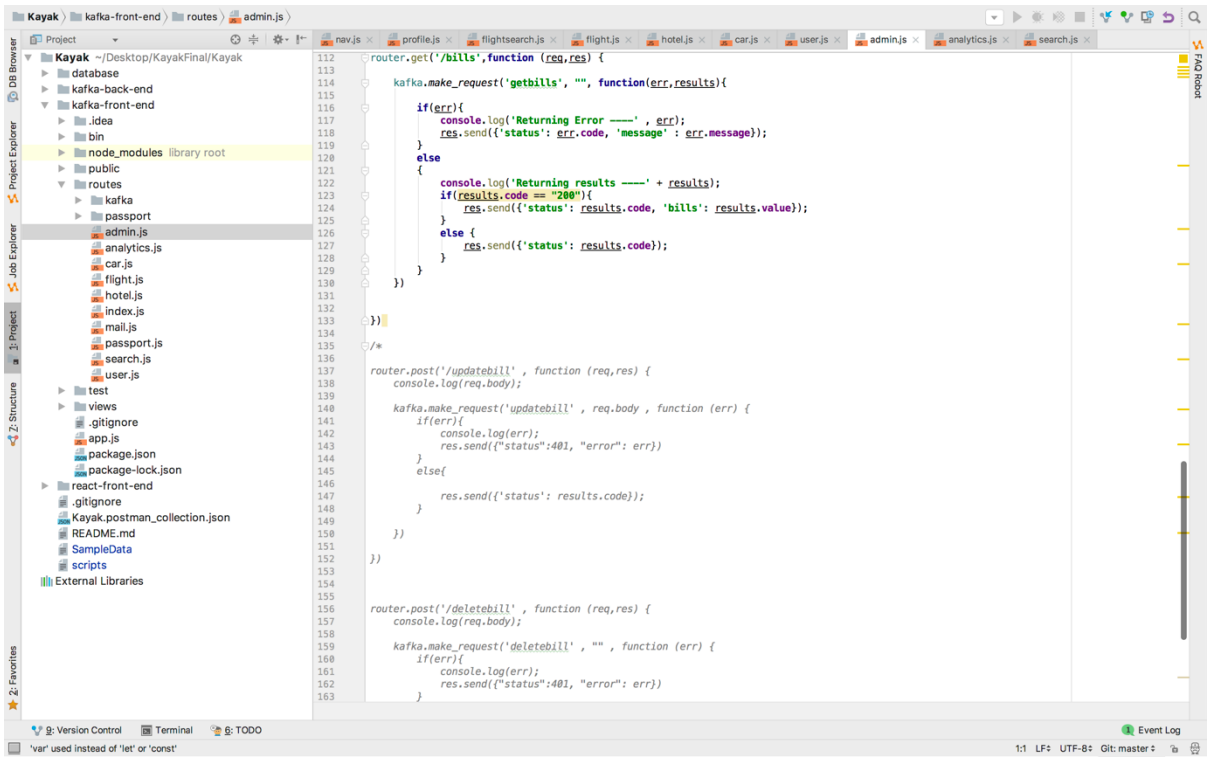
Admin Bookings Page (Node Backend):



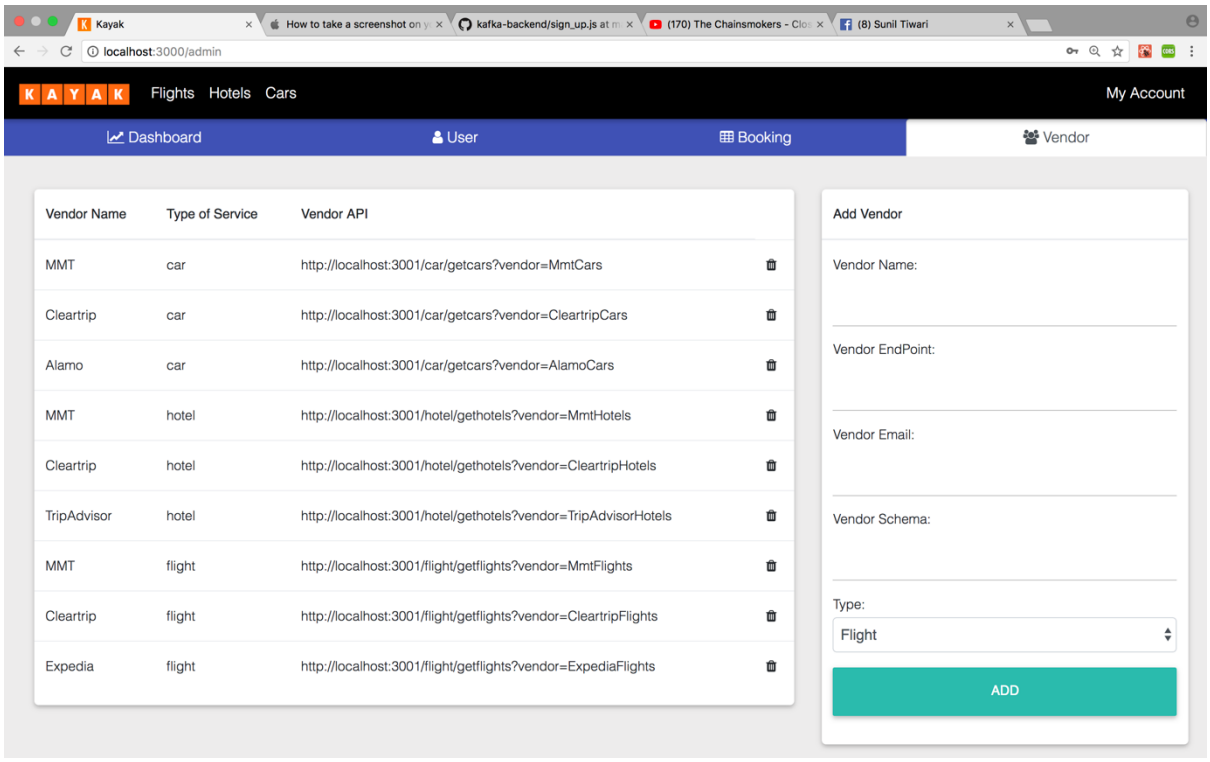
Admin Bookings Page (Kafka Backend):

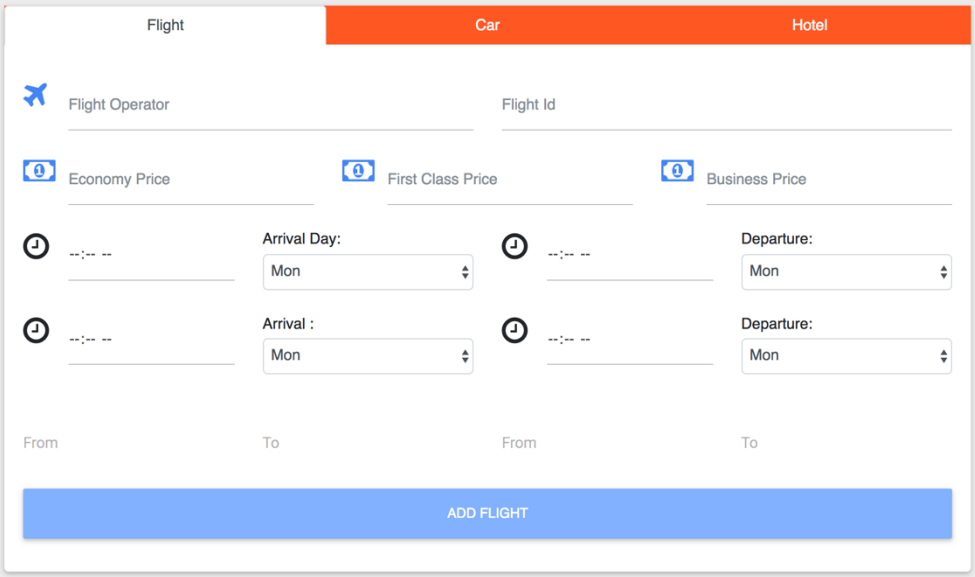


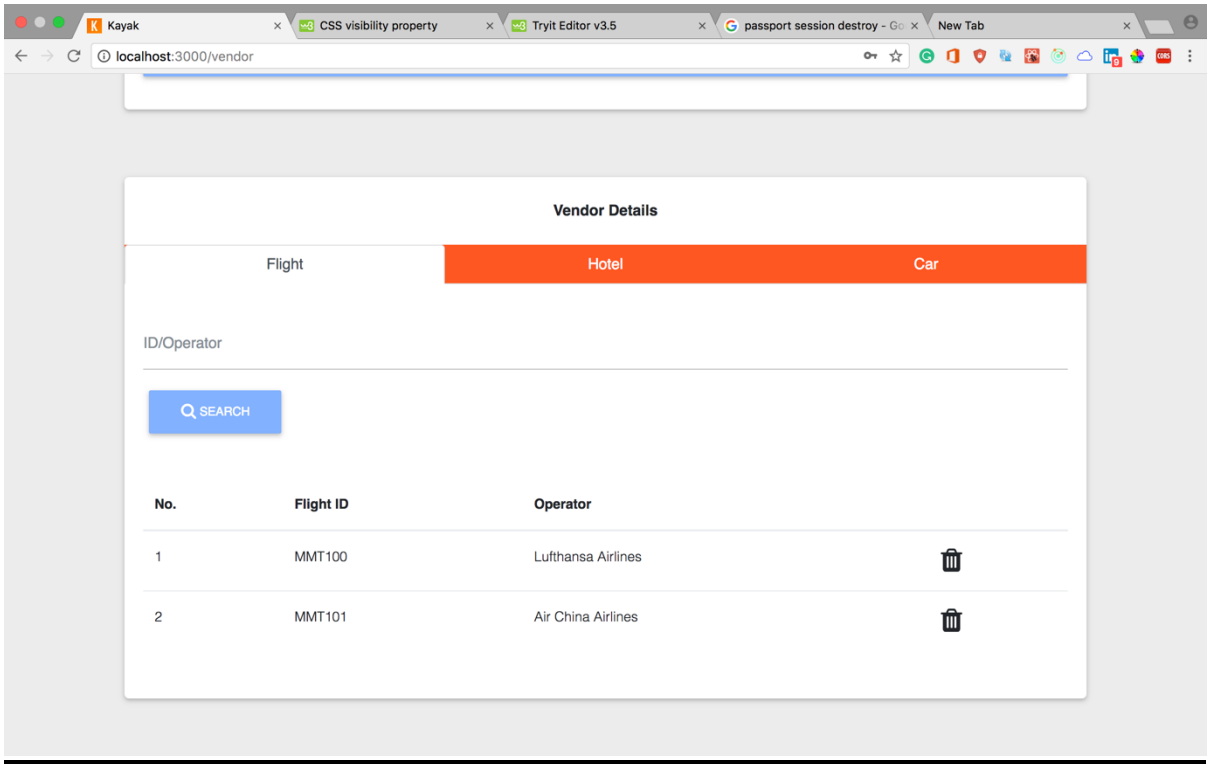
Admin Bookings Page (Business Logic):



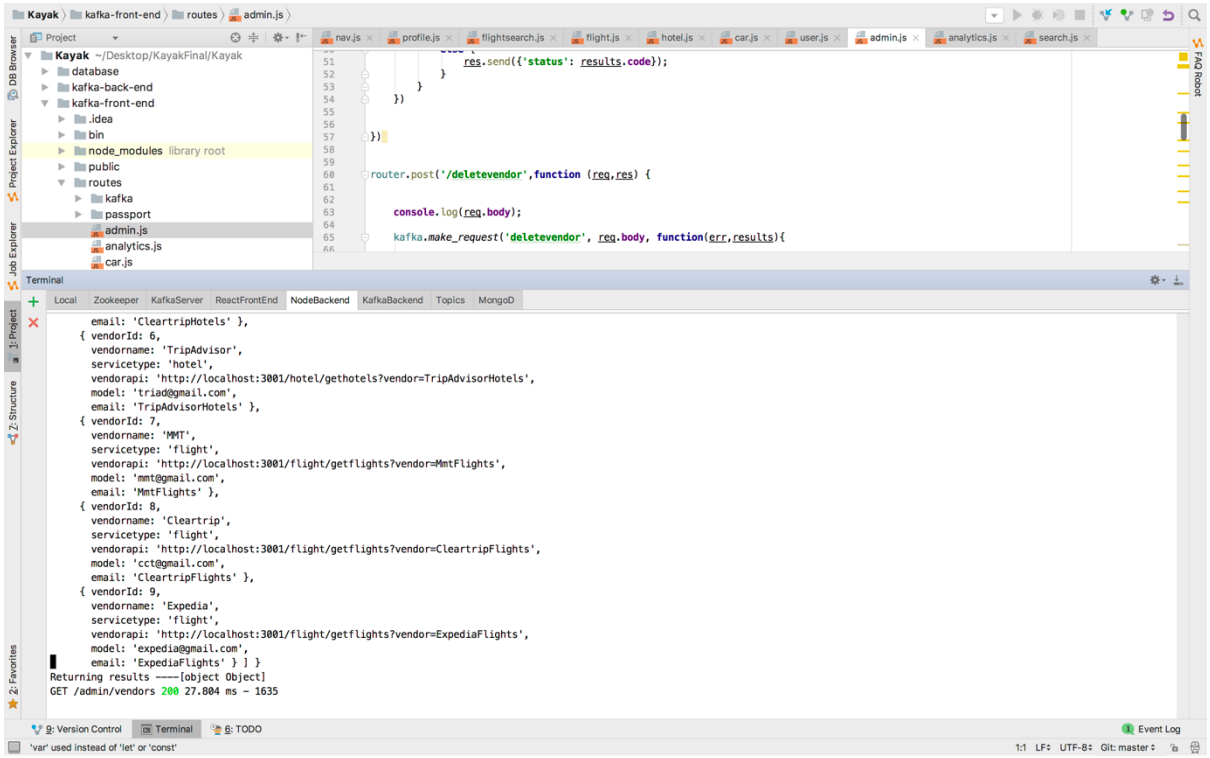
Admin Vendor Page (UI):



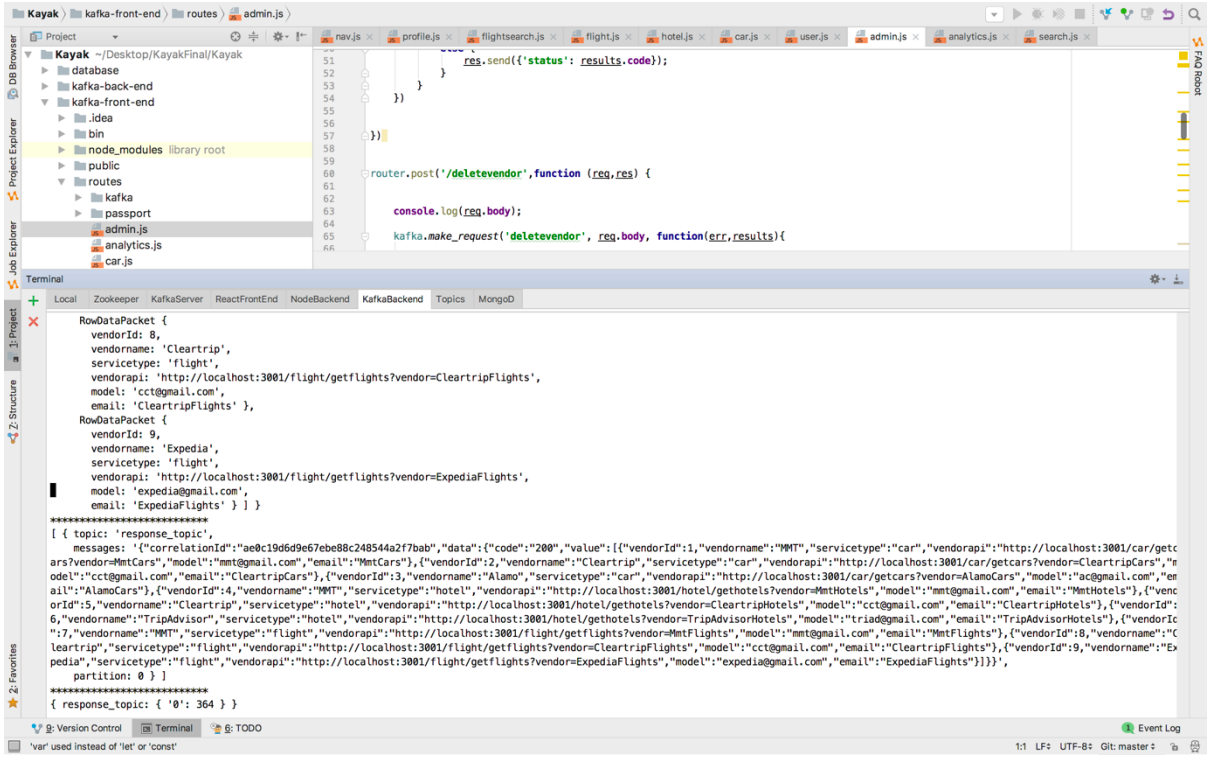




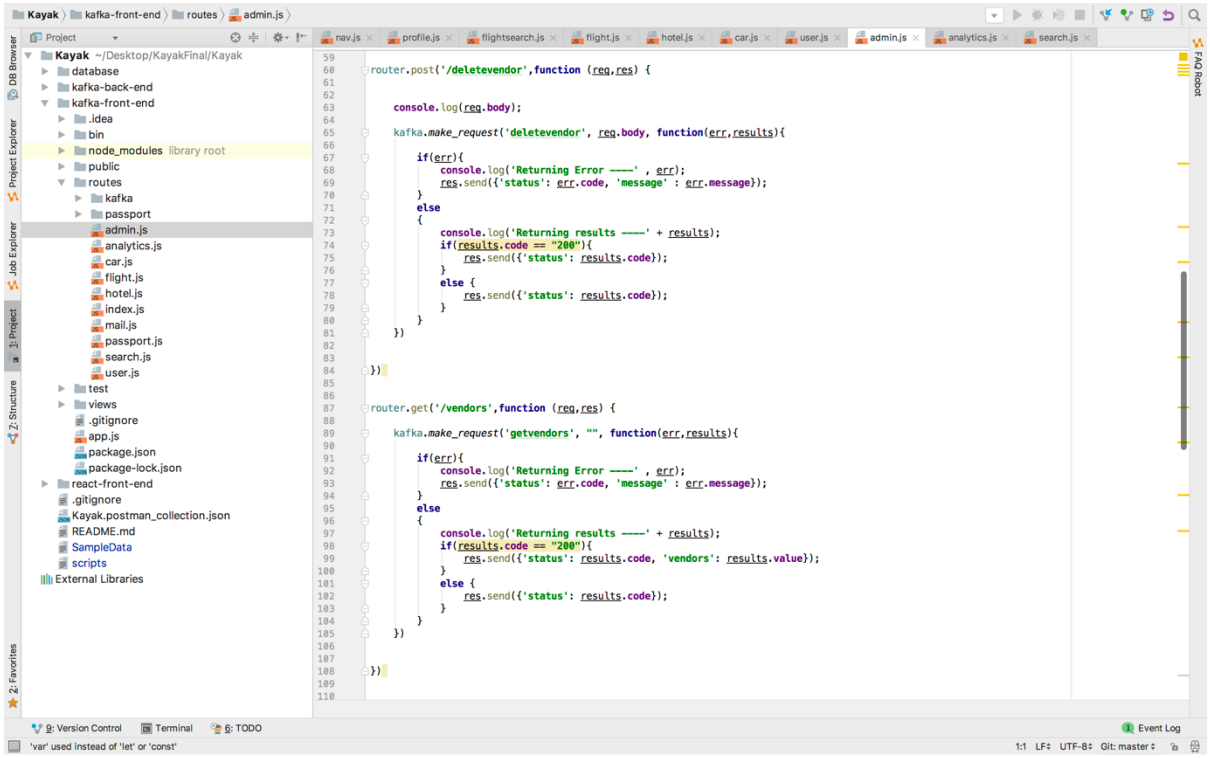
Admin Vendor Page (Node Backend):



Admin Vendor Page (Kafka Backend):



Admin Vendor Page (Business Logic):



Performance (Test Cases):

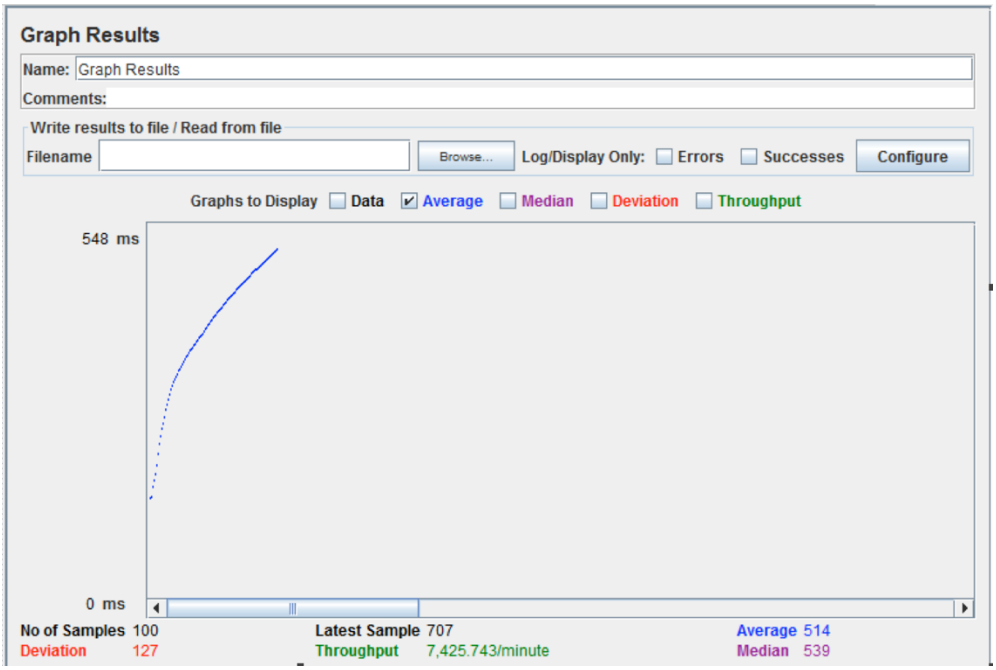
Performance with Kafka queueing is excellent. Kafka is a distributed and parallel processing architecture provides seamless asynchronous API handling experience to provide a faster performance.

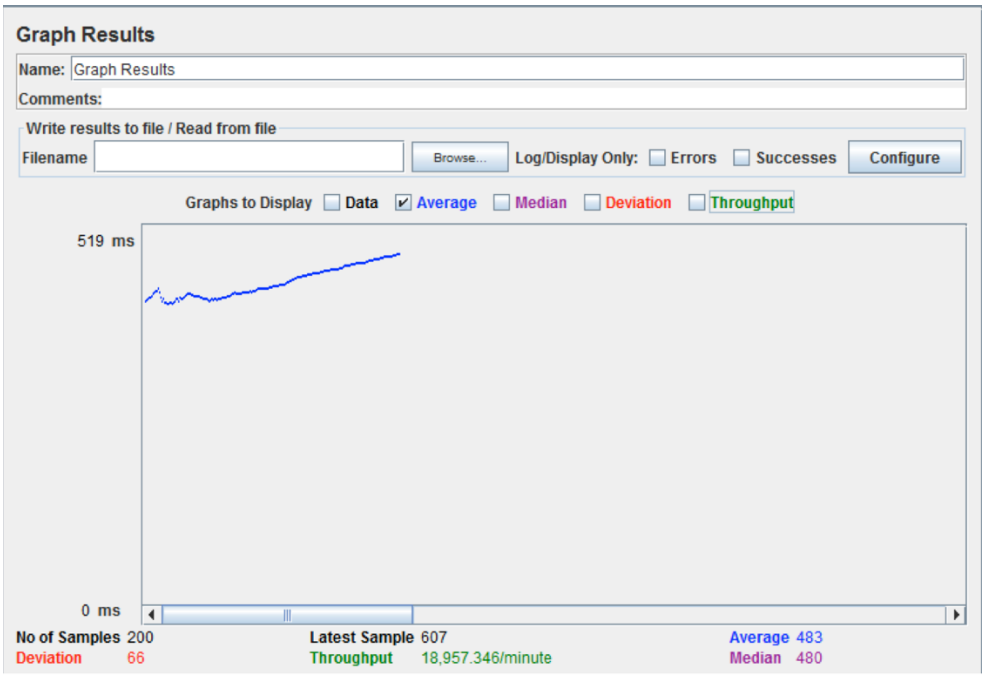
Analysis:
Kafka's performances analysis as per the following testing tells us that Kafka is good at handling large number of requests. Initially, when the requests increase time for service increases, but soon as the request become as large as thousands, it stabilizes. This tells us that our architecture is designed to handle large number of requests with same throughput.

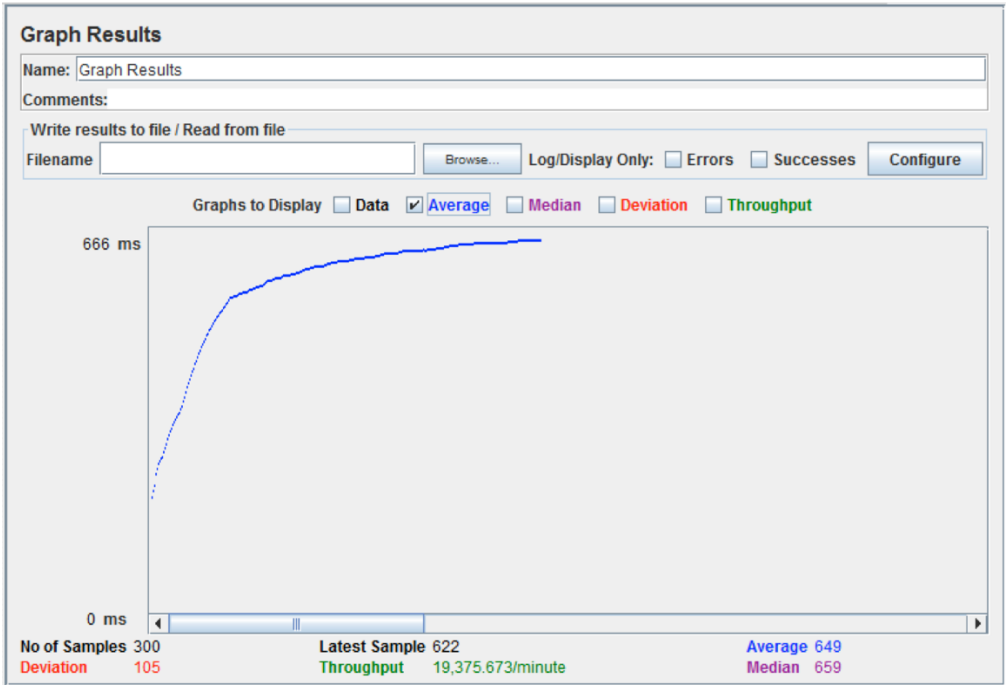
We could have achieved better performance by

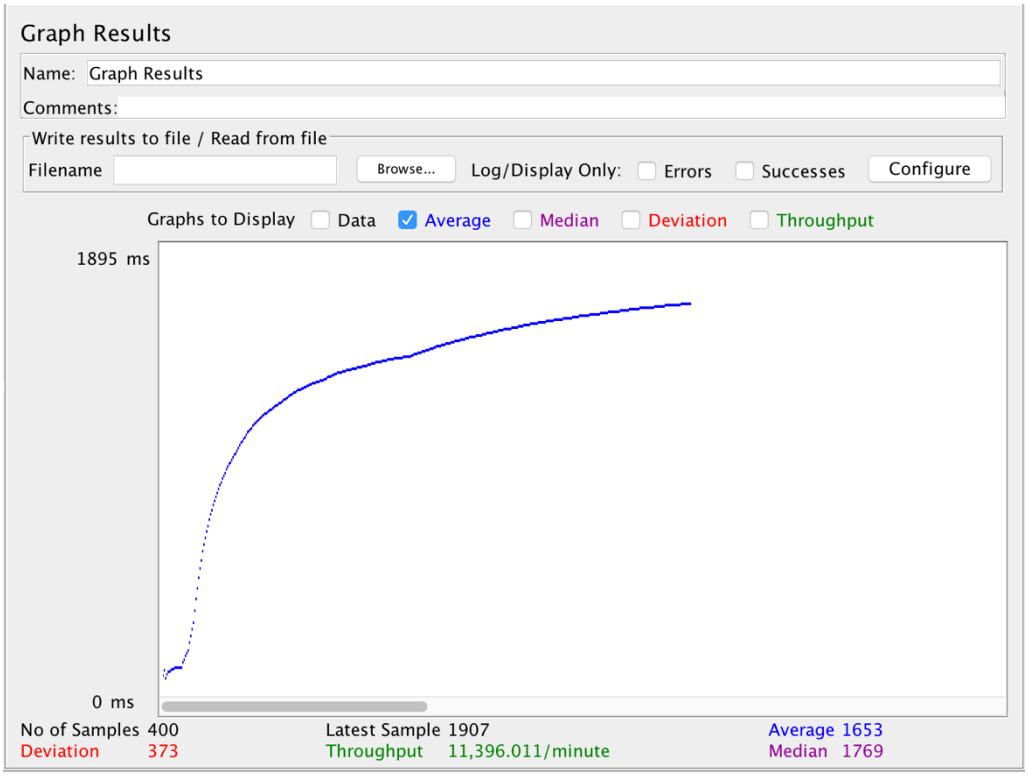
- More number of producer partitions
- Increased number of consumers in a consumer group

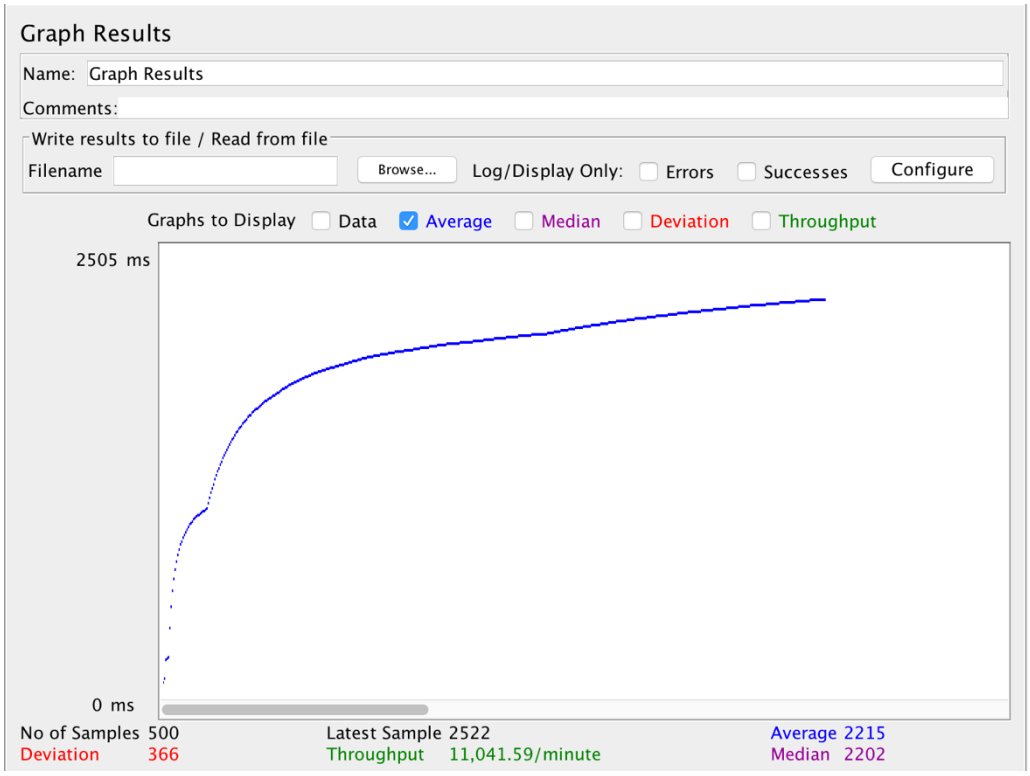
JMeter Test Case



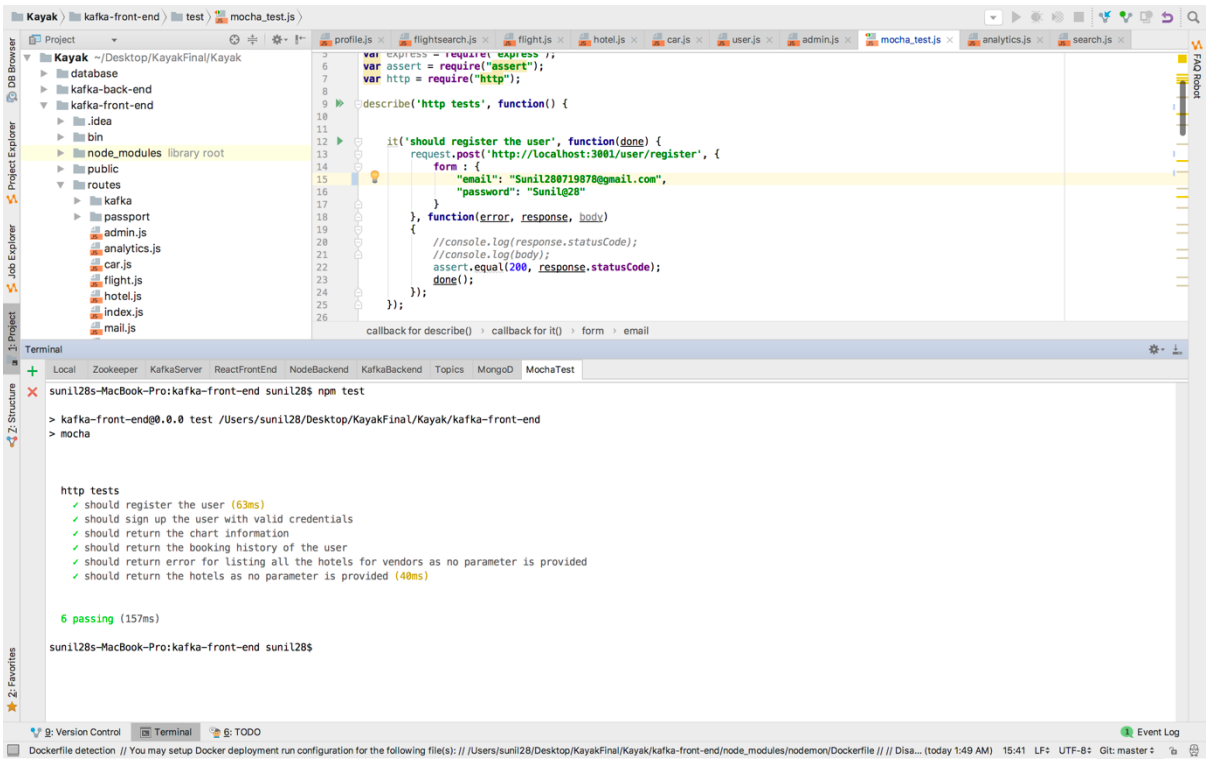



Graph Results
Name: Graph Results
Comments:
Write results to file / Read from file
Filename
Browse...
Log/Display Only: Errors Successes
Configure
Graphs to Display Data Average Median Deviation Throughput
666 ms
0 ms
No of Samples 300
Latest Sample 622
Average 649
Deviation 105
Throughput 19,375.673/minute
Median 659


Graph Results
Name: Graph Results
Comments:
Write results to file / Read from file
Filename
Browse...
Log/Display Only: Errors Successes
Configure
Graphs to Display Data Average Median Deviation Throughput
1895 ms
0 ms
No of Samples 400
Latest Sample 1907
Average 1653
Deviation 373
Throughput 11,396.011/minute
Median 1769

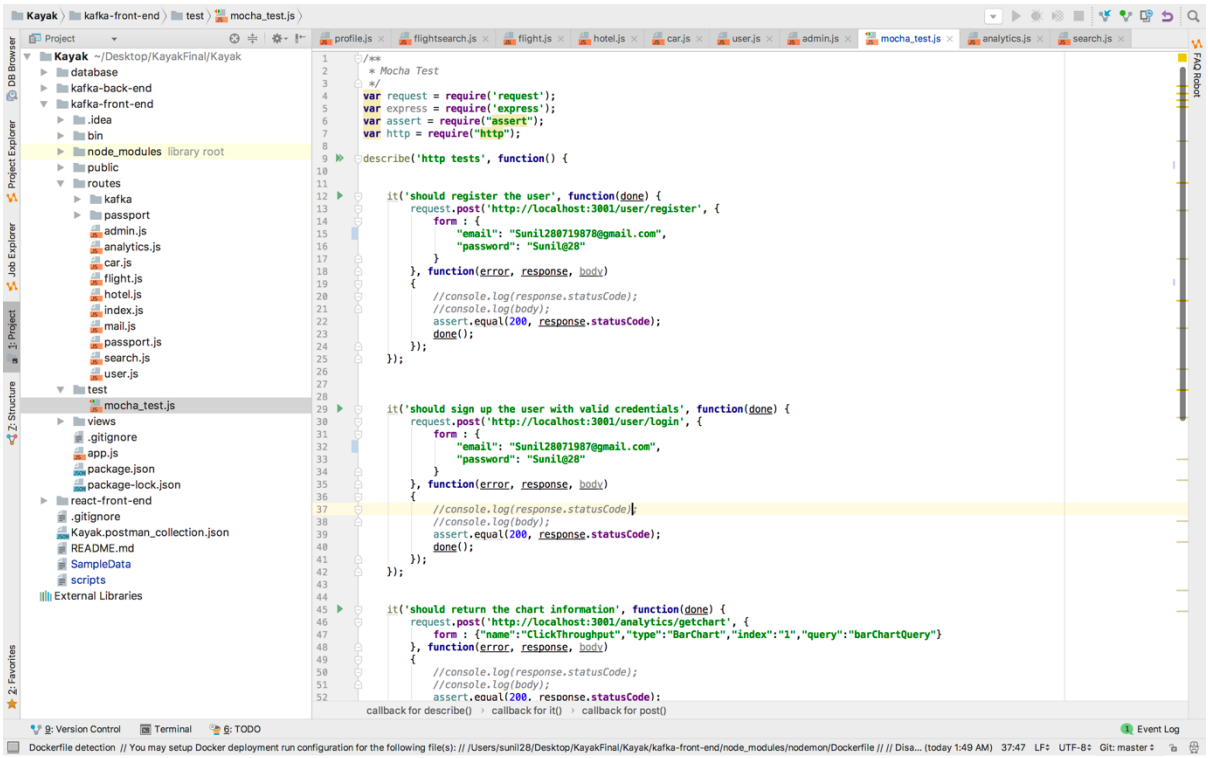

Graph Results
Name: Graph Results
Comments:
Write results to file / Read from file
Filename
Browse...
Log/Display Only:
Errors
Successes
Configure
Graphs to Display
Data
Average
Median
Deviation
Throughput
2505 ms
0 ms
No of Samples 500
Latest Sample 2522
Average 2215
Deviation 366
Throughput 11,041.59/minute
Median 2202

Mocha Test Cases

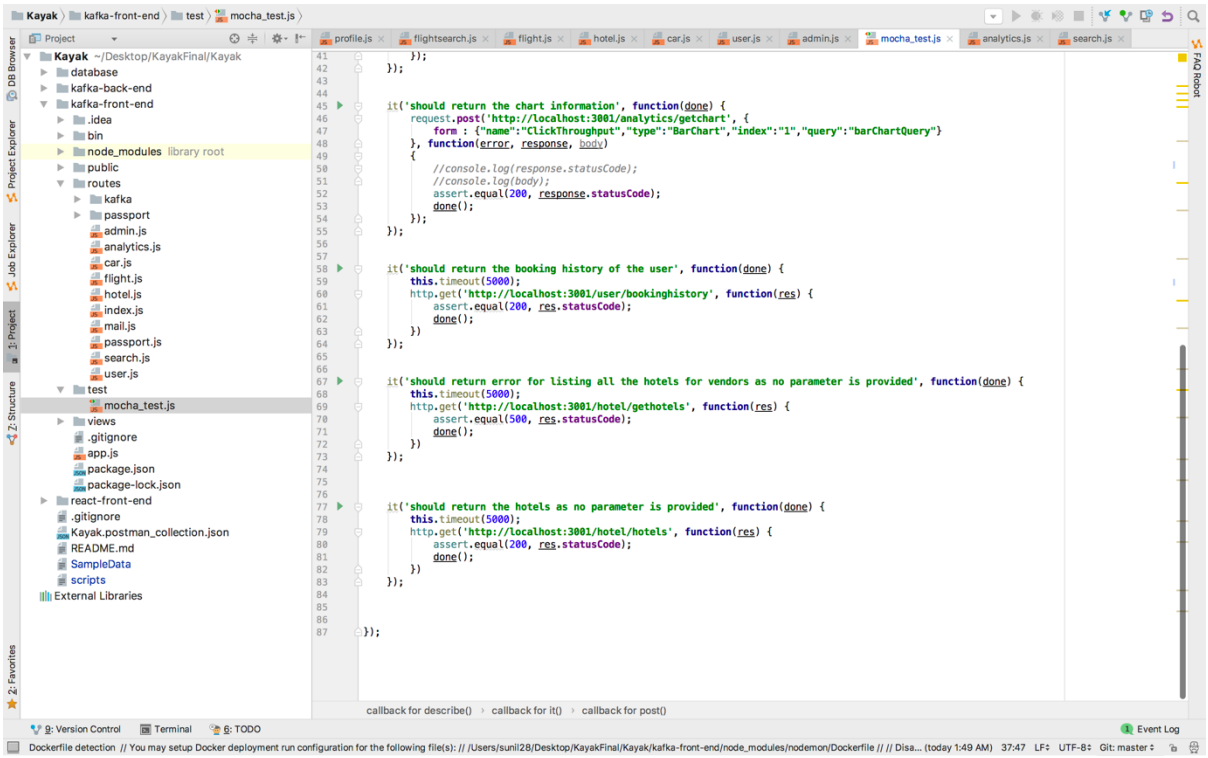


```
var express = require('express');
var assert = require("assert");
var http = require("http");

describe('http tests', function() {


    it('should register the user', function(done) {
        request.post('http://localhost:3001/user/register', {
            form : {
                "email": "Sunil280719878@gmail.com",
                "password": "Sunil@28"
            }
        }, function(error, response, body)
        {
            //console.log(response.statusCode);
            //console.log(body);
            assert.equal(200, response.statusCode);
            done();
        });
    });
```

```
sunil28s-MacBook-Pro:kafka-front-end sunil28$ npm test

> kafka-front-end@0.0.0 test /Users/sunil28/Desktop/KayakFinal/Kayak/kafka-front-end
> mocha


  http tests
    ✓ should register the user (63ms)
    ✓ should sign up the user with valid credentials
    ✓ should return the chart information
    ✓ should return the booking history of the user
    ✓ should return error for listing all the hotels for vendors as no parameter is provided
    ✓ should return the hotels as no parameter is provided (40ms)


  6 passing (157ms)

sunil28s-MacBook-Pro:kafka-front-end sunil28$
```



```
/**
 * Mocha Test
 */
var request = require('request');
var express = require('express');
var assert = require("assert");
var http = require("http");

describe('http tests', function() {


    it('should register the user', function(done) {
        request.post('http://localhost:3001/user/register', {
            form : {
                "email": "Sunil280719878@gmail.com",
                "password": "Sunil@28"
            }
        }, function(error, response, body)
        {
            //console.log(response.statusCode);
            //console.log(body);
            assert.equal(200, response.statusCode);
            done();
        });
    });


    it('should sign up the user with valid credentials', function(done) {
        request.post('http://localhost:3001/user/login', {
            form : {
                "email": "Sunil28071987@gmail.com",
                "password": "Sunil@28"
            }
        }, function(error, response, body)
        {
            //console.log(response.statusCode);
            //console.log(body);
            assert.equal(200, response.statusCode);
            done();
        });
    });


    it('should return the chart information', function(done) {
        request.post('http://localhost:3001/analytics/getchart', {
            form : {"name":"ClickThroughput","type":"BarChart","index":"1","query":"barChartQuery"}
        }, function(error, response, body)
        {
            //console.log(response.statusCode);
            //console.log(body);
            assert.equal(200, response.statusCode);
```


Kayak
kafka-front-end
test
mocha_test.js
Project
Kayak ~/Desktop/KayakFinal/Kayak
database
kafka-back-end
kafka-front-end
.idea
bin
node_modules library root
public
routes
kafka
passport
admin.js
analytics.js
car.js
flight.js
hotel.js
index.js
mail.js
passport.js
search.js
user.js
test
mocha_test.js
views
.gitignore
app.js
package.json
package-lock.json
react-front-end
.gitignore
Kayak.postman_collection.json
README.md
SampleData
scripts
External Libraries
profile.js
flightsearch.js
flight.js
hotel.js
car.js
user.js
admin.js
mocha_test.js
analytics.js
search.js
});
});
it('should return the chart information', function(done) {
request.post('http://localhost:3001/analytics/getchart', {
form : {"name":"ClickThroughput","type":"BarChart","index":"1","query":"barChartQuery"}
}, function(error, response, body)
{
//console.log(response.statusCode);
//console.log(body);
assert.equal(200, response.statusCode);
done();
});
});
it('should return the booking history of the user', function(done) {
this.timeout(5000);
http.get('http://localhost:3001/user/bookinghistory', function(res) {
assert.equal(200, res.statusCode);
done();
})
});
it('should return error for listing all the hotels for vendors as no parameter is provided', function(done) {
this.timeout(5000);
http.get('http://localhost:3001/hotel/gethotels', function(res) {
assert.equal(500, res.statusCode);
done();
})
});
it('should return the hotels as no parameter is provided', function(done) {
this.timeout(5000);
http.get('http://localhost:3001/hotel/hotels', function(res) {
assert.equal(200, res.statusCode);
done();
})
});
});
callback for describe() › callback for it() › callback for post()
9: Version Control
Terminal
6: TODO
Event Log
Dockerfile detection // You may setup Docker deployment run configuration for the following file(s): // /Users/sunil28/Desktop/KayakFinal/Kayak/kafka-front-end/node_modules/nodemon/Dockerfile // // Disa... (today 1:49 AM) 37:47 LF UTF-8 Git: master

Observations & Lessons:

It was first time for the entire team when we created an entire full stack end to end application. Kayak was a large system and we learned following lessons.

1. Designing user interface for a large system like Kayak.
2. Session & State management for large system like Kayak.
3. Designing a large system with various small sub components (React, Redux, Database, Kafka, API's) and handling interaction among them.
4. Creating and managing large number of API calls in NodeJS.
5. Creating, sending and receiving and handling distributed messaging between clients and server.
6. Handling secure user authentication using various strategies with the help of PassportJS libraries.
7. Testing the REST API's using various testing tools such as JMeter and MochaJS.
8. Caching SQL queries to improve the performance and throughput of the system.
9. Tracking the user activity through the UI and create a logging mechanism to save all that information in the database.
10. Drawing the meaningful insights from the logged information and make decisions based on user actions.