

Lab 2 Report

Name: Darshil Kapadia

013007280

Git-Repo:

<https://github.com/Anuragis/CMPE273-36.git>

HomeAway using NodeJS, ReactJS and Kafka

Introduction:

Goals of the system:

- The goal of this system is to widen the understanding of ReactJS and NodeJS. Along with that, its goal is to help understand the state management in react using Redux and to develop enterprise-wide applications which takes care of various operations like managing the information of users, managing the information of owners, managing the information of properties listed by different owners and also taking care of booking done by different travelers.
- In order to develop this application, we will get a wide understanding of different concepts of ReactJS like props, states, Redirect features, form validations and many more.
- State Management, quite important while working with ReactJS, will be done using the help of Redux and in turn learning those Redux's concept of Store, Actions and Reducers.
- This application will also help us in widening the backend i.e NodeJS concepts because of different operations like communication with MongoDB, managing session, encrypting passwords using bcrypt, protecting routes using JWT and Passport, uploading images and many more.
- This application will also help us in getting hands on experience on one of the most useful and emerging technology called Kafka. Kafka basically combines 2 features called "Publish Subscribe and Messaging Queues". We will learn basic communication over here using Kafka Services.

Purpose of the system:

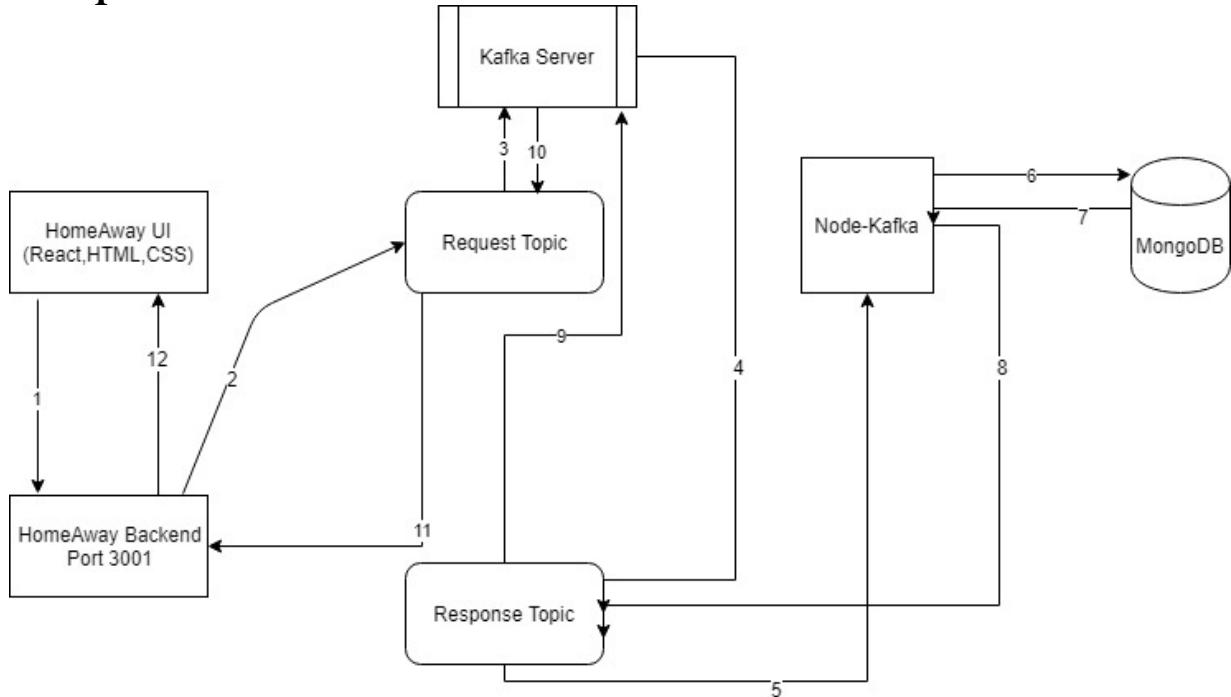
- The purpose of the system is to connect different property listers and the travelers who will access these listed properties.
- The system will allow different owners of properties around the globe to rent their properties on the system and in turn help them to earn through it.
- For travelers, this system will allow them to search for different properties all around the world and booking the one they want to rent.
- This system will take care of booking and listing of properties with proper session management.

System Design:

- The system is a HomeAway application used to achieve the goals of the system mentioned above. For UI, ReactJS and Redux is used and NodeJS handles the server and MongoDB serves as a Database.

- The request received from the client is passed to kafka's request topic. It then gets transferred to kafka server where proper handling of that request is done. Kafka, here, works with mongoDB to fetch and search through data and performs a required operation to send a desired response to the client.
- The response again is sent through one of the topic through which it is fetched and sent to the client.

Complete Flow:



Backend Routes

Routes	Method	Index Routes	Functionality
/travellogin	POST		<ul style="list-style-type: none"> - Sends the parameters like email and password and the route authenticates the user. If the credentials are correct, the user is logged in.
/travelsignup	POST		<ul style="list-style-type: none"> - Sign's up new user and send an acknowledgement on successful creation.
/ownerlogin	POST		<ul style="list-style-type: none"> - Sends the parameters like email and password and the route authenticates the owner. If the credentials are correct, the owner is logged in.

/ownersignup	POST	- Sign's up the new owner and send an acknowledgement on successful creation.
		Travel Routes
/:travelid	GET	- Get the details of the travel user having id :travelid
/:travelid	PUT	- Update the details of the travel user having id :travelid
/:travelid/editpassword	PUT	- Edits the password of the user having id :travelid
/:travelid/bookingdetails	GET	- Gets the booking details that the user having :travelid has done.
/:travelid/upload	POST	- Updates the profile pic of the traveler having id :travelid.
/:travelid/question	GET	- Shows all the questions asked by the traveler having id :travelid under their Inbox.
/:travelid/question	POST	- Allows traveler to post questions on properties listed by owners.
		Owner Routes
/:ownerid	GET	- Get the details of the owner user having id :ownerid
/:ownerid	PUT	- Updates the detail of the owner having id :ownerid.
/:ownerid/editpassword	PUT	- Edits the password of the user having id :ownerid
/:ownerid/property	GET	- Gives all the properties listed by the owner having id :ownerid
/:ownerid/dashboard	GET	- Gives the details about who has booked the properties listed by the owner having id :ownerid
/:ownerid/question	GET	- Shows all the questions asked on the properties listed by the owner having id :ownerid.
/:travelid/question	POST	- Allows owner to post answers on questions asked by travellers.
/		Property Routes
	POST	- Adds a new property

/:propertyid	GET	- Gets the details of the property having id :propertyid
/:propertid	PUT	- Updates the details of the property.
/search	POST	- Searches for properties with filters provided in the body.
/:propertyid/book	POST	- Books the property having id :propertyid
/:propertyid/upload	POST	- Uploads photos for the property having id :propertyid.

Test Cases and Results

We'll consider different cases and will visit them one by one. The use cases that I'll be discussing are as follows:

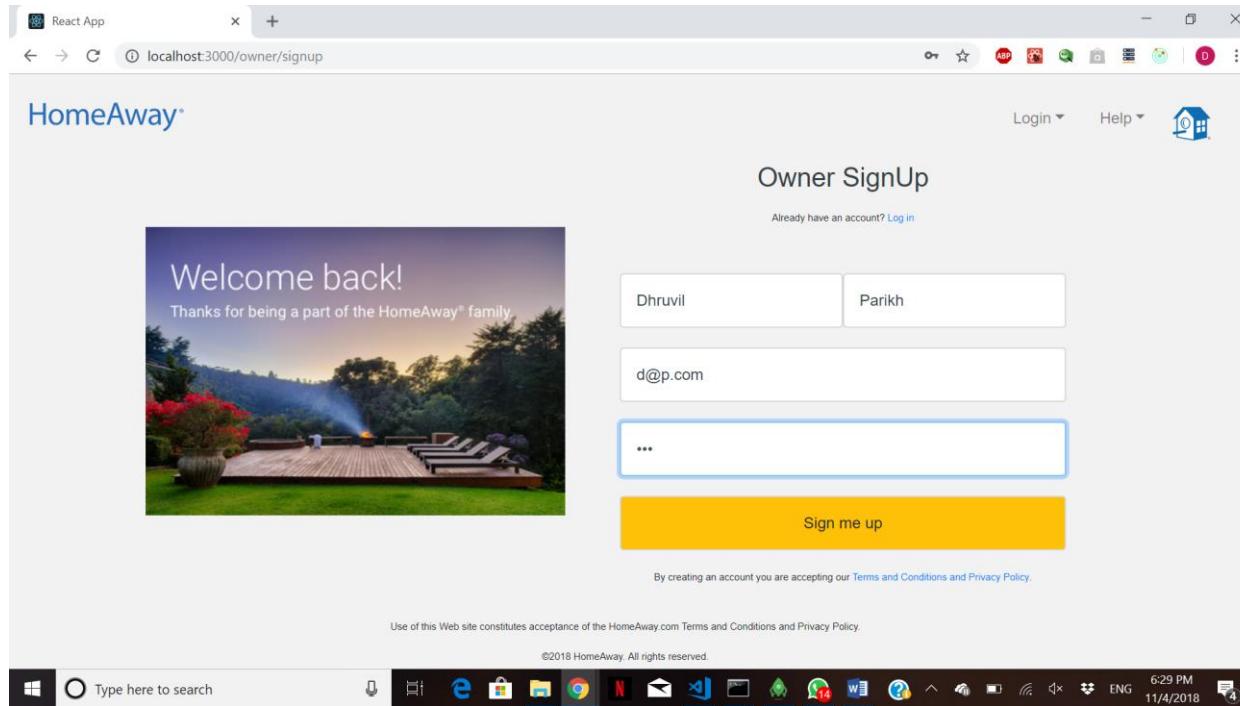
- 1) Creating a new owner account.**
- 2) Logging in with the newly created owner account.**
- 3) Listing a new property from this owner account.**
- 4) Displaying the properties listed by the owner.**
- 5) Editing property details.**
- 6) Editing the owner's detail like billing address, contact no and other things.**
- 7) Logging off from the owner account.**
- 8) Creating a new travel account.**
- 9) Logging in with the newly created travel account.**
- 10) Editing the basic details of travel account.**
- 11) Searching a property based on “city, arrival, departure and # of guests”.**
- 12) Displaying the result of the above search and filtering through it.**
- 13) Asking Owner the question on any of the property searched.**
- 14) Selecting a property from search result and booking it with the travel account.**
- 15) Looking at the recent trips that I have booked and filtering through it.**
- 16) Logging off from the travel account.**
- 17) Answering to the question asked by the traveler.**
- 18) Looking at Owner's Dashboard.**

We'll also consider different use cases which will have invalid inputs in form like arrival date is after the departure date and many more. Different use cases that we'll consider for validations are as follows:

- 1) Keeping the email/password field empty**
- 2) Logging in with incorrect credentials**
- 3) Searching the property without location city or arrival date or departure date or guests.**
- 4) Editing the information and keeping required fields like name, contact no blank.**

Test Cases:

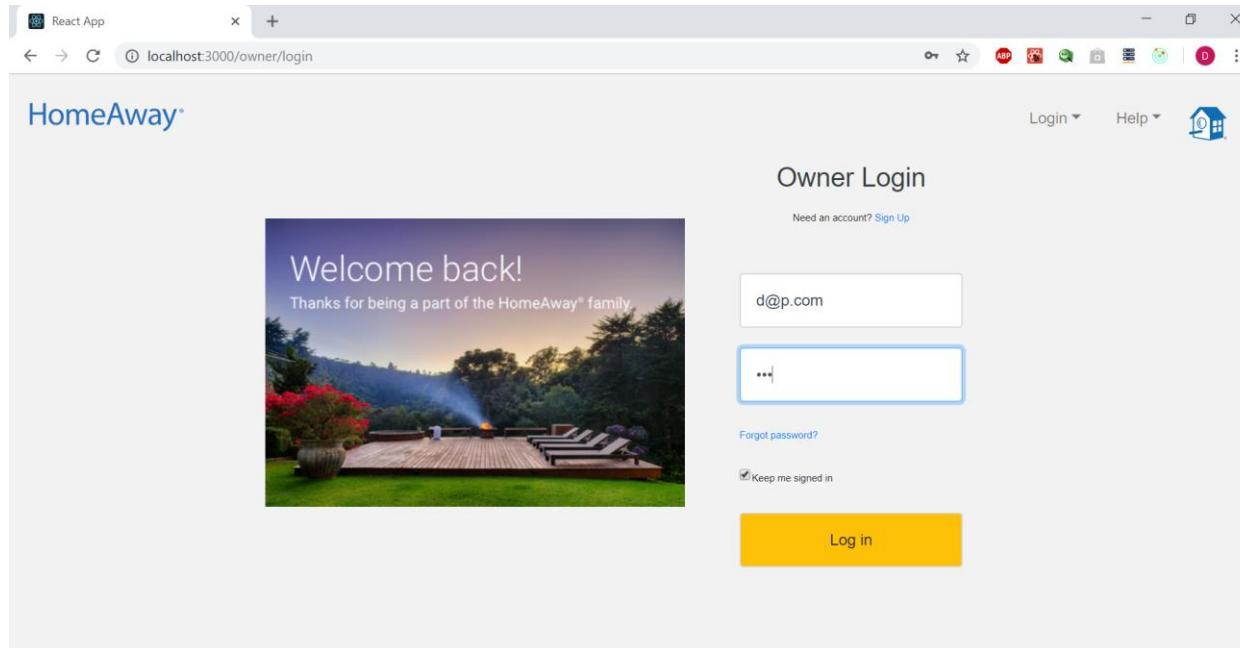
Creating a new owner account

A screenshot of a terminal window titled 'React App'. It shows the command 'node .bin/www' being run. The log output includes: 'Connected [nodemon] restarting due to changes...', '[nodemon] starting 'node ./bin/www'', 'producer ready', '(node:29860) DeprecationWarning: current URL string parser is deprecated, and will be removed in a future version. To use the new parser, pass option { useNewUrlParser: true } to MongoClient.connect.', '(node:29860) DeprecationWarning: collection.ensureIndex is deprecated. Use createIndexes instead.', 'Connected', 'clear'. Below this, it says 'OPTIONS /ownersignup 204 2.065 ms - -'. Then it shows a 'Message' object with properties: {}, bookings: [], id: '5b6e322c40d74a417d99d', email: 'd@p.com', password: '\$2a\$10\$whVj9XAn5opYgGdpQm604kP/9LgtLuOyAfPv5r3Az9zIF5hajetq', firstname: 'Dhruvil', lastname: 'Parikh', v: 0. The final line is 'POST /ownersignup 200 232.917 ms - -'.

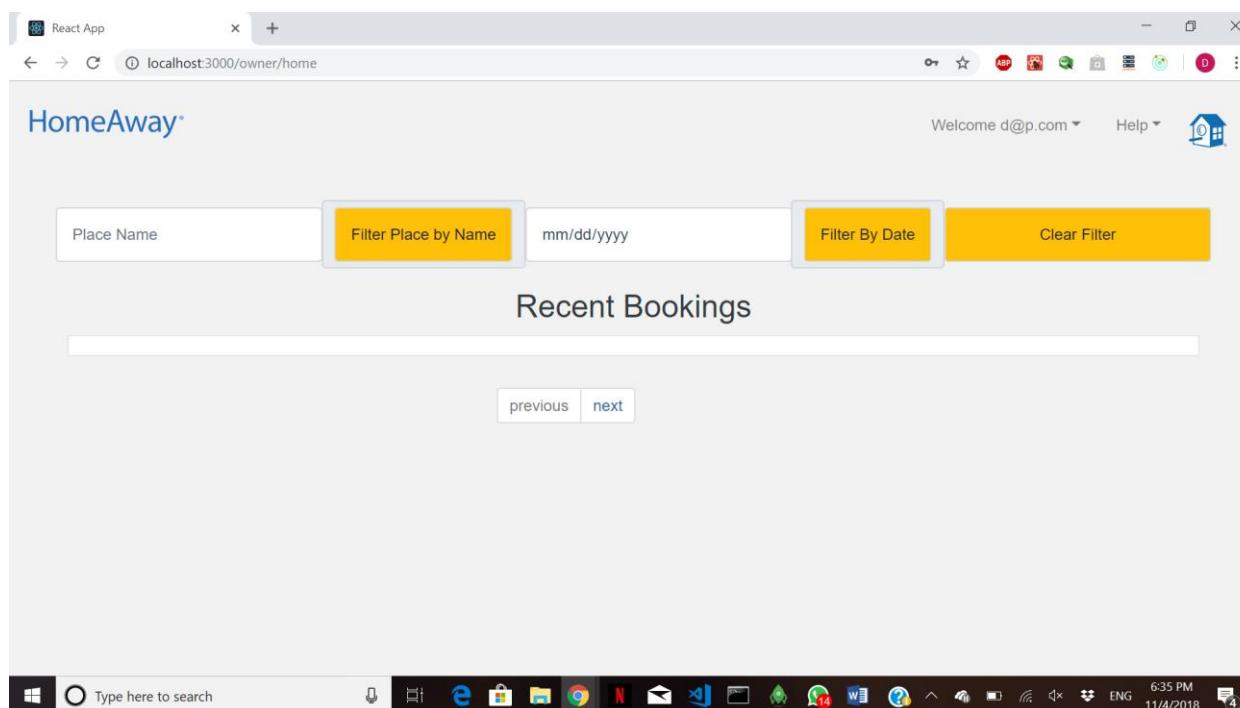
Explanation:

- A new owner will be created named "Dhruvil Parikh" and login credentials of d@p.com and the password encrypted with bcrypt.
- As one can see in the output of the terminal, the post request has been successfully completed and the password too has been stored after being encrypted with bcrypt and the salt cycle running for 10 rounds.

Logging in with the newly created owner account



The screenshot shows a web browser window titled "React App" with the URL "localhost:3000/owner/login". The page is titled "Owner Login" and features a "Welcome back!" message with a photo of a deck at sunset. It includes fields for email ("d@p.com") and password, a "Forgot password?" link, a "Keep me signed in" checkbox, and a yellow "Log in" button.



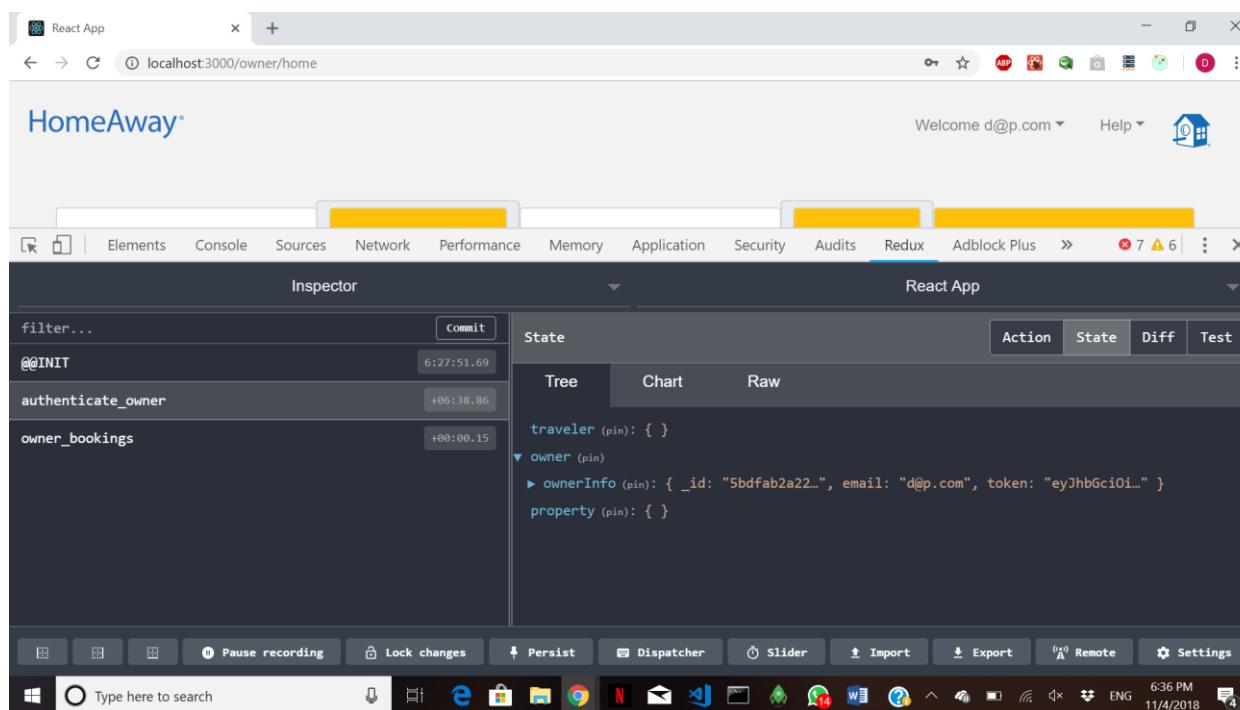
The screenshot shows a web browser window titled "React App" with the URL "localhost:3000/owner/home". The page is titled "Recent Bookings" and displays a search bar with fields for "Place Name", "Filter Place by Name", "mm/dd/yyyy", "Filter By Date", and "Clear Filter". Below the search bar is a "previous" and "next" navigation button.

The screenshot shows the Visual Studio Code interface. The left sidebar has icons for file operations like Open, Save, Find, and Refresh. The main editor area shows code for `Details.js`. The terminal below shows a sequence of API requests and responses:

```

OPTIONS /ownerlogin 204 0.413 ms - 
Inside Login Post Request
d@p.com 123
password matched
POST /ownerlogin 200 285.206 ms - -
Trying to fetch booking details
in make request
{ owner_id: '5bdfab2a22c40d74a417d99d' }
1
returning next
in response: getOwnerBookings
in response1
true
in response2
{ getOwnerBookings: { '0': 63 } }
client ready!
msg received
Got the booking data for 5bdfab2a22c40d74a417d99d
GET /owner/5bdfab2a22c40d74a417d99d/dashboard 200 109.033 ms - -
  
```

The status bar at the bottom indicates the file is `LoginPage.js - Lab2 - Visual Studio Code`, line 88, column 71, with 4 spaces, using UTF-8 encoding, and ESLint is active.

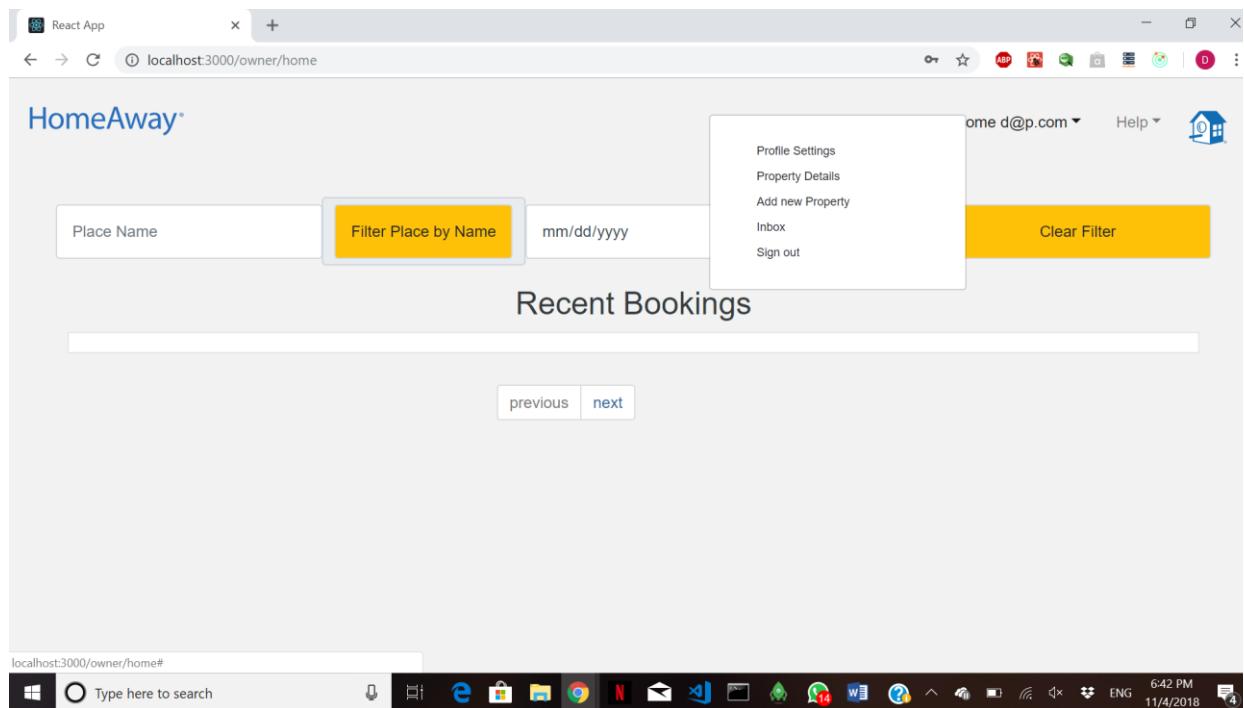


Explanation:

- The owner logged in using his login credentials.
- Notice the navigation bar, when the owner has not logged in and thus getting the option of “Login” only at the top.
- Please note the change once we login, the navigation bar now has welcome d@p.com on top.

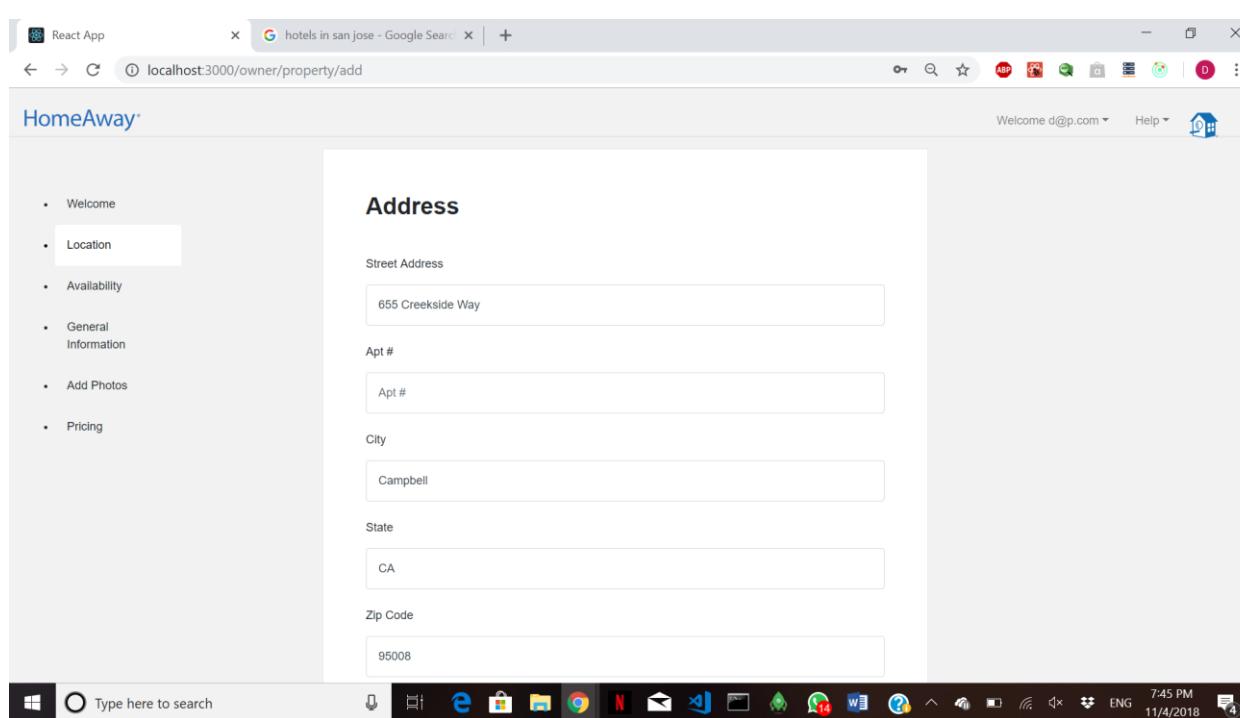
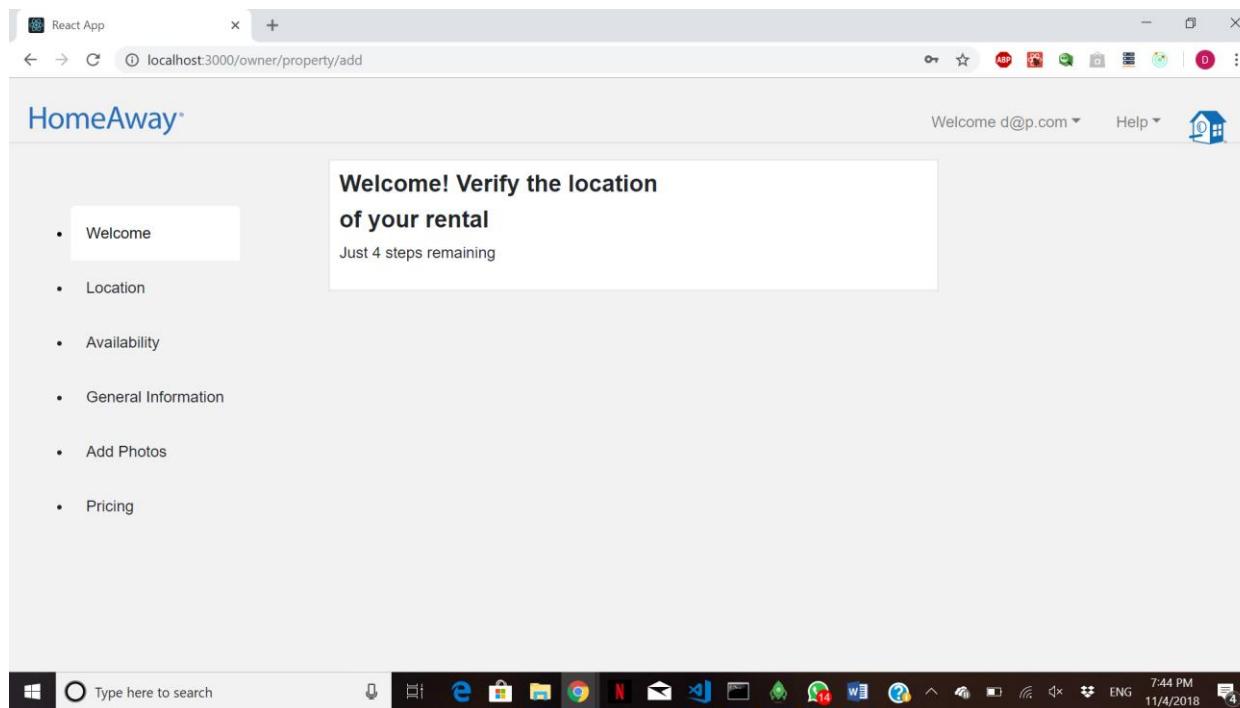
- After logging in, the redux store updates itself with owner information.

Listing a new property from this owner account



Explanation:

- This is the home screen of the owner.
- As you can see on the navigation bar, the “login” option has changed and the owner now gets access to options like “Add new property, profile settings and many more.”
- This screen usually shows the bookings that the travelers have done on the listings that that owner has listed for the rent.
- The screen also allows user to filter through the bookings done on their properties.
- We will be able to see the booking over here after we will create a new traveler who will book this owner’s properties.



React App X Google hotels in san jose - Google Search | +

localhost:3000/owner/property/add

Welcome d@p.com Help

HomeAway®

- Welcome
- Location
- Availability
- General Information
- Add Photos
- Pricing

Availability

Already know when would you like your property to be available?

You can also make changes after publishing your listing

Start Date

11/06/2018

Start Date

10/31/2019 X ▾ ▾

Use of this Web site constitutes acceptance of the HomeAway.com Terms and conditions and Privacy policy.

©2018 HomeAway. All rights reserved

Start Co-browse

This screenshot shows the 'Availability' step of a property listing process. On the left, a sidebar lists steps: Welcome, Location, Availability (which is selected), General Information, Add Photos, and Pricing. The main area has a heading 'Availability' and two text input fields for start dates: '11/06/2018' and '10/31/2019'. Below these are links for terms and privacy policy, and a copyright notice. At the bottom right is a 'Start Co-browse' button.

React App X Google hotels in san jose - Google Search | +

localhost:3000/owner/property/add

Welcome d@p.com Help

HomeAway®

- Welcome
- Location
- Availability
- General Information
- Add Photos
- Pricing

Describe your property

Start out by entering the property name

Courtyard by Marriott San Jose Campbell

Tell us a headline

Live with Luxury

Give us a detailed description of your property

This modern hotel is a 5-minute walk from the 9.7-mile Los Gatos Creek Trail, and a 2.9-mile dr

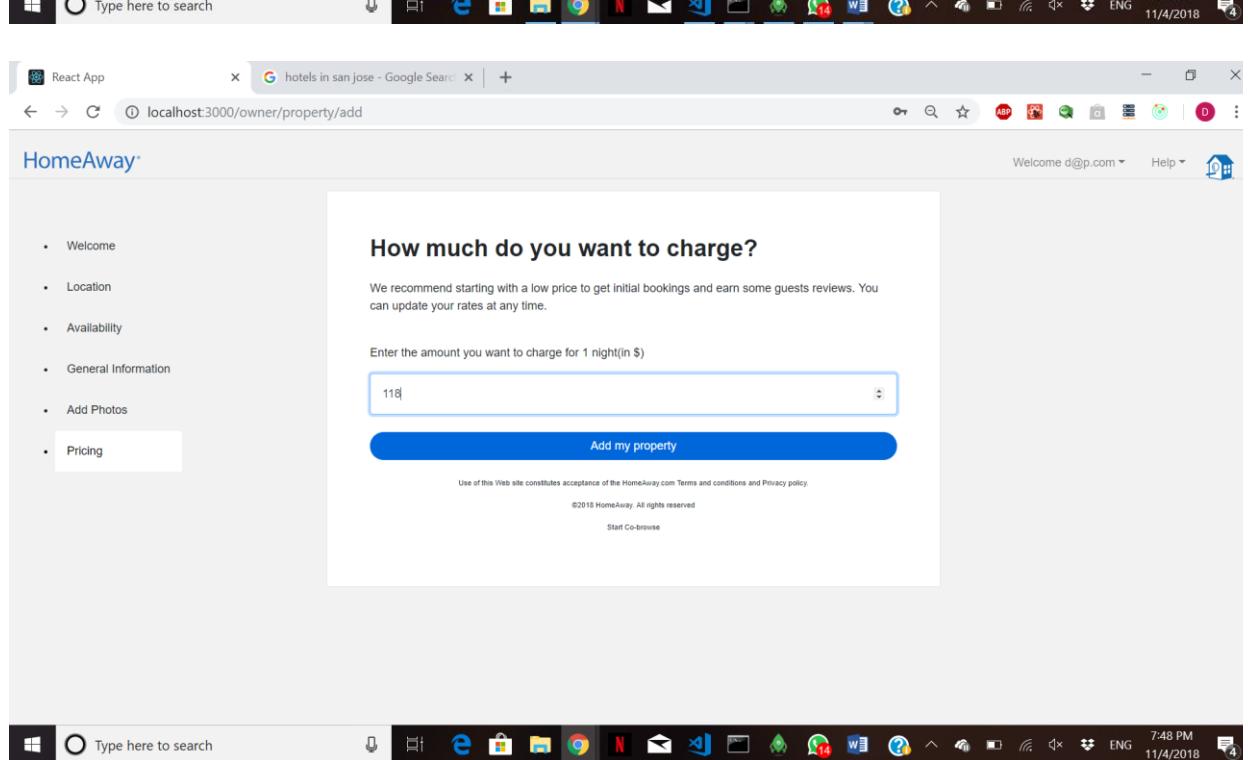
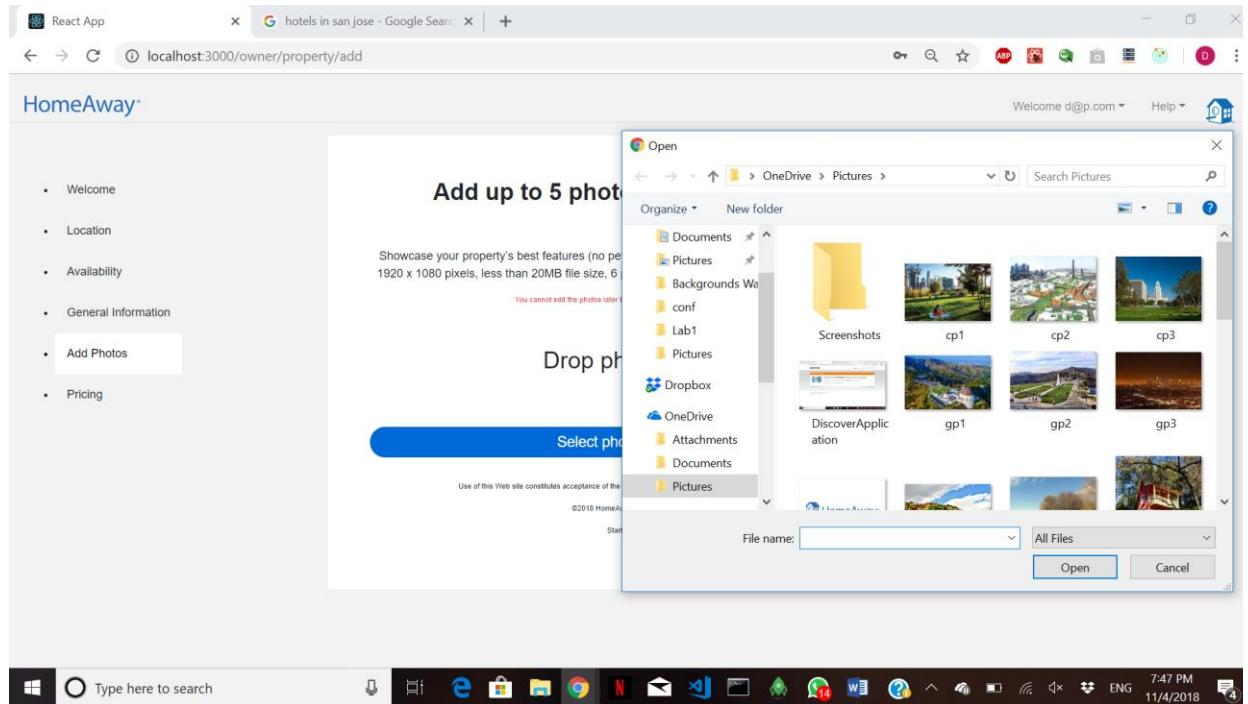
How many bedrooms do you have?

2

What about bathrooms?

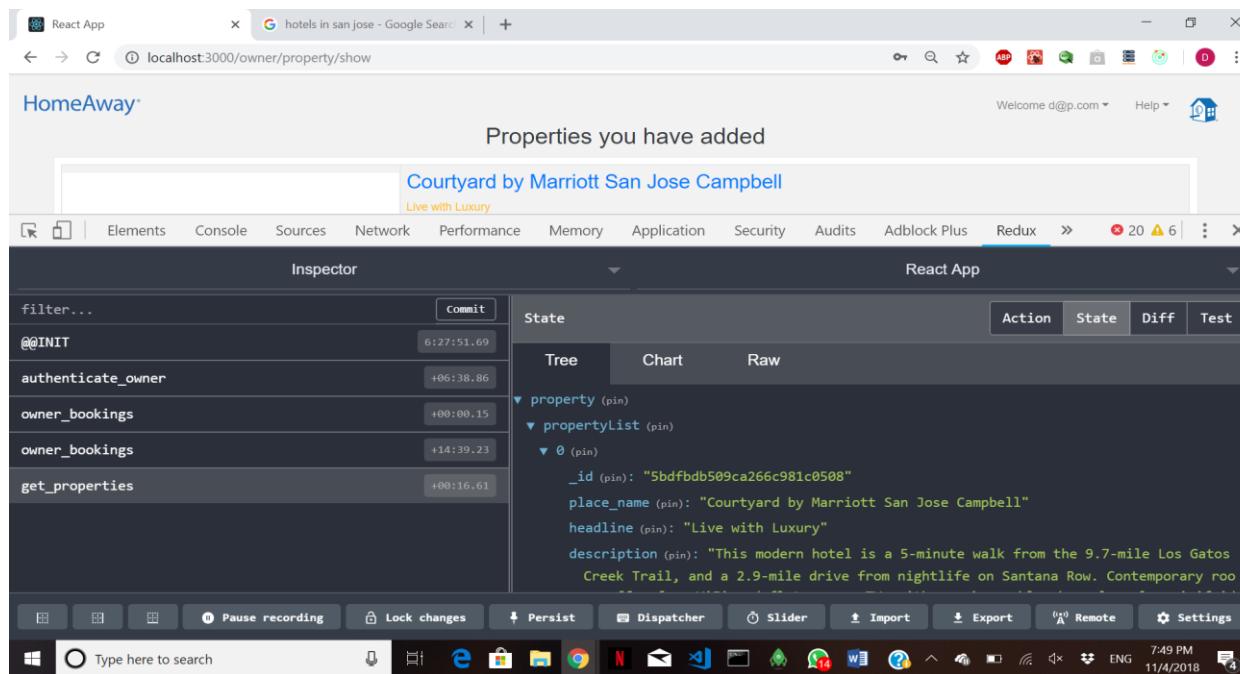
2

This screenshot shows the 'Describe your property' step of the listing process. The sidebar shows 'General Information' is selected. The main area includes fields for property name ('Courtyard by Marriott San Jose Campbell'), headline ('Live with Luxury'), detailed description ('This modern hotel is a 5-minute walk from the 9.7-mile Los Gatos Creek Trail, and a 2.9-mile dr'), number of bedrooms ('2'), and number of bathrooms ('2').



```
File Edit Selection View Go Debug Terminal Help • LoginPage.js - Lab2 - Visual Studio Code
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
producer ready
(node:34804) DeprecationWarning: current URL string parser is deprecated, and will be removed in a future version. To use the new parser, pass option { useNewUrlParser: true } to MongoClient.connect.
(node:34804) DeprecationWarning: collection.ensureIndex is deprecated. Use createIndexes instead.
Connected
OPTIONS /property/ 2.305 ms - 0
Trying to add a new property for ownerid: 5bdfab2a22c40d74a417d99d
in make request
{
  owner_id: '5bdfab2a22c40d74a417d99d',
  place_name: 'Courtyard by Marriott San Jose Campbell',
  street: '655 Creekside Way',
  apt: '',
  state: 'CA',
  zipcode: '95008',
  country: 'US',
  location_city: 'Campbell',
  available_from: '2018-11-06',
  available_to: '2019-10-31',
  bedrooms: '2',
  bathrooms: '2',
  accommodates: '5',
  headline: 'Live with Luxury',
  description: 'This modern hotel is a 5-minute walk from the 9.7-mile Los Gatos Creek Trail, and a 2.9-mile drive from nightlife on Santana Row. Contemporary rooms offer free WiFi and flat-screen TVs with premium cable channels, plus minifridges and coffeemakers. Suites and upgraded rooms add separate living areas with pull-out sofas, and some have microwaves.'
```

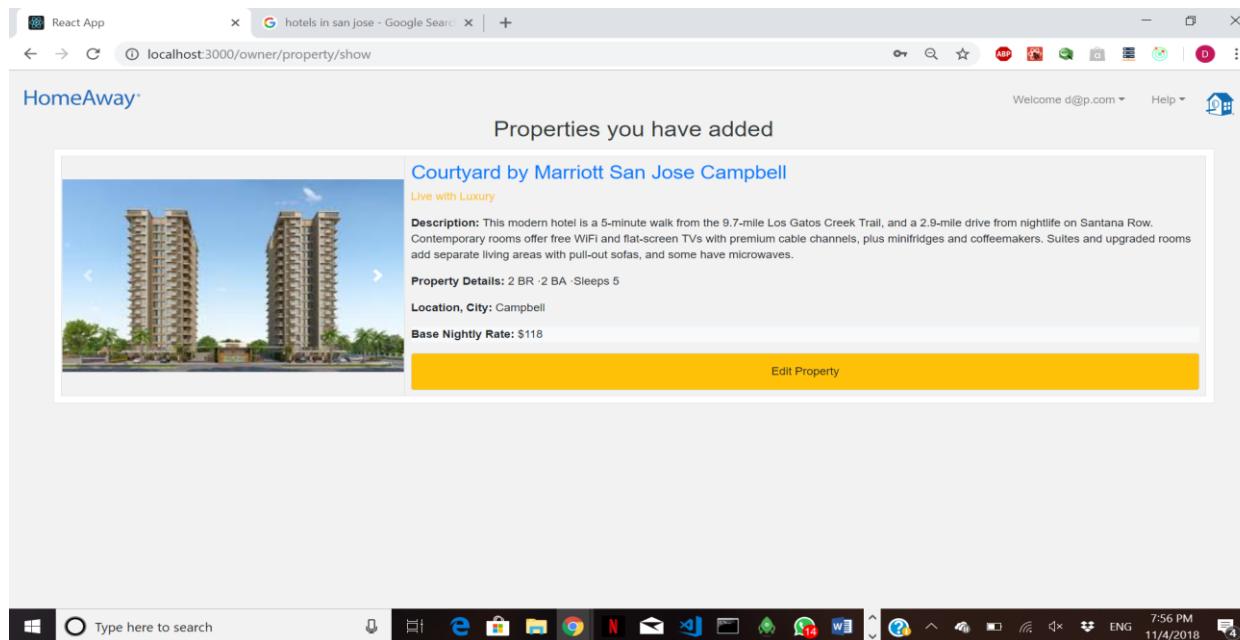
```
File Edit Selection View Go Debug Terminal Help • LoginPage.js - Lab2 - Visual Studio Code
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
00:00:00.000Z", "2019-10-16T00:00:00.000Z", "2019-10-17T00:00:00.000Z", "2019-10-18T00:00:00.000Z", "2019-10-19T00:00:00.000Z", "2019-10-20T00:00:00.000Z", "2019-10-21T00:00:00.000Z", "2019-10-22T00:00:00.000Z", "2019-10-23T00:00:00.000Z", "2019-10-24T00:00:00.000Z", "2019-10-25T00:00:00.000Z", "2019-10-26T00:00:00.000Z", "2019-10-27T00:00:00.000Z", "2019-10-28T00:00:00.000Z", "2019-10-29T00:00:00.000Z", "2019-10-30T00:00:00.000Z", "2019-10-31T00:00:00.000Z"}}
Inside getOwnerDetails kafka backend
after callback
after handle{ _v: 0,
  place_name: 'Courtyard by Marriott San Jose Campbell',
  headline: 'Live with Luxury',
  description: 'This modern hotel is a 5-minute walk from the 9.7-mile Los Gatos Creek Trail, and a 2.9-mile drive from nightlife on Santana Row. Contemporary rooms offer free WiFi and flat-screen TVs with premium cable channels, plus minifridges and coffeemakers. Suites and upgraded rooms add separate living areas with pull-out sofas, and some have microwaves.',
  street: '655 Creekside Way',
  apt: '',
  location_city: 'Campbell',
  state: 'CA',
  zipcode: '95008',
  country: 'US',
  available_from: 2018-11-06T00:00:00.000Z,
  available_to: 2019-10-31T00:00:00.000Z,
  bedrooms: 2,
  bathrooms: 2,
  accommodates: 5,
  price: 118,
  _id: 5bdfbdb509ca266c981c0508,
  bookings: []},
```

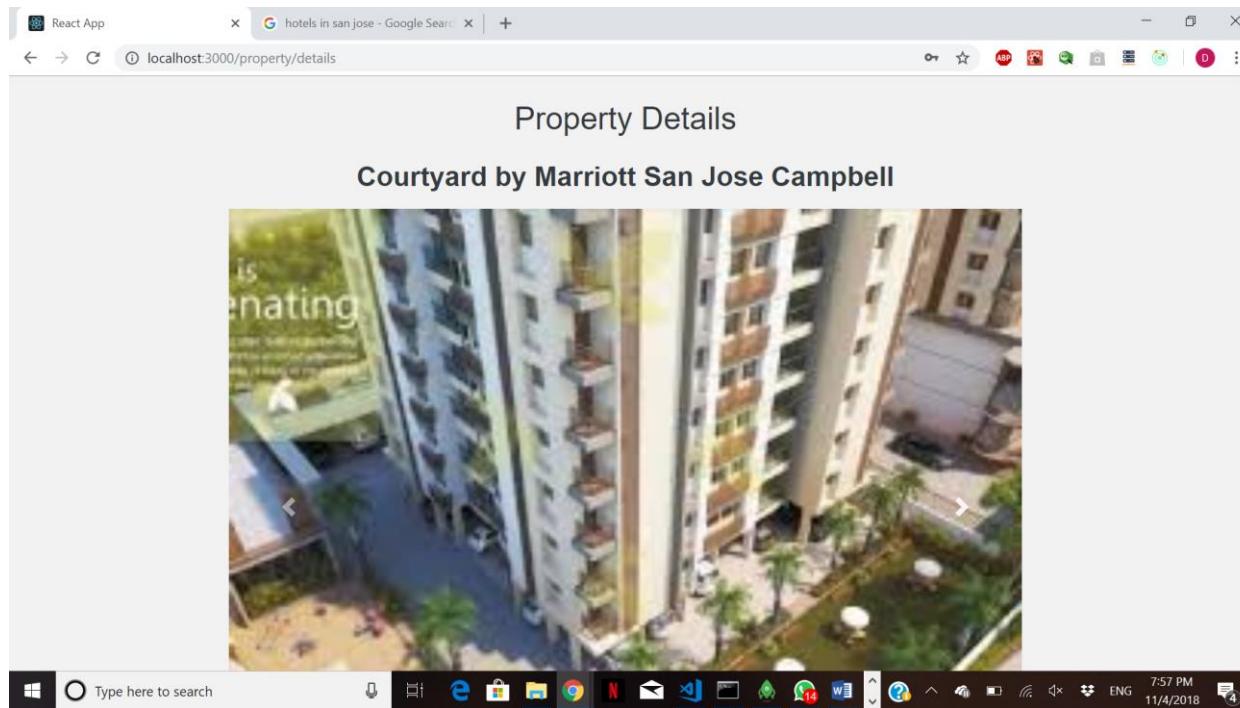


Explanation:

- These screenshots showcases the steps of how owner can add a new property listing.
- The property is located in “Campbell” city.
- The redux store shows that the store has now updated and it has stored the newest property added by the owner.

Displaying the properties listed by the owner





File Edit Selection View Go Debug Terminal Help • LoginPage.js - Lab2 - Visual Studio Code

details.js JS OwnerHomePage.js JS Inbox.js JS EditProfile.js JS index.js JS OwnerLoginPage.js JS LoginPage.js • JS Main.js JS ListPlaces.js

```
84     console.log(nextProps.error)
85     if(nextProps.travelerInfo){
86       localStorage.setItem("loginuser",nextProps.travelerInfo._id)
87       localStorage.setItem("loginemail",nextProps.travelerInfo.email)
88       localStorage.setItem("token",nextProps.travelerInfo.token)
89     }
90   }else if(nextProps.error){
91     this.setState({
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL 2: node

```
POST /property/5bdfbdb509ca266c981c0508/upload 200 99.472 ms - -
should be pushed
POST /property/5bdfbdb509ca266c981c0508/upload 200 458.675 ms - -
Trying to get properties listed by owner id: 5bdfab2a22c40d74a417d99d
in make request
{ owner_id: '5bdfab2a22c40d74a417d99d' }
in response: getOwnerProperties
in response1
true
in response2
{ getOwnerProperties: { '0': 24 } }
msg received
GET /owner/5bdfab2a22c40d74a417d99d/property 200 190.422 ms - -
GET /public/uploads/ha.jpeg 304 4.639 ms - -
GET /public/uploads/property-5bdfbdb509ca266c981c0508-xy1.jpg 200 5.069 ms - 6639
GET /public/uploads/property-5bdfbdb509ca266c981c0508-xy3.jpg 200 9.651 ms - 12741
GET /public/uploads/property-5bdfbdb509ca266c981c0508-xy2.jpg 200 11.471 ms - 9628
clearTrying to get details of property having id: 5bdfbdb509ca266c981c0508
in make request
{ propertyId: '5bdfbdb509ca266c981c0508' }
in response: getPropertyDetails
in response1
true
in response2
{ getPropertyDetails: { '0': 24 } }
```

master* 0 7:59 PM 11/4/2018

```

84     console.log(nextProps.error)
85     if(nextProps.travelerInfo){
86       localStorage.setItem("loginuser",nextProps.travelerInfo._id)
87       localStorage.setItem("loginemail",nextProps.travelerInfo.email)
88       localStorage.setItem("token",nextProps.travelerInfo.token)
89       this.props.history.push('/traveller/home')
90     }else if(nextProps.error){
91       this.setState({
92         error:nextProps.error
93       })
94     }
95   }
96 }

after handle
{
  response_topic: { '0': 1536 }
}
message received for getOwnerProperties { handle_request: [Function: handle_request] }
"{"correlationId":"e6e19eeaaadc29362a8e5825bb98b0e45","replyTo":"response_topic","data":{"owner_id":"5bdfab2a22c40d74a417d99d"}}
Inside getOwnerDetails kafka backend
after callback
after handle{ _id: 5bdfab2a22c40d74a417d99d,
  place_name: 'Courtyard by Marriott San Jose Campbell',
  headline: 'Live with Luxury',
  description: 'This modern hotel is a 5-minute walk from the 9.7-mile Los Gatos Creek Trail, and a 2.9-mile drive from nightlife on Santana Row. Contemporary rooms offer free WiFi and flat-screen TVs with premium cable channels, plus minifridges and coffeemakers. Suites and upgraded rooms add separate living areas with pull-out sofas, and some have microwaves.',
  street: '655 Creekside Way',
  apt: '',
  location_city: 'Campbell',
  state: 'CA',
  zipcode: '95008',
  country: 'US',
  available_from: 2018-11-06T00:00:00.000Z,
  available_to: 2019-10-31T00:00:00.000Z,
  bedrooms: 2,
  bathrooms: 2,
  accommodates: 5,
  price: 118,
  _v: 0,
}

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

Ln 88, Col 71 Spaces: 4 UTF-8 CRLF Javascript (Babel) ESLint

Windows Taskbar: Type here to search, File Explorer, Edge, Google Chrome, Mail, Netflix, HomeAway, HomeAway logo, HomeAway Help, HomeAway Welcome d@p.com, HomeAway Logout, HomeAway Home, HomeAway Properties you have added, Courtyard by Marriott San Jose Campbell, Live with Luxury, Description: This modern hotel is a 5-minute walk from the 9.7-mile Los Gatos Creek Trail, and a 2.9-mile drive from nightlife on Santana Row. Contemporary rooms offer free WiFi and flat-screen TVs with premium cable channels, plus minifridges and coffeemakers. Suites and upgraded rooms add separate living areas with pull-out sofas, and some have microwaves., Property Details: 2 BR · 2 BA · Sleeps 5, Location, City: Campbell, Base Nightly Rate: \$118, Edit Property.

Explanation:

- These screenshots show the properties listed by the owner.
- On the property list, if we click on the property name, it redirects to a propertyDetails page as shown in the screenshot.
- This request is being processed by “getOwnerProperties” topic in kafka.

Editing property details

React App | G - hotels in san jose - Google Search | +

localhost:3000/owner/property/show

Welcome d@p.com | Help | HomeAway

Properties you have added

Courtyard by Marriott San Jose Campbell

Live with Luxury

Description: This modern hotel is a 5-minute walk from the 9.7-mile Los Gatos Creek Trail, and a 2.9-mile drive from nightlife on Santana Row. Contemporary rooms offer free WiFi and flat-screen TVs with premium cable channels, plus minifridges and coffeemakers. Suites and upgraded rooms add separate living areas with pull-out sofas, and some have microwaves.

Property Details: 2 BR · 2 BA · Sleeps 5

Location, City: Campbell

Base Nightly Rate: \$118

Edit Property

Windows Taskbar: Type here to search, File Explorer, Edge, Google Chrome, Mail, Netflix, HomeAway, HomeAway logo, HomeAway Help, HomeAway Welcome d@p.com, HomeAway Logout, HomeAway Home, HomeAway Properties you have added, Courtyard by Marriott San Jose Campbell, Live with Luxury, Description: This modern hotel is a 5-minute walk from the 9.7-mile Los Gatos Creek Trail, and a 2.9-mile drive from nightlife on Santana Row. Contemporary rooms offer free WiFi and flat-screen TVs with premium cable channels, plus minifridges and coffeemakers. Suites and upgraded rooms add separate living areas with pull-out sofas, and some have microwaves., Property Details: 2 BR · 2 BA · Sleeps 5, Location, City: Campbell, Base Nightly Rate: \$118, Edit Property.

A screenshot of a Microsoft Edge browser window. The address bar shows 'localhost:3000/owner/property/edit'. The page title is 'React App' and the sub-page title is 'hotels in san jose - Google Search'. The main content area has a heading 'Describe your property'. To the left is a sidebar with a list of steps: 'Welcome', 'Location', 'Availability', 'General Information' (which is selected), and 'Pricing'. Below the sidebar are four input fields: 'Start out by entering the property name' containing 'Courtyard by Marriott San Jose Campbell', 'Tell us a headline' containing 'Live with Luxury', 'Give us a detailed description of your property' containing 'This modern hotel is a 5-minute walk from the 9.7-mile Los Gatos', and 'How many bedrooms do you have?' containing '2'. The taskbar at the bottom shows various pinned icons.

A screenshot of the same Microsoft Edge browser window, showing the state after a user has typed 'Marriott' into the 'Start out by entering the property name' input field. The input field now contains 'Courtyard by Marriot' with a blue outline, indicating it is the active field. The other input fields ('headline', 'description', and 'bedrooms') remain unchanged. The sidebar and taskbar are also visible.

React App x Google hotels in san jose - Google Search x | +

localhost:3000/owner/property/show

HomeAway® Welcome d@p.com Help ▾

Properties you have added

Courtyard by Marriott

Live with Luxury

Description: This modern hotel is a 5-minute walk from the 9.7-mile Los Gatos Creek Trail, and a 2.9-mile drive from nightlife on Santana Row. Contemporary rooms offer free WiFi and flat-screen TVs with premium cable channels, plus minifridges and coffeemakers. Suites and upgraded rooms add separate living areas with pull-out sofas, and some have microwaves.

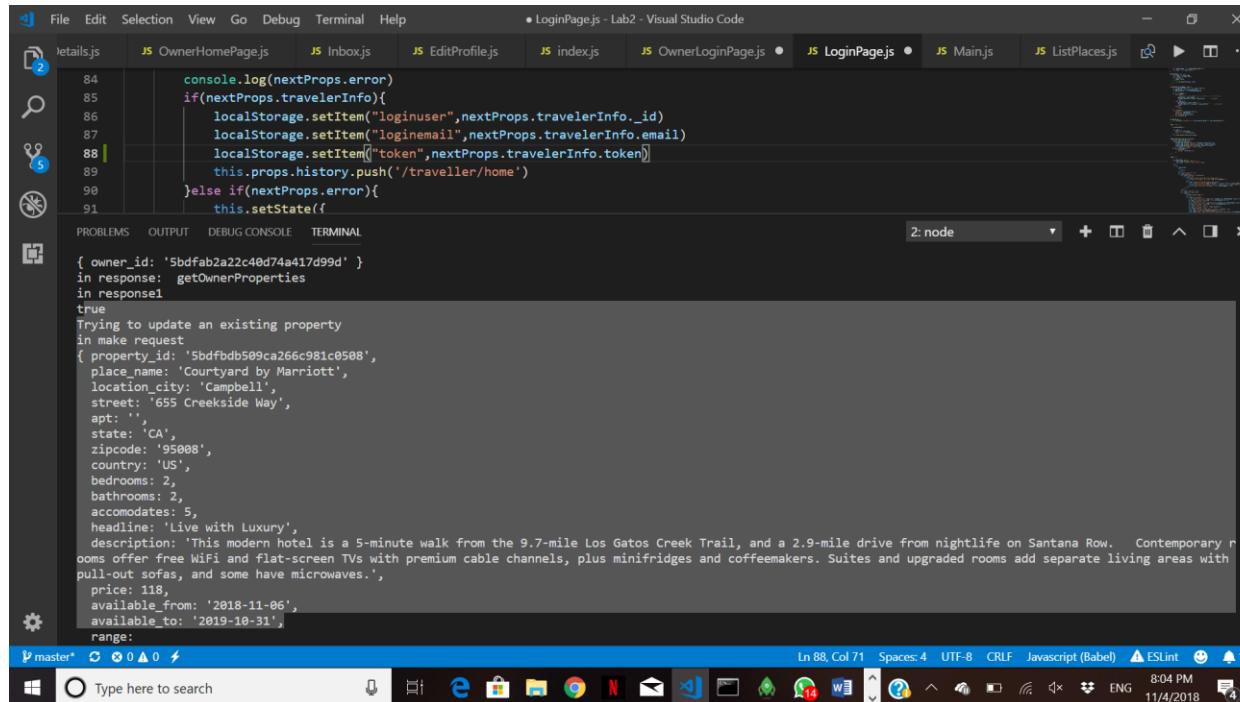
Property Details: 2 BR · 2 BA · Sleeps 5

Location, City: Campbell

Base Nightly Rate: \$118

Edit Property





```
84     console.log(nextProps.error)
85     if(nextProps.travelerInfo){
86       localStorage.setItem("loginuser",nextProps.travelerInfo._id)
87       localStorage.setItem("loginemail",nextProps.travelerInfo.email)
88       localStorage.setItem("token",nextProps.travelerInfo.token)
89       this.props.history.push('/traveller/home')
90     }else if(nextProps.error){
91       this.setState({
```

{ owner_id: '5bdfab2a22c40d74a417d99d' }

in response: getOwnerProperties

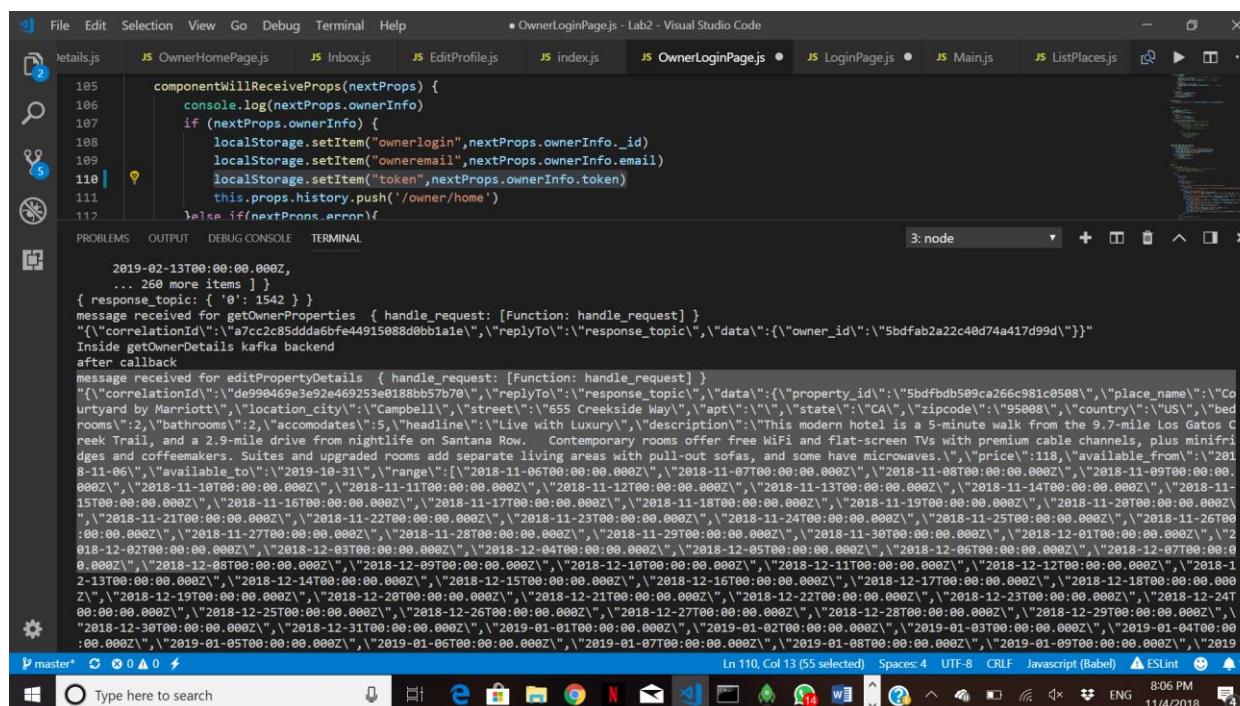
in response1

true

Trying to update an existing property

in make request

{ property_id: '5bdfbdb509ca266c981c0508', place_name: 'Courtyard by Marriott', location_city: 'Campbell', street: '655 Creekside Way', apt: '', state: 'CA', zipcode: '95008', country: 'US', bedrooms: 2, bathrooms: 2, accommodates: 5, headline: 'Live with Luxury', description: 'This modern hotel is a 5-minute walk from the 9.7-mile Los Gatos Creek Trail, and a 2.9-mile drive from nightlife on Santana Row. Contemporary rooms offer free WiFi and flat-screen TVs with premium cable channels, plus minifridges and coffeemakers. Suites and upgraded rooms add separate living areas with pull-out sofas, and some have microwaves.', price: 118, available_from: '2018-11-06', available_to: '2019-10-31', range:



```
105     componentWillReceiveProps(nextProps) {
106       console.log(nextProps.ownerInfo)
107       if (nextProps.ownerInfo) {
108         localStorage.setItem("ownerlogin",nextProps.ownerInfo._id)
109         localStorage.setItem("owneremail",nextProps.ownerInfo.email)
110         localStorage.setItem("token",nextProps.ownerInfo.token)
111         this.props.history.push('/owner/home')
112       }else if(nextProps.error){
```

2019-02-13T00:00:00.000Z, ... 260 more items] }

{ response_topic: { '0': 1542 } }

message received for getOwnerProperties { handle_request: [Function: handle_request] }

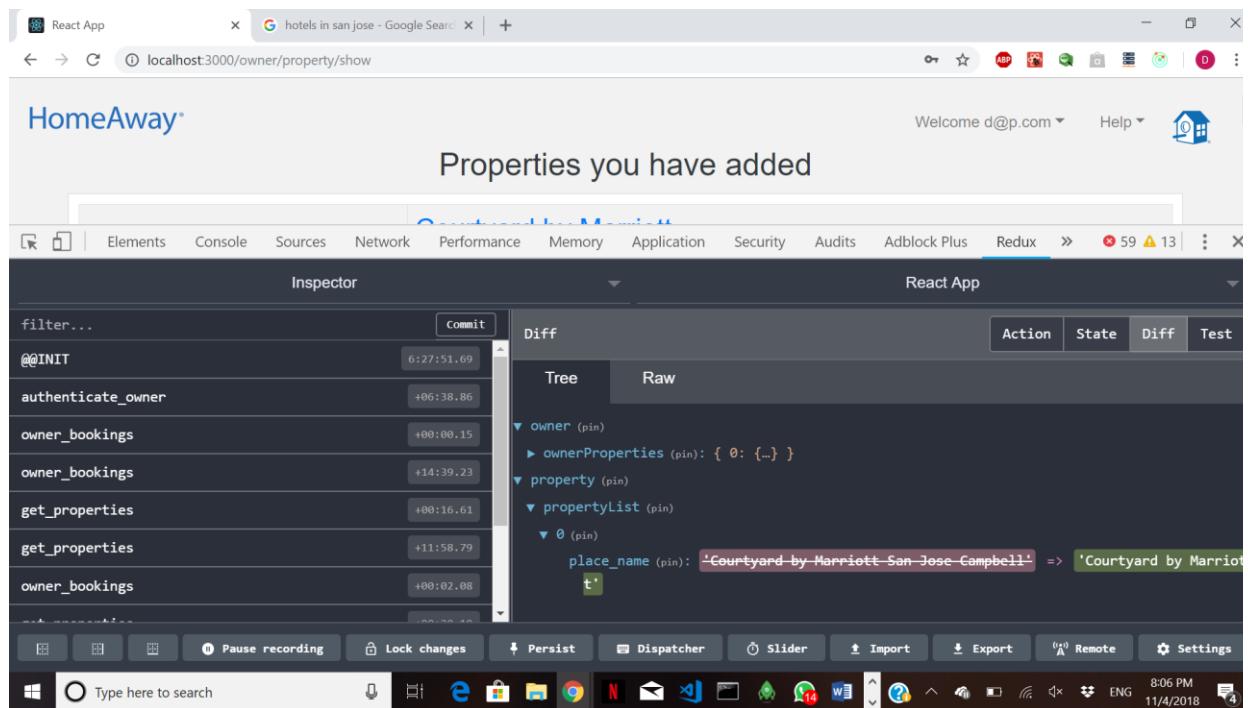
"{"correlationId":"a7cc2c85ddda6bfe44915088d0bb1a1","replyTo":"response_topic","data":{"owner_id":"5bdfab2a22c40d74a417d99d"}}"

Inside getOwnerDetails kafka backend

after callback

message received for editPropertyDetails { handle_request: [Function: handle_request] }

"{"correlationId":"de99469e3e92e469253e0188b057b70","replyTo":"response_topic","data":{"property_id":"5bdfbdb509ca266c981c0508","place_name":"Courtyard by Marriott","location_city":"Campbell","street":"655 Creekside Way","apt":"","state":"CA","zipcode":"95008","country":"US","bedrooms":2,"bathrooms":2,"accommodates":5,"headline":"Live with Luxury","description":"This modern hotel is a 5-minute walk from the 9.7-mile Los Gatos Creek Trail, and a 2.9-mile drive from nightlife on Santana Row. Contemporary rooms offer free WiFi and flat-screen TVs with premium cable channels, plus minifridges and coffeemakers. Suites and upgraded rooms add separate living areas with pull-out sofas, and some have microwaves.","price":118,"available_from":"2018-11-06T00:00:00.000Z","2018-11-07T00:00:00.000Z","2018-11-08T00:00:00.000Z","2018-11-09T00:00:00.000Z","2018-11-10T00:00:00.000Z","2018-11-11T00:00:00.000Z","2018-11-12T00:00:00.000Z","2018-11-13T00:00:00.000Z","2018-11-14T00:00:00.000Z","2018-11-15T00:00:00.000Z","2018-11-16T00:00:00.000Z","2018-11-17T00:00:00.000Z","2018-11-18T00:00:00.000Z","2018-11-19T00:00:00.000Z","2018-11-20T00:00:00.000Z","2018-11-21T00:00:00.000Z","2018-11-22T00:00:00.000Z","2018-11-23T00:00:00.000Z","2018-11-24T00:00:00.000Z","2018-11-25T00:00:00.000Z","2018-11-26T00:00:00.000Z","2018-11-27T00:00:00.000Z","2018-11-28T00:00:00.000Z","2018-11-29T00:00:00.000Z","2018-11-30T00:00:00.000Z","2018-12-01T00:00:00.000Z","2018-12-02T00:00:00.000Z","2018-12-03T00:00:00.000Z","2018-12-04T00:00:00.000Z","2018-12-05T00:00:00.000Z","2018-12-06T00:00:00.000Z","2018-12-07T00:00:00.000Z","2018-12-08T00:00:00.000Z","2018-12-09T00:00:00.000Z","2018-12-10T00:00:00.000Z","2018-12-11T00:00:00.000Z","2018-12-12T00:00:00.000Z","2018-12-13T00:00:00.000Z","2018-12-14T00:00:00.000Z","2018-12-15T00:00:00.000Z","2018-12-16T00:00:00.000Z","2018-12-17T00:00:00.000Z","2018-12-18T00:00:00.000Z","2018-12-19T00:00:00.000Z","2018-12-20T00:00:00.000Z","2018-12-21T00:00:00.000Z","2018-12-22T00:00:00.000Z","2018-12-23T00:00:00.000Z","2018-12-24T00:00:00.000Z","2018-12-25T00:00:00.000Z","2018-12-26T00:00:00.000Z","2018-12-27T00:00:00.000Z","2018-12-28T00:00:00.000Z","2018-12-29T00:00:00.000Z","2018-12-30T00:00:00.000Z","2018-12-31T00:00:00.000Z","2019-01-01T00:00:00.000Z","2019-01-02T00:00:00.000Z","2019-01-03T00:00:00.000Z","2019-01-04T00:00:00.000Z","2019-01-05T00:00:00.000Z","2019-01-06T00:00:00.000Z","2019-01-07T00:00:00.000Z","2019-01-08T00:00:00.000Z","2019-01-09T00:00:00.000Z"]



Explanation:

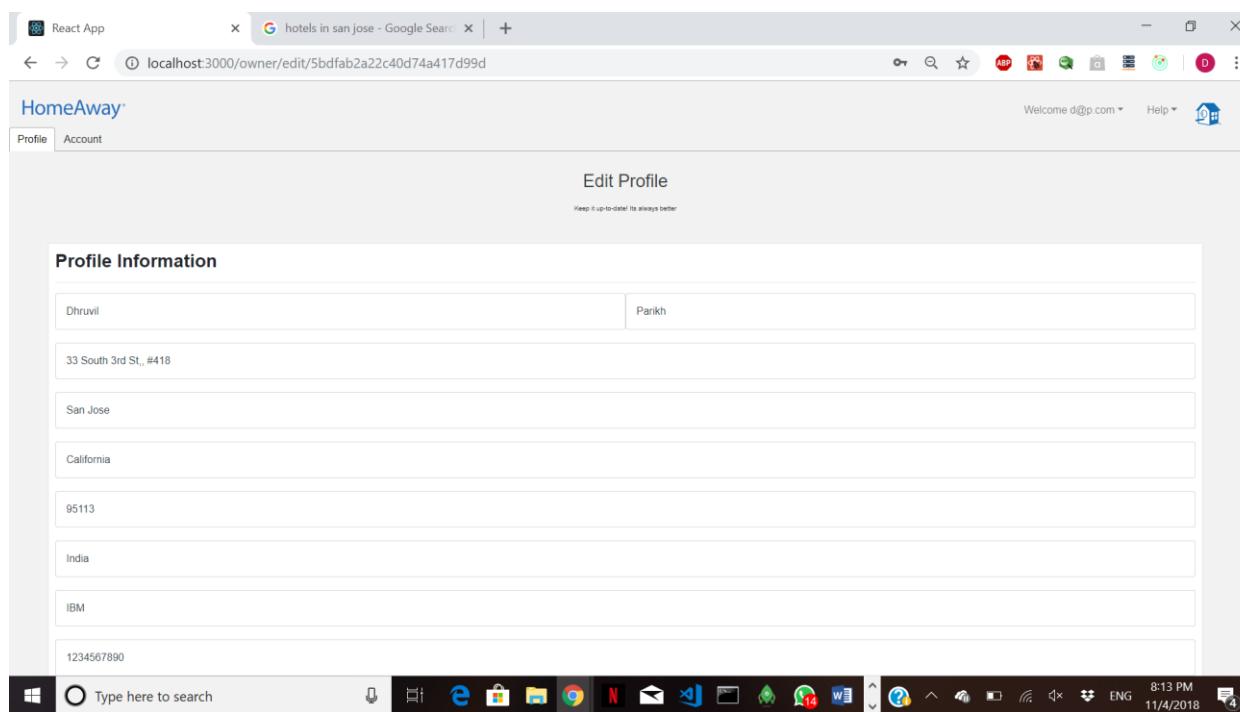
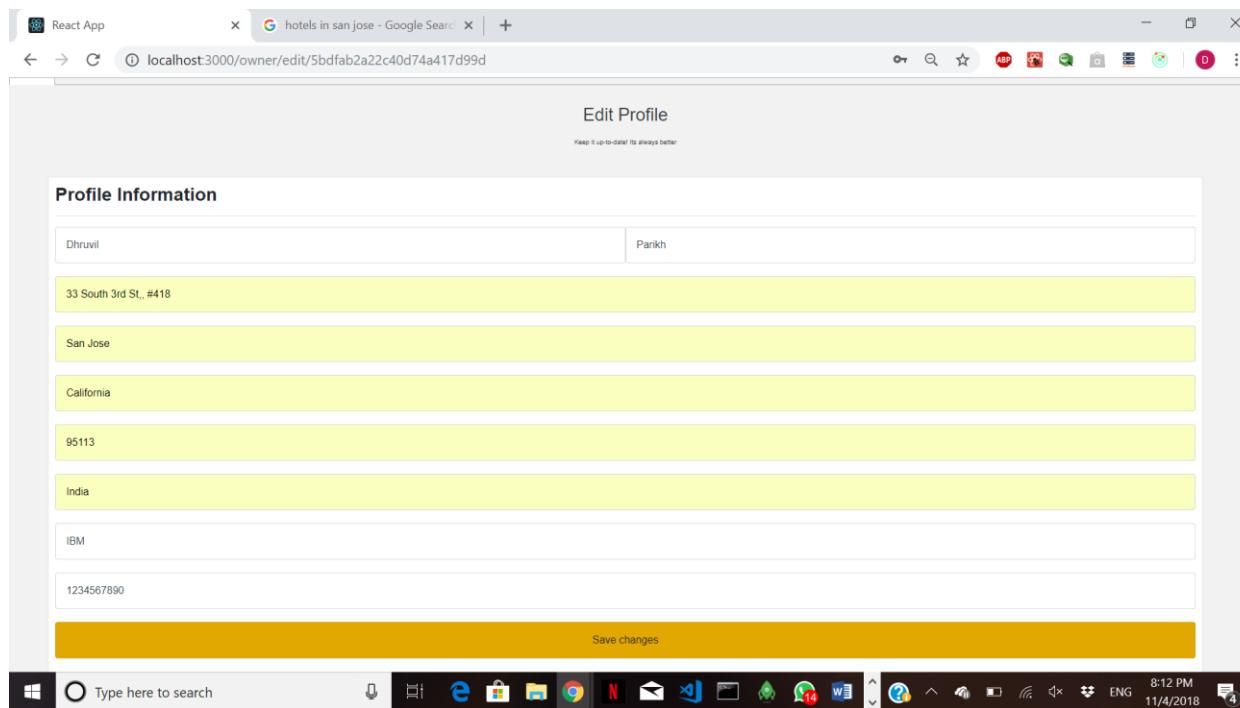
- These screenshots show the steps to edit the property details.
- Here, we are changing the property name from “Courtyard by Marriot San Jose Campbell” to only “Courtyard by Marriot”.
- The redux screenshot also shows us that the property has been correctly edited and the redux store reflects the changing as shown in the screenshot.

Editing the owner's detail like billing address, contact no and other things

A screenshot of a web browser window titled "React App" showing the "hotels in san jose - Google Search" results. The URL is "localhost:3000/owner/home". A context menu is open over the user profile area, listing options: Profile Settings, Property Details, Add new Property, Inbox, and Sign out. A yellow button labeled "Clear Filter" is also visible.



A screenshot of a web browser window titled "React App" showing the "hotels in san jose - Google Search" results. The URL is "localhost:3000/owner/edit/5bdfab2a22c40d74a417d99d". The "Profile" tab is selected. The main content area shows the "Edit Profile" section with the heading "Edit Profile" and the sub-instruction "Keep it up-to-date! Its always better". Below this is a "Profile Information" form containing fields for name, address, city, and state. The name field contains "Dhruvil Parikh". The address, city, and state fields are empty. The Windows taskbar at the bottom shows the date and time as "8:11 PM 11/4/2018".



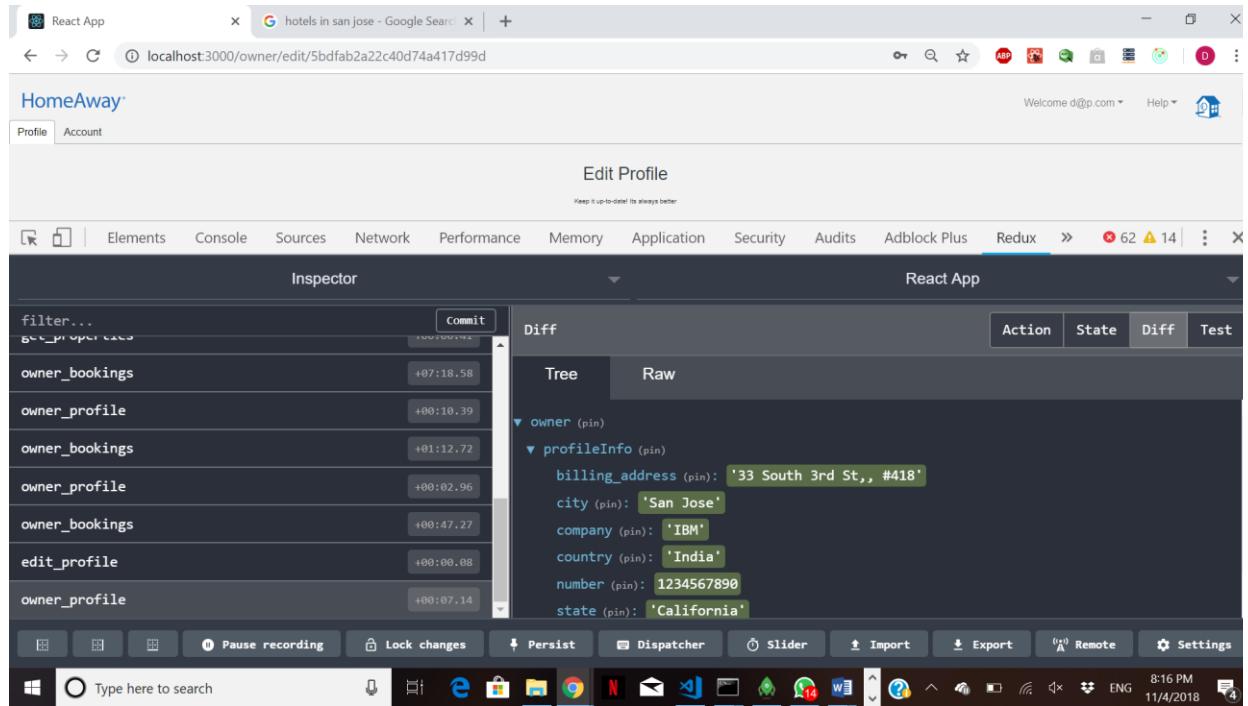
```
componentWillReceiveProps(nextProps) {
  console.log(nextProps.ownerInfo)
  if (nextProps.ownerInfo) {
    localStorage.setItem("ownerlogin",nextProps.ownerInfo._id)
    localStorage.setItem("owneremail",nextProps.ownerInfo.email)
    localStorage.setItem("token",nextProps.ownerInfo.token)
    this.props.history.push('/owner/home')
  } else if(nextProps.error){
  }
}

GET /owner/5bdfab2a22c40d74a417d99d 200 95.123 ms - -
OPTIONS /owner/5bdfab2a22c40d74a417d99d 204 0.269 ms - -
Trying to fetch booking details
in make request
{ owner_id: '5bdfab2a22c40d74a417d99d' }
in response: getOwnerBookings
in response1
true
changing owner details
in make request
{ email: 'd@p.com',
  firstname: 'Dhruvil',
  lastname: 'Parikh',
  company: 'IBM',
  address: '33 South 3rd St,, #418',
  city: 'San Jose',
  state: 'California',
  zipcode: '95113',
  country: 'India',
  number: '1234567890',
  ownerid: '5bdfab2a22c40d74a417d99d' }
in response: editOwnerDetails
in response1
true
in response2
```

Ln 110, Col 13 (55 selected) Spaces: 4 UTF-8 CRLF Javascript (Babel) ESLint ENG 8:15 PM 11/4/2018

```
"{"correlationId":"8a4548ba41663c1bfdef778a2d7505b2","replyTo":"response_topic","data":{"owner_id":"5bdfab2a22c40d74a417d99d}}"
Inside getOwnerDetails kafka backend
after callback
message received for editOwnerDetails { handle_request: [Function: handle_request] }
"{"correlationId":"a28211caf29c751d4f7703ca827b891","replyTo":"response_topic","data":{"email":"d@p.com","firstname":"Dhruvil","lastname":"Parikh","company":"IBM","address":"33 South 3rd St,, #418","city":"San Jose","state":"California","zipcode":"95113","country":"India","number":"1234567890","ownerid":"5bdfab2a22c40d74a417d99d"}}
Inside editOwnerPassword kafka backend
after callback
after handle
{ response_topic: { '0': 1549 } }
after handle[_id: 5bdfab2a22c40d74a417d99d,
  email: 'd@p.com',
  password: '$2a$10$KwhVj9XAnA5opYgGdPqM604kP/9LgLuOyAFPvSr3Az9zIF5Haj0tq',
  firstname: 'Dhruvil',
  lastname: 'Parikh',
  _v: 0,
  bookings: [],
  properties: [ 5bdfbdb509ca266c981c0508 ] }
{ response_topic: { '0': 1550 } }
message received for getOwnerDetails { handle_request: [Function: handle_request] }
"{"correlationId":"0fc0aeb4be75e8dbadfd45b088857edd","replyTo":"response_topic","data":{"ownerid":"5bdfab2a22c40d74a417d99d}}"
Inside getOwnerDetails kafka backend
after callback
after handle[_id: 5bdfab2a22c40d74a417d99d,
```

Ln 110, Col 13 (55 selected) Spaces: 4 UTF-8 CRLF Javascript (Babel) ESLint ENG 8:14 PM 11/4/2018



Explanation:

- These steps show how to change the details after creating an account.
- As you can see in the 2nd screenshot, the details which were already provided during the signup are pre-loaded using `componentDidMount()` lifecycle method.
- We added the basic information which weren't asked while creating the account.
- On saving the changes, the change request is passed through "editOwnerDetails" of the kafka-backend.
- The redux store also shows the difference in store, indicating that the following fields have been added to the owner profile.

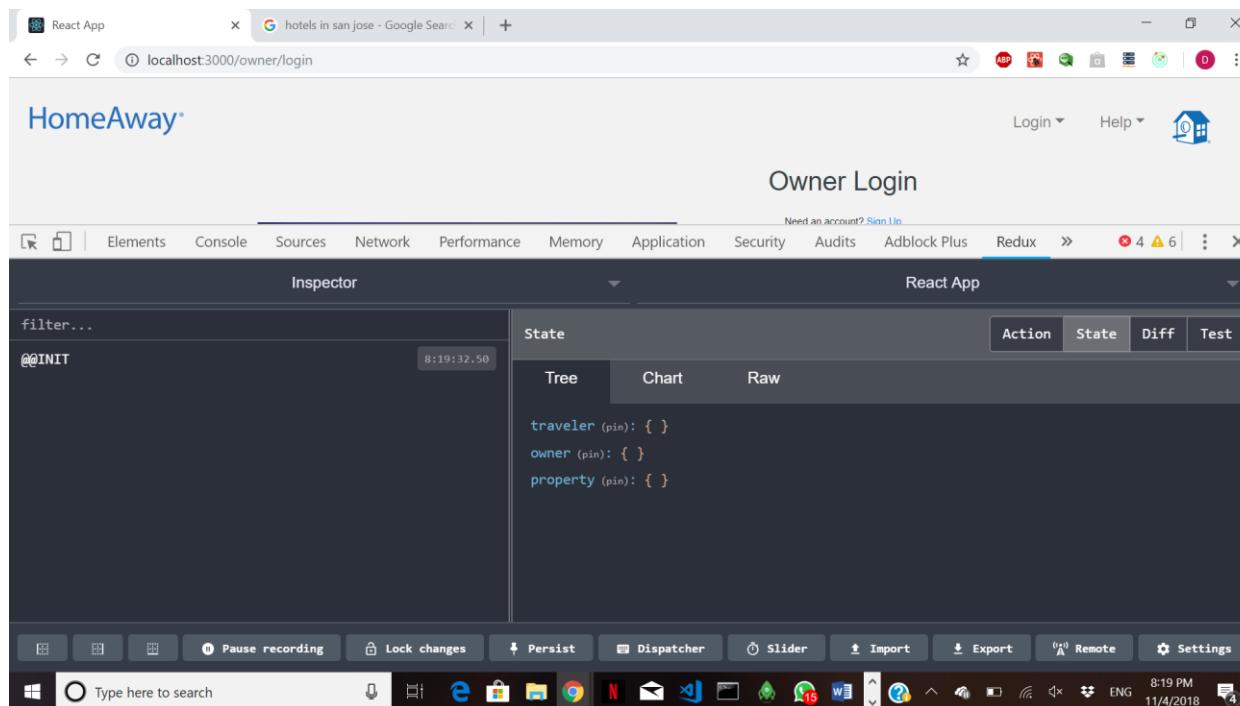
Logging off from the owner account

A screenshot of a web browser window titled "hotels in san jose - Google Search". The address bar shows "localhost:3000/owner/home". The main content area displays the "HomeAway" logo, search filters for "Place Name" and "mm/dd/yyyy", and a "Recent Bookings" section. A user profile dropdown menu is open, showing options like "Profile Settings", "Property Details", "Add new Property", "Inbox", and "Sign out". A yellow "Clear Filter" button is also visible.



A screenshot of a web browser window titled "localhost:3000/owner/login". The main content area features a "Welcome back!" message with a background image of a deck overlooking a forest at sunset. To the right, there is a "Owner Login" form with fields for "Email address" and "Password", a "Forgot password?" link, a "Keep me signed in" checkbox, and a yellow "Log in" button. Above the form, there are links for "Login" and "Help".

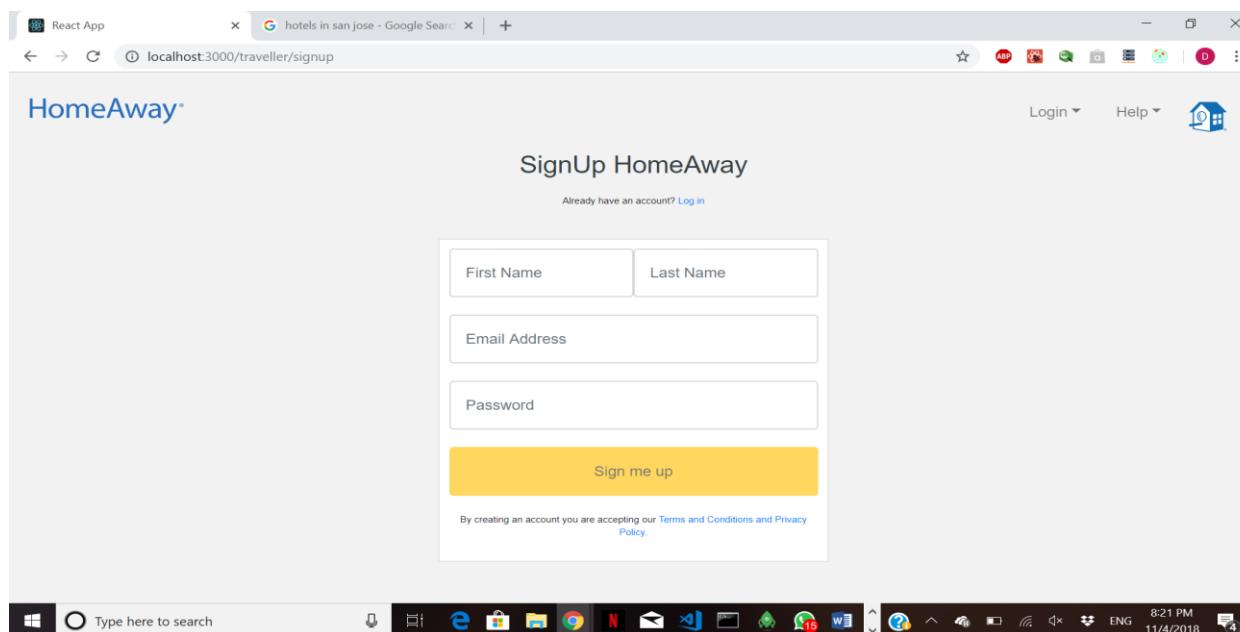


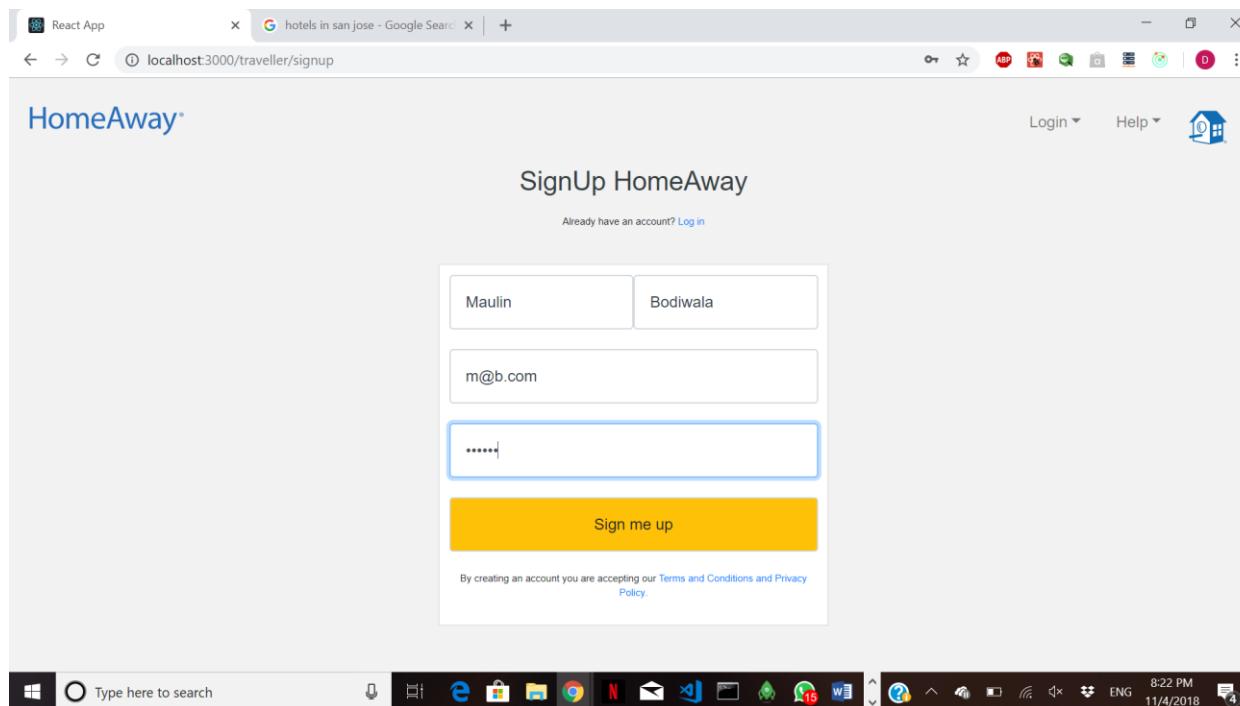


Explanation:

- Once the owner logs out, the owner will be directed to the login page again and the options are gone too.
- Redux store is cleared too.

Creating a new travel account

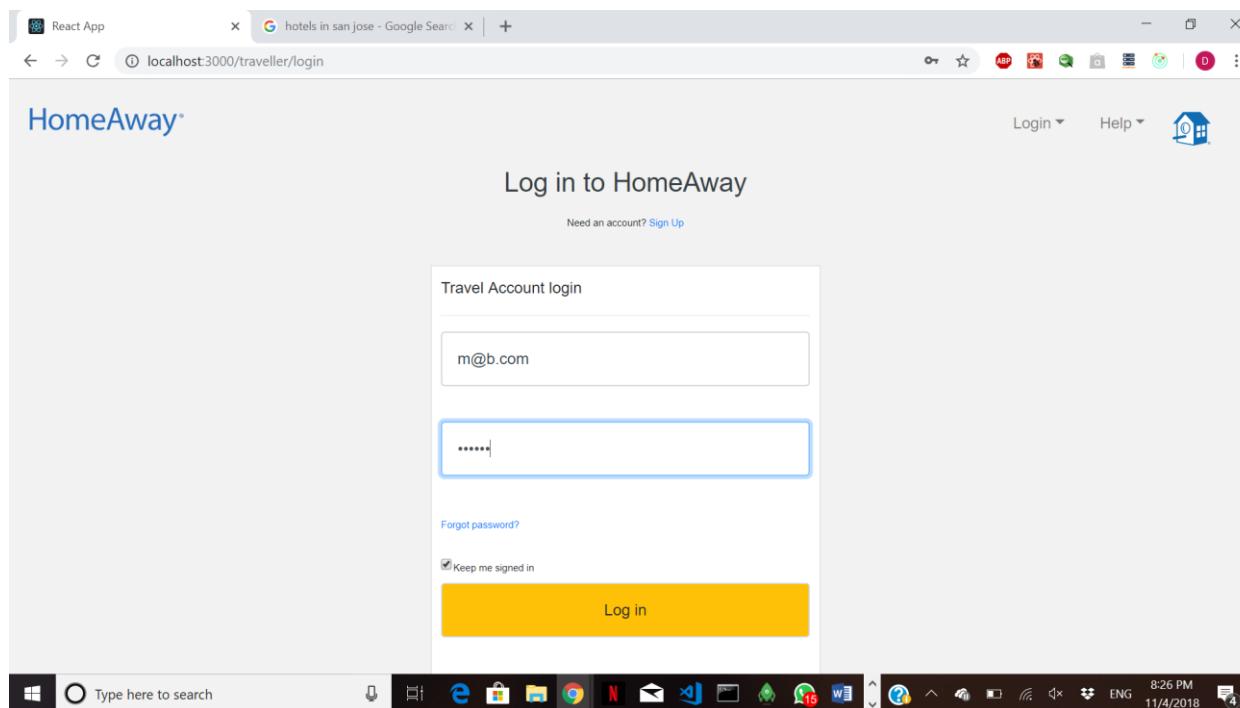


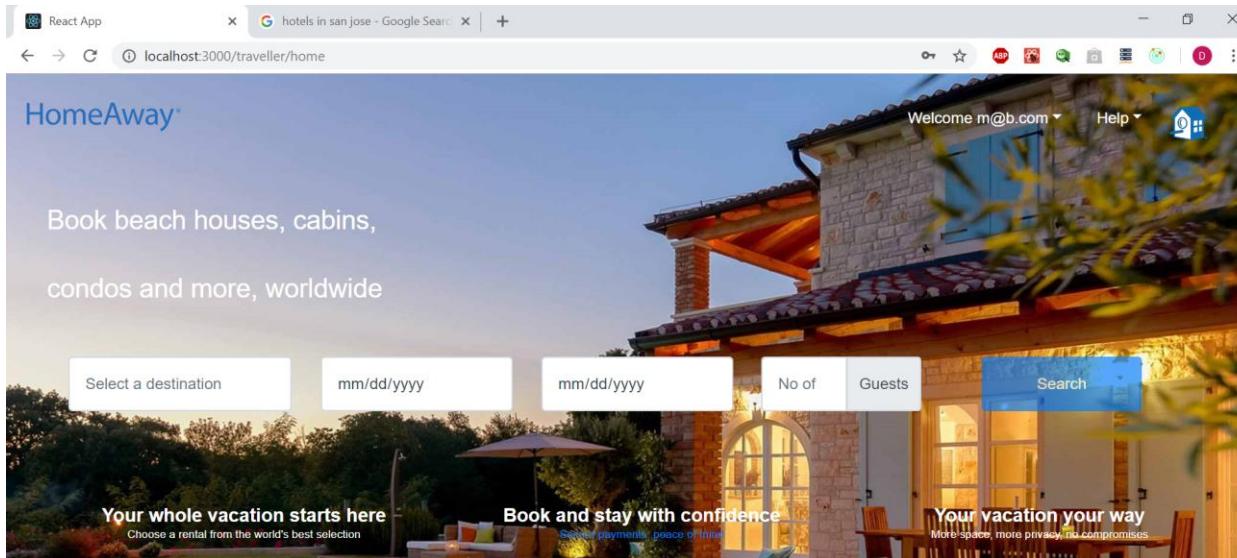


Explanation:

- We created a new travel account with Name: Maulin Bodiwala and Email id: m@b.com and password=maulin which has been encrypted using bcrypt.

Logging in with new travel account





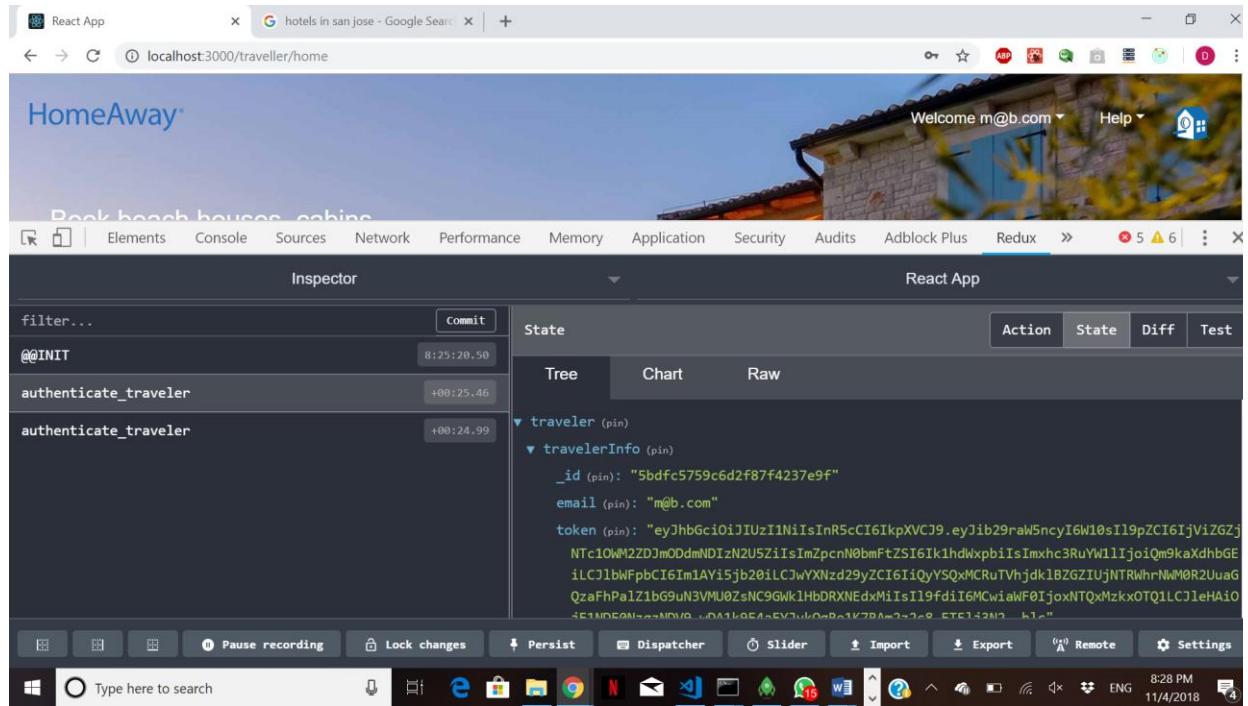
Trending

Explore these trending places

A screenshot of Visual Studio Code. The code editor shows a file named "OwnerHomePage.js" with some JavaScript code. The terminal below shows a sequence of API requests and responses, indicating a login process. The status bar at the bottom of the code editor shows "Ln 107, Col 35" and "Spaces: 4".

```
componentWillReceiveProps(nextProps) {
  console.log(nextProps.ownerInfo)
  if (nextProps.ownerInfo) {
    localStorage.setItem("ownerlogin",nextProps.ownerInfo._id)
    localStorage.setItem("owneremail",nextProps.ownerInfo.email)
    localStorage.setItem("token",nextProps.ownerInfo.token)
    this.props.history.push('/owner/home')
  } else if(nextProps.error) {
    console.log(nextProps.error)
  }
}

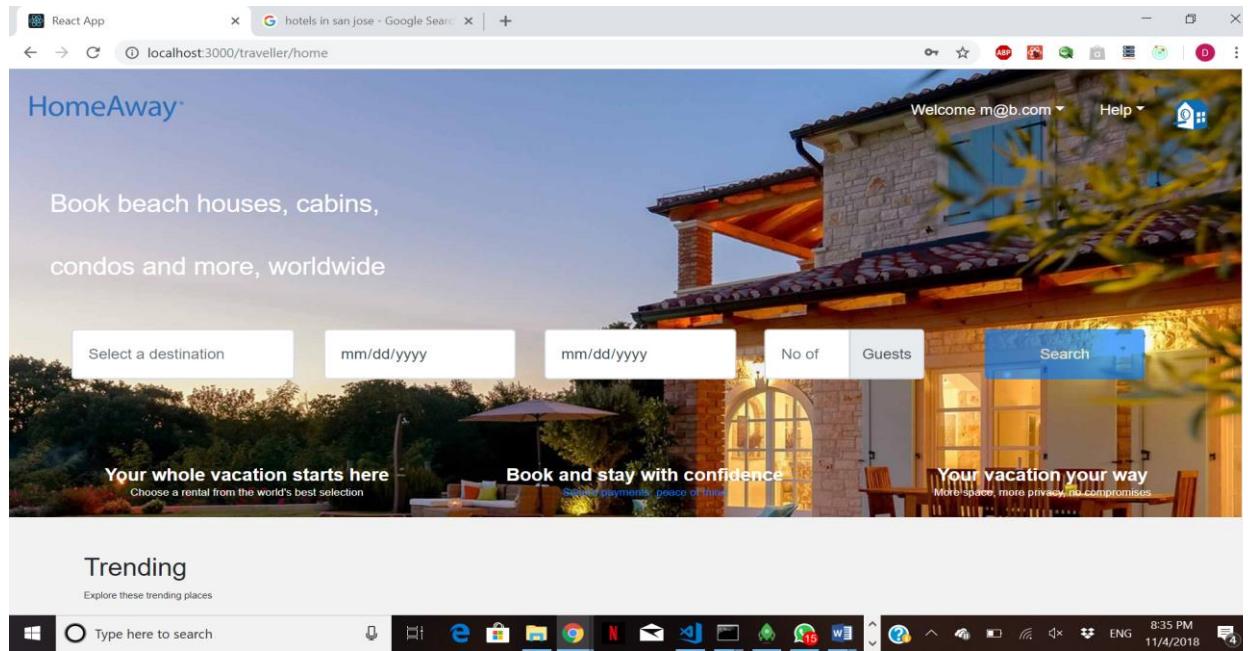
in response1
true
in response2
{ getOwnerBookings: { '0': 69 } }
msg received
Got the booking data for 5bdfab2a22c40d74a417d99d
GET /owner/5bdfab2a22c40d74a417d99d/dashboard 200 98.695 ms - -
OPTIONS /travelsignup 204 0.445 ms - 0
Inside signup Post Request
POST /travelsignup 200 260.805 ms - -
OPTIONS /travellogin 204 0.433 ms - 0
Inside Login Post Request
m@.com maulin
$2a$10$nXcvIAdfHR3SEhk5c4Ge.hd3hXOjVuIon7uLSF14/FZIGl4W4Gq2 maulin
true
password matched
POST /travellogin 200 454.586 ms - -
OPTIONS /travellogin 204 0.418 ms - 0
Inside Login Post Request
m@.com maulin
$2a$10$nXcvIAdfHR3SEhk5c4Ge.hd3hXOjVuIon7uLSF14/FZIGl4W4Gq2 maulin
true
password matched
POST /travellogin 200 448.332 ms - -
```



Explanation:

- We have logged in using the login credentials that we just used for signing up.
- The redux store has stored the traveler's information into its store.

Searching a property based on “city, arrival, departure and # of guests”



React App Google hotels in san jose - Google Search

localhost:3000/traveller/home

Welcome m@b.com Help

HomeAway®

Book beach houses, cabins, condos and more, worldwide

san jose 12/04/2018 12/07/2018 2 Guests Search

Your whole vacation starts here Choose a rental from the world's best selection Book and stay with confidence Secure payments. peace of mind Your vacation your way More space, more privacy, no compromises

Trending

Explore these trending places

Type here to search

This screenshot shows the HomeAway homepage. At the top, there are input fields for location ('san jose'), check-in date ('12/04/2018'), check-out date ('12/07/2018'), number of guests ('2'), and a 'Guests' dropdown. A large search button is visible. Below the search bar is a banner with the text 'Your whole vacation starts here' and 'Choose a rental from the world's best selection'. To the right, another banner says 'Book and stay with confidence' with the subtext 'Secure payments. peace of mind'. Further right is a banner for 'Your vacation your way' with the subtext 'More space, more privacy, no compromises'. The main content area features a large image of a house at sunset. Below the image, a section titled 'Trending' encourages users to explore trending vacation spots.

React App Google hotels in san jose - Google Search

localhost:3000/traveller/show

san jose 12/04/2018 12/07/2018 2 Guests Search

Maximum Price No of Clear Filter

Filter Maximum Price Filter Minimum Bedrooms

Search Results

33 South Third Apartments
Newly Renovated
Description: In the Downtown
Property Details: 2 BR · 2 BA · Sleeps 5
Location, City: San Jose
Base Nightly Rate: \$109 Ask Owner a Question

Timber Leaf Apartments
Newly Renovated
Description: Opposite AMC Theatre
Property Details: 4 BR · 2 BA · Sleeps 7 Ask Owner a Question

Type here to search

This screenshot shows the search results page for the HomeAway application. It displays two property listings: '33 South Third Apartments' and 'Timber Leaf Apartments'. Each listing includes a thumbnail image, the property name, a 'Newly Renovated' badge, a brief description, property details (number of bedrooms and bathrooms), location, and base nightly rate. A blue 'Ask Owner a Question' button is located next to each listing. The search interface at the top is identical to the one in the first screenshot, with fields for location, dates, and guests, along with a search button. The Windows taskbar at the bottom shows various open applications and the system clock.

File Edit Selection View Go Debug Terminal Help OwnerLoginPage.js - Lab2 - Visual Studio Code

```
105     componentWillMount(nextProps) {
106       console.log(nextProps.ownerInfo)
107       if (nextProps.ownerInfo) {
108         localStorage.setItem("ownerlogin",nextProps.ownerInfo._id)
109         localStorage.setItem("owneremail",nextProps.ownerInfo.email)
110         localStorage.setItem("token",nextProps.ownerInfo.token)
111         this.props.history.push('/owner/home')
112       } else if(nextProps.error){
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
in response2
{ searchProperties: { '0': 1112 } }
msg received
POST /property/search 200 102.727 ms -
GET /public/uploads/property-5bdf34410f120d226e392721-wm5.jpg 404 5.355 ms - 195
GET /public/uploads/property-5bdf34410f120d226e392721-wm3.jpg 404 2.452 ms - 195
GET /public/uploads/property-5bdf34410f120d226e392721-wm6.jpg 404 4.520 ms - 195
OPTIONS /property/search 204 0.190 ms - 0
inside post search
[ 2018-12-04T00:00:00.000Z,
  2018-12-05T00:00:00.000Z,
  2018-12-06T00:00:00.000Z ]
in make request
{ place: 'san jose',
  available_from: '2018-12-04',
  available_to: '2018-12-06',
  accommodates: 3,
  range:
    [ 2018-12-04T00:00:00.000Z,
      2018-12-05T00:00:00.000Z,
      2018-12-06T00:00:00.000Z ] }
in response: searchProperties
in response1
true
in response2
```

Ln 107, Col 35 Spaces: 4 UTF-8 CRLF Javascript (Babel) ESLint 8:36 PM 11/4/2018

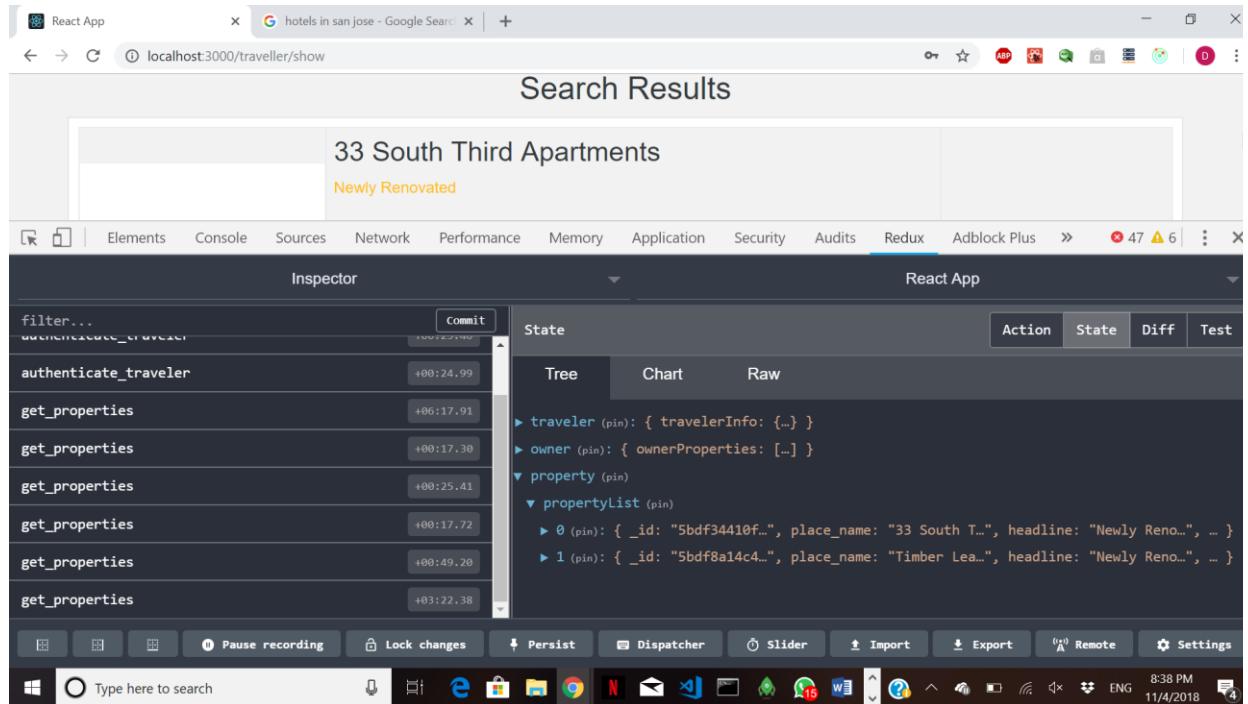
File Edit Selection View Go Debug Terminal Help OwnerLoginPage.js - Lab2 - Visual Studio Code

```
105     componentWillMount(nextProps) {
106       console.log(nextProps.ownerInfo)
107       if (nextProps.ownerInfo) {
108         localStorage.setItem("ownerlogin",nextProps.ownerInfo._id)
109         localStorage.setItem("owneremail",nextProps.ownerInfo.email)
110         localStorage.setItem("token",nextProps.ownerInfo.token)
111         this.props.history.push('/owner/home')
112       } else if(nextProps.error){
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
message received for searchProperties { handle_request: [Function: handle_request] }
{"correlationId":"f31301c8e2463cb7e09932730636d4e3","replyTo":"response_topic","data": {"place": "san jose", "available_from": "2018-12-04", "available_to": "2018-12-07", "accommodates": 2, "range": ["2018-12-04T00:00:00.000Z", "2018-12-05T00:00:00.000Z", "2018-12-06T00:00:00.000Z", "2018-12-07T00:00:00.000Z"]}}
Inside getOwnerDetails kafka backend
booking criteria: [ '2018-12-04T00:00:00.000Z',
  '2018-12-05T00:00:00.000Z',
  '2018-12-06T00:00:00.000Z',
  '2018-12-07T00:00:00.000Z' ]
after callback
before handle{_id: 5bdf34410f120d226e392721,
  place_name: '33 South Third Apartments',
  headline: 'Newly Renovated',
  description: 'In the Downtown',
  street: '33 South 3rd St',
  apt: '#418',
  location_city: 'San Jose',
  state: 'CA',
  zipcode: '95113',
  country: 'United States',
  available_from: 2018-11-05T00:00:00.000Z,
  available_to: 2019-04-30T00:00:00.000Z,
  bedrooms: 2,
  bathrooms: 2,
  accommodates: 5,
```

Ln 107, Col 35 Spaces: 4 UTF-8 CRLF Javascript (Babel) ESLint 8:37 PM 11/4/2018



Explanation:

- As you can see in the first screenshot, because we haven't entered all data into the form, the search button is disabled and will not let us access the search button.
- After we provide all the details(as done in the 2nd screenshot), the search button will get enabled and we will get our results.
- The console output shows the topic which is being used to search the properties, as, we have to take care of all the details provided and plus, we have to make sure that the dates provided are not booked already.
- After we click on search, we will be directed to a page which will show the search results. But, notice how the inline form has the search parameters intact even after we have directed to the next page. This was persisted using redux store.
- The redux store also has stored the properties that we just searched.

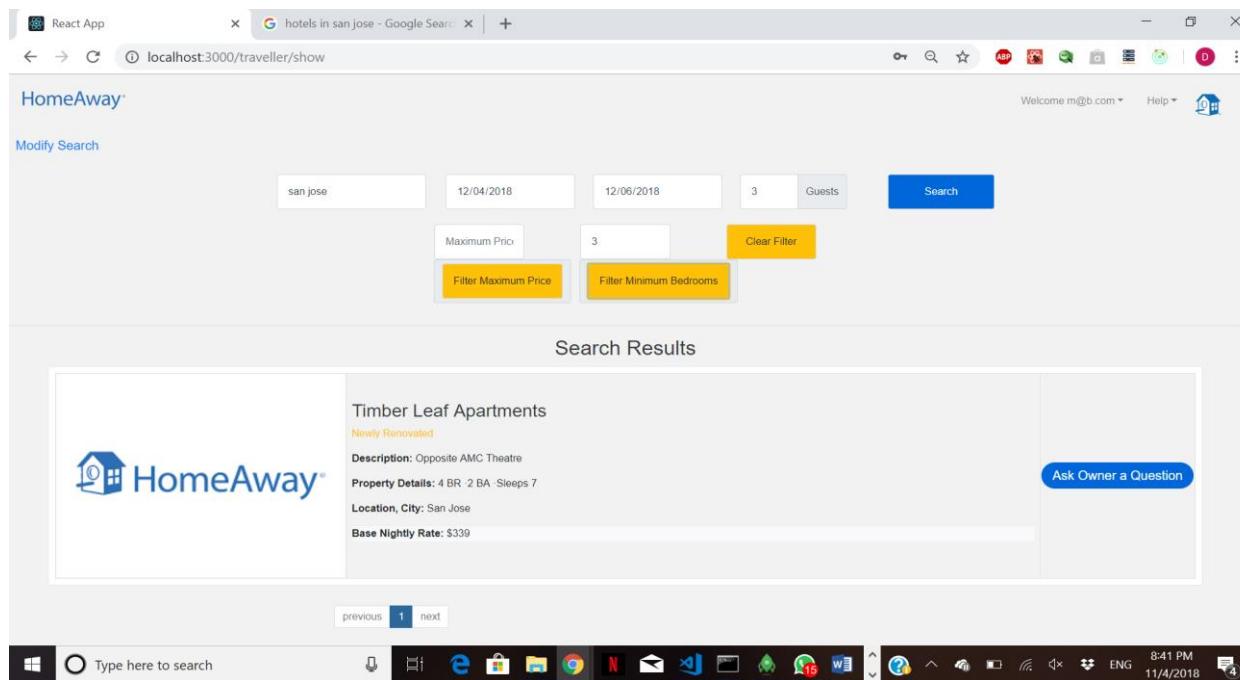
Displaying the result of the above search and filtering through it

The screenshot shows a web browser window with the URL localhost:3000/traveller/show. The search bar contains "san jose", "12/04/2018", "12/06/2018", "3 Guests", and a "Search" button. Below the search bar are filters for "Maximum Price" (dropdown), "No of" (dropdown), and "Clear Filter". Buttons for "Filter Maximum Price" and "Filter Minimum Bedrooms" are also present. The main area is titled "Search Results" and lists two properties:

- 33 South Third Apartments**
Newly Renovated
Description: In the Downtown
Property Details: 2 BR, 2 BA, Sleeps 5
Location, City: San Jose
Base Nightly Rate: \$109
- Timber Leaf Apartments**
Newly Renovated
Description: Opposite AMC Theatre
Property Details: 4 BR, 2 BA, Sleeps 7
Location, City: San Jose
Base Nightly Rate: \$339

The browser interface includes a taskbar at the bottom with various icons and a system tray on the right showing the date and time.

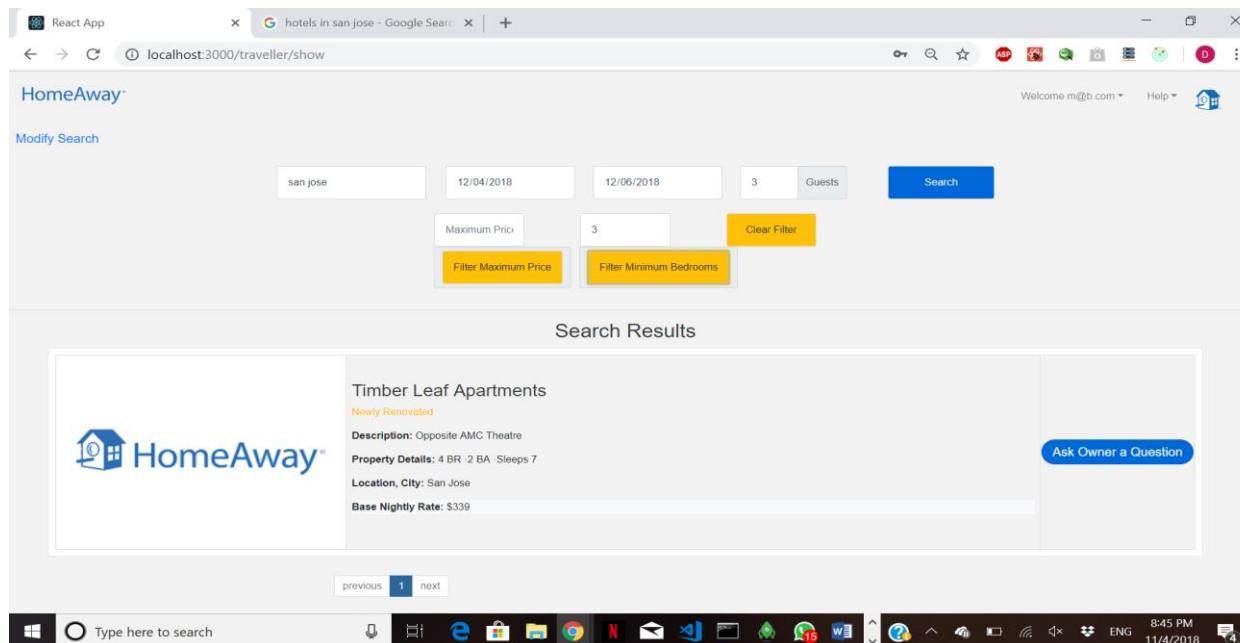
This screenshot shows the same web browser setup as the first one, but with a different filter applied. The "No of" dropdown in the search bar is set to "3". The search results page now only displays the "33 South Third Apartments" listing, as the "Timber Leaf Apartments" listing no longer meets the minimum bedroom requirement.

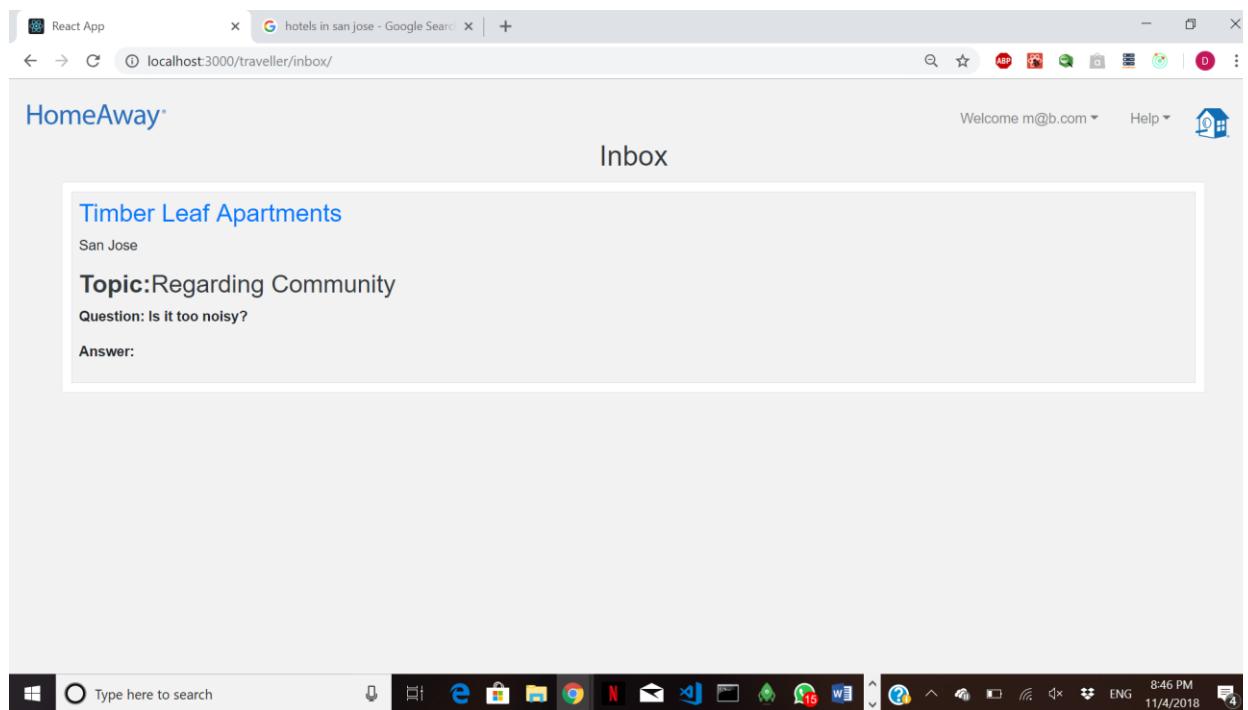
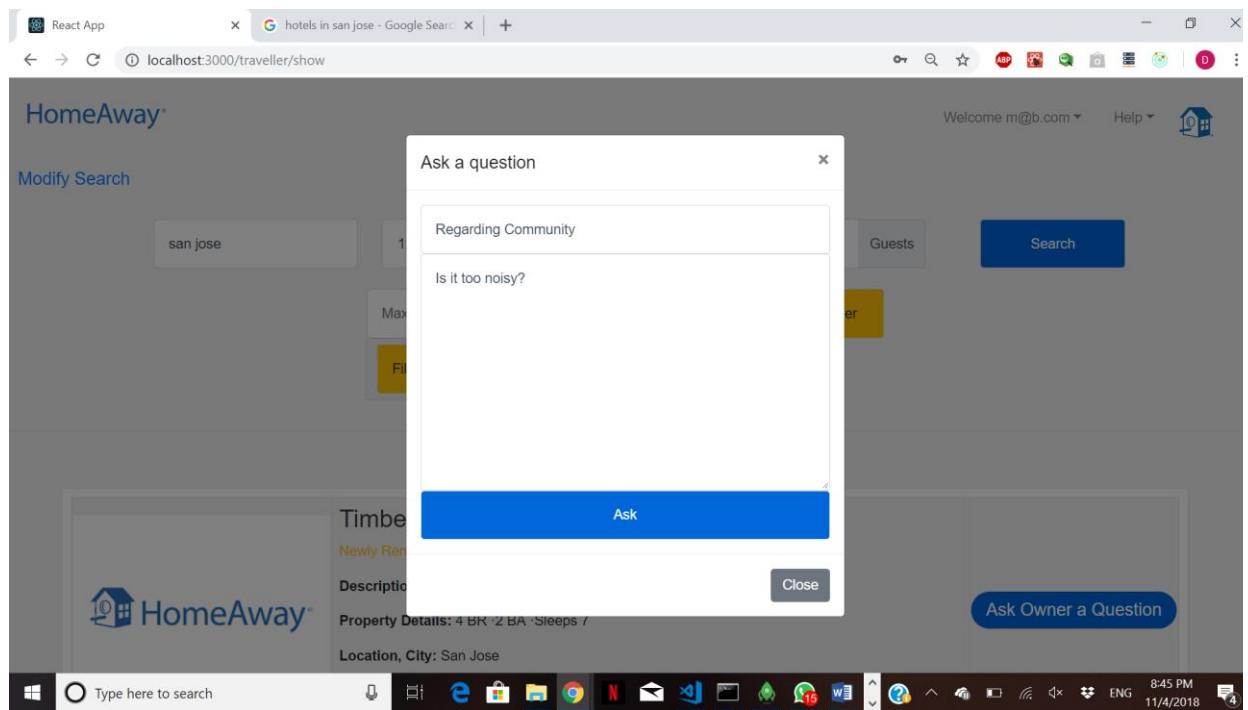


Explanation:

- Here, we are filtering the results to get all the properties which has more than 3 bedrooms.
- As shown in the screenshot, the only property in the searched list having more than 3 bedrooms is Timber Leaf Apartments and thus it is returned.
- The filtering is done by ‘_’ operator imported from lodash library.

Asking Owner the question on any of the property searched





The screenshot shows a Visual Studio Code interface with multiple tabs open. The active tab is `OwnerLoginPage.js`. The code in this file includes logic for handling props and setting local storage items like `ownerlogin`, `owneremail`, and `token`. The terminal below shows several log entries from 2019-02-11T00:00:00Z to 2019-02-17T00:00:00Z, indicating messages received for `postUserQuestion`, `getUserInbox`, and `getTravelerInbox` requests. The logs also mention a `posting user question under postUserQuestion kafka request` and a `getting user inbox under getUserInbox kafka request`.

```
componentWillReceiveProps(nextProps) {
  console.log(nextProps.ownerInfo)
  if (nextProps.ownerInfo) {
    localStorage.setItem("ownerlogin",nextProps.ownerInfo._id)
    localStorage.setItem("owneremail",nextProps.ownerInfo.email)
    localStorage.setItem("token",nextProps.ownerInfo.token)
    this.props.history.push('/owner/home')
  } else if(nextProps.error){
    // handle error
  }
}

2019-02-11T00:00:00Z,
2019-02-12T00:00:00Z,
2019-02-13T00:00:00Z,
2019-02-14T00:00:00Z,
2019-02-15T00:00:00Z,
2019-02-16T00:00:00Z,
... 43 more items ]
{ response_topic: { '0': 1558 } }

message received for postUserQuestion { handle_request: [Function: handle_request] }
{"correlationId":"72b17c5688cdabc933933fa79b19b4","replyTo":"response_topic","data":{"_id":"5bdf8a14c4691677acd31454","topic":"Regarding Community","question":"Is it too noisy?","travelerId":"5bdfc5759c6d2f87f4237e9f"}}

posting user question under postUserQuestion kafka request
after callback
5bdf33bbf8d8102258c9dd39
after handle{ _v: 0,
  travel: 5bdfc5759c6d2f87f4237e9f,
  property: 5bdf8a14c4691677acd31454,
  owner: 5bdf33bbf8d8102258c9dd39,
  topic: 'Regarding Community',
  question: 'Is it too noisy?',
  _id: 5bdcfb0b09ca266c981c0309
} { response_topic: { '0': 1559 } }

message received for getUserInbox { handle_request: [Function: handle_request] }
{"correlationId":"0e4418db73e601b57a83486d56c0b33c","replyTo":"response_topic","data":{"travelerId":"5bdfc5759c6d2f87f4237e9f"}}

getting user inbox under getUserInbox kafka request
```

Explanation:

- As shown in the screenshot, we asked owner the question about the community.
- The inbox of the traveler shows that the question has been posted.
- The answer is not there, as for that the owner will have to reply first. We'll do that later.
- The redux store is not storing the questions as those aren't needed to pass to any other component.

Selecting a property from search result and booking it with the travel account

Let's book Timber-Leaf Apartments

The screenshot shows a search interface for "hotels in san jose - Google Search". At the top, there are input fields for "san jose", "12/04/2018", "12/06/2018", "3 Guests", and a "Search" button. Below these are filters for "Maximum Price" (with a dropdown set to "3") and "Clear Filter". There are also buttons for "Filter Maximum Price" and "Filter Minimum Bedrooms". The main area is titled "Search Results" and displays a listing for "Timber Leaf Apartments" from HomeAway. The listing includes a thumbnail, the property name, a "Newly Renovated" badge, a description ("Opposite AMC Theatre"), property details ("4 BR · 2 BA · Sleeps 7"), location ("San Jose"), and a base nightly rate of "\$339". A blue button "Ask Owner a Question" is visible. At the bottom of the results page, there are navigation buttons for "previous", "1", and "next". The browser status bar at the bottom indicates "localhost:3000/traveller/show#".

The screenshot shows the property details for "Timber Leaf Apartments" from "localhost:3000/traveller/property/show". On the left is a large image of a city skyline at night. To the right is a "BOOKING OVERVIEW" sidebar with the following information:

- Arrival: 2018-12-04
- Departure: 2018-12-06
- Base rate/night: \$339
- Total Nights: 2

A red text box highlights the total cost: "Total to be paid at checkout \$678". A yellow "Book Now" button is at the bottom. The browser status bar at the bottom indicates "localhost:3000/traveller/property/show".

React App x Google hotels in san jose - Google Search | +

localhost:3000/traveller/property/show

Timber Leaf Apartments

Newly Renovated

Details

Town House Sleeps 7 Bedrooms 4 Bathrooms: 2 Stay: 2

About the property

Opposite AMC Theatre

What's nearby:

This house is located near the North Park neighborhood, where you'll find a bustling collection of cafés, bars, and restaurants. It is also a short drive to the San Diego Zoo, the Air & Space Museum, Morley Field Sports Complex, and Balboa Park, among other attractions. The beach is also within easy driving distance.

Amenities

Property Type: house

Floor Area: 468sq ft.

House Rules Check-in: 4:00 PM / Check-out: 11:00 AM Parties/events not allowed

Type here to search

React App x Google hotels in san jose - Google Search | +

localhost:3000/traveller/property/show



Booking Overview

Arrival:
2018-12-04

Departure:
2018-12-06

Base rate/night:
\$339

Total Nights:
2

Total to be paid at checkout
\$678

Book Now

Type here to search

The screenshot shows a web browser window with the URL localhost:3000/traveller/edit/5bdfc5759c6d2f87f4237e9f. The page title is "React App" and the search bar contains "hotels in san jose - Google Search". The main content area is titled "HomeAway*" and includes tabs for "My Trips", "Profile", and "Account". There are input fields for "Place Name" and "mm/d", a "Clear Filter" button, and two yellow buttons: "Filter Place by Name" and "Filter By Date". Below this is a section titled "Recent Trips" featuring a card for "Timber Leaf Apartments" located in San Jose, which is "Newly Renovated" from Tuesday, December 4, 2018, to Thursday, December 6, 2018, for 3 guests at a base nightly rate of \$339.

The screenshot shows the Visual Studio Code interface with multiple tabs open: "details.js", "OwnerHomePage.js", "Inbox.js", "EditProfile.js", "index.js", "OwnerLoginPage.js", "LoginPage.js", "Main.js", and "ListPlaces.js". The "OwnerLoginPage.js" tab is active, displaying code related to handling props and localStorage. The "TERMINAL" tab shows a sequence of API requests and responses, including:

```
componentWillReceiveProps(nextProps) {
  console.log(nextProps.ownerInfo)
  if (nextProps.ownerInfo) {
    localStorage.setItem("ownerlogin",nextProps.ownerInfo._id)
    localStorage.setItem("owneremail",nextProps.ownerInfo.email)
    localStorage.setItem("token",nextProps.ownerInfo.token)
    this.props.history.push('/owner/home')
  }
}

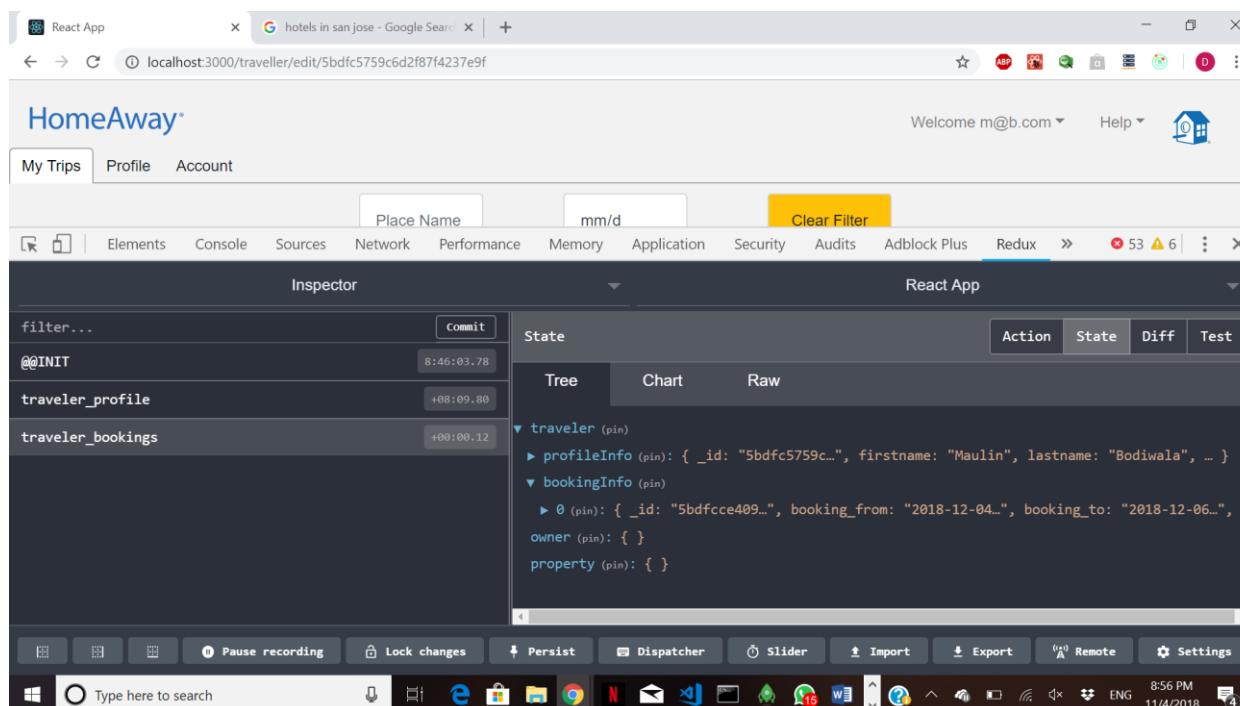
true
in response2
{ bookProperty: { '0': 7 } }
msg received
POST /property/5bdf8a14c4691677acd31454/book 200 173.289 ms -
in make request
{ travelId: '5bdfc5759c6d2f87f4237e9f' }
in response: getUserDetails
in response1
true
Trying to fetch booking details
in make request
{ travel_id: '5bdfc5759c6d2f87f4237e9f' }
in response: getUserBookings
in response1
true
in response2
{ getUserDetails: { '0': 104 } }
in response2
{ getUserBookings: { '0': 91 } }
msg received
GET /travel/5bdfc5759c6d2f87f4237e9f 200 99.266 ms -
msg received
GET /travel/5bdfc5759c6d2f87f4237e9f/bookingdetails 200 216.691 ms -
```

```

105     componentWillReceiveProps(nextProps) {
106       console.log(nextProps.ownerInfo)
107       if (nextProps.ownerInfo) {
108         localStorage.setItem("ownerlogin",nextProps.ownerInfo._id)
109         localStorage.setItem("owneremail",nextProps.ownerInfo.email)
110         localStorage.setItem("token",nextProps.ownerInfo.token)
111         this.props.history.push('/owner/home')
112       } else if(nextProps.error){
113     }
114   }
115
116   PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
117
118   2019-02-14T00:00:00.000Z,
119   2019-02-15T00:00:00.000Z,
120   2019-02-16T00:00:00.000Z,
121   ... 43 more items ...
122   { response_topic: { '0': 1561 } }
123   message received for bookProperty { handle_request: [Function: handle_request] }
124   {"correlationId": "a9bf0397472304a74a56ffbb8430ad4", "replyTo": "response_topic", "data": {"_id": "5bdf8a14c4691677acd31454", "travel_id": "5bdfc5759c6d2f87f4237e9f", "booking_from": "2018-12-04", "booking_to": "2018-12-06", "guests": 3, "range": ["2018-12-04T00:00:00.000Z", "2018-12-05T00:00:00.000Z"], "propertyid": "5bdf8a14c4691677acd31454"}}
125   Inside getOwnerDetails kafka backend
126   after callback
127   after handle{ _v: 0,
128     booking_from: 2018-12-04T00:00:00.000Z,
129     booking_to: 2018-12-06T00:00:00.000Z,
130     guests: 3,
131     property: 5bdf8a14c4691677acd31454,
132     traveler: 5bdfc5759c6d2f87f4237e9f,
133     owner: 5bdf33bbf8d102258c9dd39,
134     _id: 5bdfcce409ca266c981c050a }
135   { response_topic: { '0': 1562 } }
136   message received for getUserDetails { handle_request: [Function: handle_request] }
137   {"correlationId": "476242af9f7a4a07aa8b74b7cba6a45", "replyTo": "response_topic", "data": {"travelid": "5bdfc5759c6d2f87f4237e9f"}}
138   Inside editUserDetails kafka backend
139   5bdfc5759c6d2f87f4237e9f
140   after callback

```

Ln 107, Col 35 Spaces: 4 UTF-8 CRLF Javascript (Babel) ESLint 8:55 PM 11/4/2018



Explanation:

- The above screenshots show the details of the property that we just selected i.e "Timber Leaf Apartments."
- On the side, we can re-confirm our booking details and after clicking on book, we book that property for that date and for that user.
- The console output and the redux store will certify that thing.

Editing the basic details of travel account

The screenshot shows a web browser window with the URL localhost:3000/traveller/edit/5bdxfc5759c6d2f87f4237e9f. The page title is "Edit Profile". The "Profile Information" section contains four input fields: "Name" (Maulin Bodiwala), "city", "school", and "company". The browser taskbar at the bottom shows various pinned icons and the system tray indicates the date and time as 9:01 PM on 11/4/2018.

The screenshot shows the same web browser window after changes have been made. The "Profile Information" section now contains different values: "Name" (Surat SJSU), "city" (Google), "school" (2222222222), and "company" (I'll tell later). A dropdown menu for gender has been opened, showing "Male" as the selected option. A yellow "Save changes" button is visible at the bottom of the form. The browser taskbar and system tray remain the same as in the previous screenshot.

React App Google hotels in san jose - Google Search

localhost:3000/traveller/edit/5bdfc5759c6d2f87f4237e9f

Welcome m@b.com Help

HomeAway

My Trips Profile Account

Account Settings

Keep it up-to-date! It's always better

Example placeholder avatar
Maulin Bodiwala

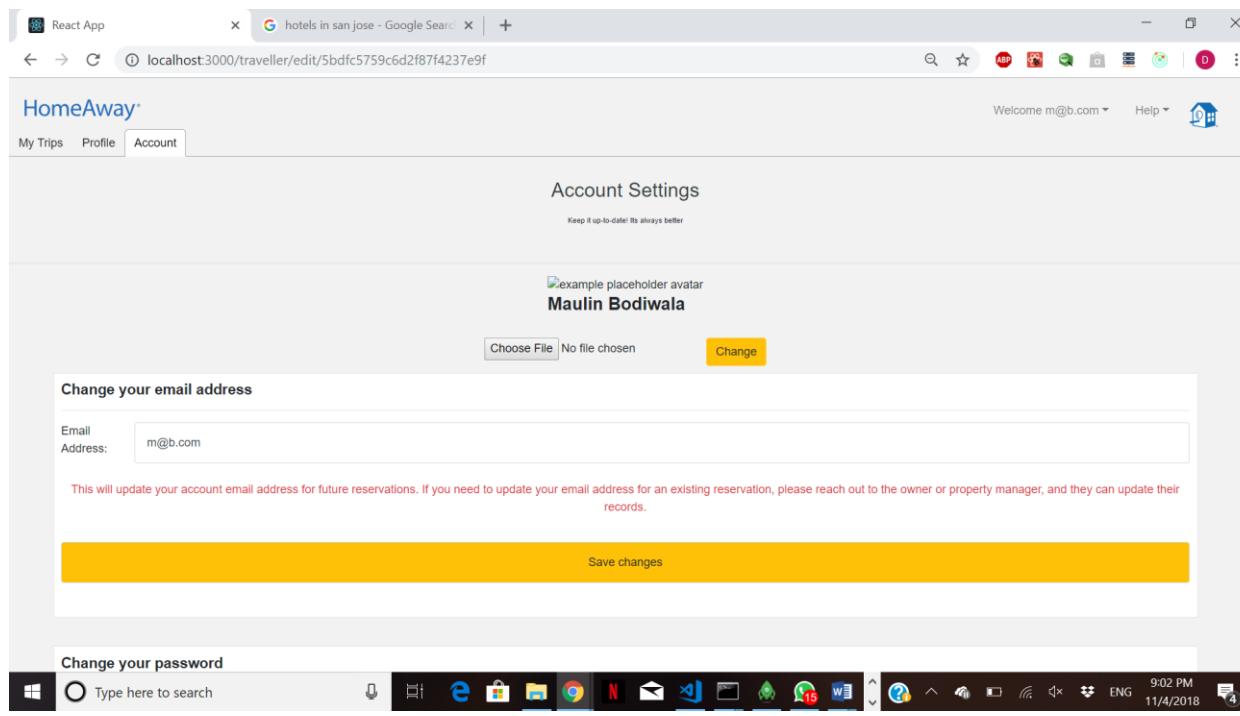
Choose File No file chosen Change

Change your email address

Email Address: m@b.com

This will update your account email address for future reservations. If you need to update your email address for an existing reservation, please reach out to the owner or property manager, and they can update their records.

Save changes



React App Google hotels in san jose - Google Search

localhost:3000/traveller/edit/5bdfc5759c6d2f87f4237e9f

Welcome m@b.com Help

HomeAway

My Trips Profile Account

Account Settings

Keep it up-to-date! It's always better

Example placeholder avatar
Maulin Bodiwala

Choose File No file chosen Change

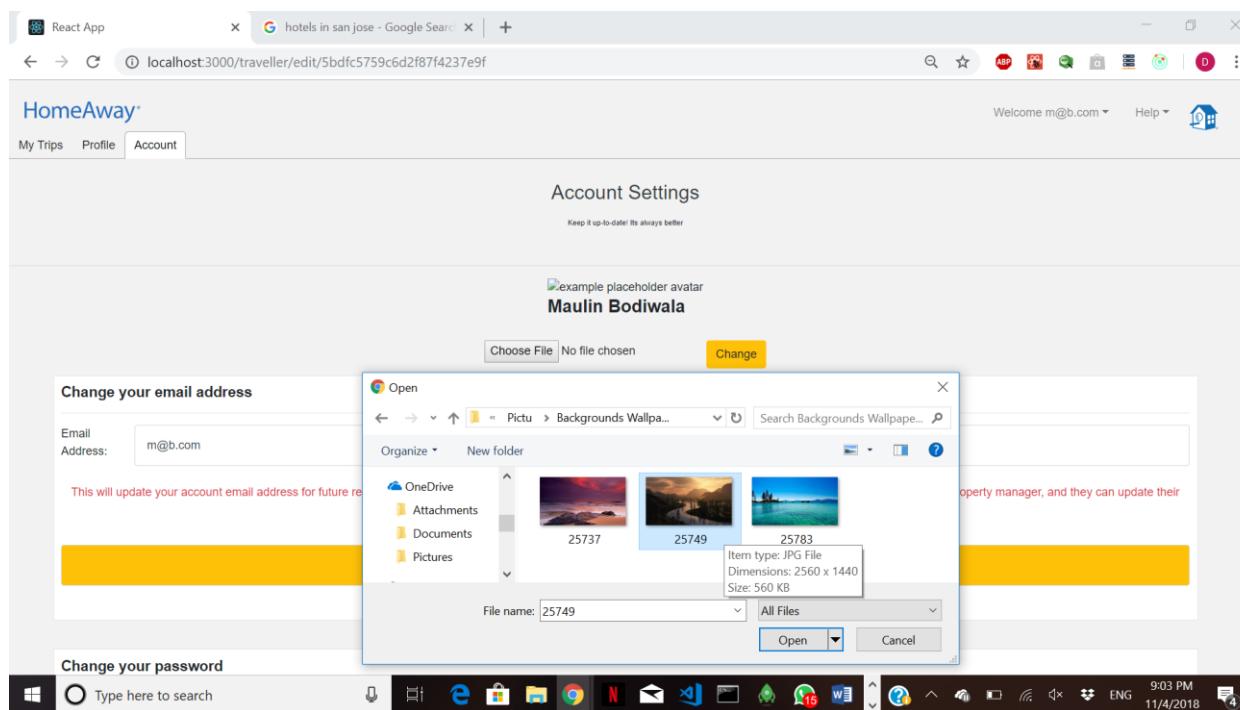
Change your email address

Email Address: m@b.com

This will update your account email address for future reservations. If you need to update your email address for an existing reservation, please reach out to the owner or property manager, and they can update their records.

Save changes

Change your password



A file selection dialog is open, showing a folder structure with 'Pictures' selected. It displays three images: '25737', '25749' (selected), and '25783'. A tooltip provides details: 'Item type: JPG File', 'Dimensions: 2560 x 1440', and 'Size: 560 KB'. The 'File name:' field shows '25749'. Buttons for 'Open' and 'Cancel' are at the bottom.

The screenshot shows a web browser window with the URL `localhost:3000/traveller/edit/5bdfc5759c6d2f87f4237e9f`. The page title is "HomeAway". The main content is titled "Account Settings" with a sub-instruction "Keep it up-to-date! It's always better". It features a placeholder profile picture with the name "Maulin Bodiwala". Below this is a file input field with the placeholder "Choose File No file chosen" and a "Change" button. A section titled "Change your email address" contains an input field with the value "m@b.com". A note below states: "This will update your account email address for future reservations. If you need to update your email address for an existing reservation, please reach out to the owner or property manager, and they can update their records." At the bottom is a yellow "Save changes" button.

The screenshot shows the Visual Studio Code interface with multiple tabs open: "OwnerLoginPage.js - Lab2 - Visual Studio Code", "index.js", "EditProfile.js", "Inbox.js", "OwnerHomePage.js", "details.js", "LoginPage.js", "Main.js", and "ListPlaces.js". The "details.js" tab is active, displaying the following code:

```
105 componentWillReceiveProps(nextProps) {
106   console.log(nextProps.ownerInfo)
107   if (nextProps.ownerInfo) {
108     localStorage.setItem("ownerlogin",nextProps.ownerInfo._id)
109     localStorage.setItem("owneremail",nextProps.ownerInfo.email)
110     localStorage.setItem("token",nextProps.ownerInfo.token)
111     this.props.history.push('/owner/home')
112   }else if(nextProps.error){}
```

The "TERMINAL" tab shows the following log output:

```
in response2
{ getUserDetails: { '0': 106 } }
in response2
{ getUserBookings: { '0': 93 } }
msg received
GET /travel/5bdfc5759c6d2f87f4237e9f 200 94.424 ms -
msg received
GET /travel/5bdfc5759c6d2f87f4237e9f/bookingdetails 200 194.449 ms -
trying to upload a file
m@b.com
25749.jpg
public/uploads/userprofile-5bdfc5759c6d2f87f4237e9f.jpg
POST /travel/5bdfc5759c6d2f87f4237e9f/upload 200 152.539 ms -
in make request
{ travelId: '5bdfc5759c6d2f87f4237e9f' }
in response: getUserDetails
in response1
true
Trying to fetch booking details
in make request
{ travel_id: '5bdfc5759c6d2f87f4237e9f' }
in response: getUserBookings
in response1
true
in response2
```

The status bar at the bottom indicates "Ln 111, Col 51" and "Spaces: 4".

```

105     componentWillReceiveProps(nextProps) {
106       console.log(nextProps.ownerInfo)
107       if (nextProps.ownerInfo) {
108         localStorage.setItem("ownerlogin",nextProps.ownerInfo._id)
109         localStorage.setItem("owneremail",nextProps.ownerInfo.email)
110         localStorage.setItem("token",nextProps.ownerInfo.token)
111         this.props.history.push(['owner/home'])
112     }else if(nextProps.error){

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```

5bdfcce409ca266c981c050a ],
properties: [ 5bdf34410f12ed226e392721, 5bdf8a14c4691677acd31454 ],
_v: 0
message received for editUserDetails { handle_request: [Function: handle_request] }
{"correlationId":"e13f90cc1f3ebc35b152d5aa6881ea8","replyTo":"response_topic","data":{ "email":"m@b.com","firstname":"Maulin","lastname":"Bodivala","school":"SJSU","company":"Google","address":"Surat","number":"2222222222","aboutme":"I'll tell later","languages":"Hindi","travelid":"5bdfc5759c6d2f87f4237e9f"}}
Inside editeduserdetails kafka backend
5bdfc5759c6d2f87f4237e9f
after callback
{
_id: 5bdfc5759c6d2f87f4237e9f,
firstname: 'Maulin',
lastname: 'Bodivala',
email: 'm@b.com',
password: '$2a$10$NMcvIAdfHR3SEhk5c4Ge.hd3hXOjVulon7uLSF14/FZIG14W4Gq2',
_v: 0,
bookings: [ 5bdfcce409ca266c981c050a ]
}
after handle(_id: 5bdfc5759c6d2f87f4237e9f,
firstname: 'Maulin',
lastname: 'Bodivala',
email: 'm@b.com',
password: '$2a$10$NMcvIAdfHR3SEhk5c4Ge.hd3hXOjVulon7uLSF14/FZIG14W4Gq2',
_v: 0,
bookings: [ 5bdfcce409ca266c981c050a ]
}

```

Ln 111, Col 51 Spaces: 4 UTF-8 CRLF Javascript (Babel) ESLint 9:05 PM 11/4/2018

React App hotels in san jose - Google Search

localhost:3000/traveller/home

Welcome m@b.com Help

HomeAway

Book beach houses, cabins,

Inspector React App

filter...	Commit
@@INIT	9:03:11.84
traveler_profile	+00:00:19
traveler_bookings	+00:00:09

State

Action State Diff Test

Tree Chart Raw

```

gender (pin): null
languages (pin): "Hindi"
number (pin): 2222222222
school (pin): "SJSU"
profilePic (pin): "/public/uploads/userprofile-5bdfc5759c6d2f87f4237e9f.jpg"
▶ bookings (pin): ["5bdfcce409..."]
owner (pin): { }
property (pin): { }

```

Pause recording Lock changes Persist Dispatcher Slider Import Export Remote Settings

Explanation:

- This section shows how we updated the profile picture of the travel user and edited some details.
- The profile pic is stored on the server side at the location /public/uploads/profile-{travel_id}.jpg

- After that, one screenshot shows the recent changes done on the user profile on the redux store.

Looking at the recent trips that I have booked and filtering through it

The screenshot shows a web application interface for managing travel trips. At the top, there's a navigation bar with tabs for "My Trips", "Profile", and "Account". Below the navigation bar, there are search fields for "Place Name" and "mm/dd" along with a "Clear Filter" button. Underneath these are two buttons: "Filter Place by Name" and "Filter By Date".

The main content area displays a list of "Recent Trips" in a table format:

Trips	Location, City	From	To	Guests	Base Nightly Rate was:
Timber Leaf Apartments	Location, City: San Jose	Neverly Renovated From: Tue Dec 04 2018	To: Thu Dec 06 2018	Guests: 3	Base Nightly Rate was: \$339
Courtyard by Marriott	Location, City: Campbell	Live with Luxury From: Wed Nov 14 2018	To: Thu Nov 15 2018	Guests: 3	Base Nightly Rate was: \$118

At the bottom of the table, there are navigation links for "previous", "1", "2", and "next".

The bottom of the screen shows a Windows taskbar with various pinned icons and system status indicators like battery level and network connection.

Explanation:

- I booked few other properties to demonstrate pagination and filtering options.
- The screenshot shows that I have booked 4 properties recently, and the bookings are divided into 2 pages as shown in the screenshots.(Pagination size is 2)

Lets filter through these bookings.

The screenshot shows a web browser window with the title "React App" and the URL "localhost:3000/traveller/edit/5bdfc5759c6d2f87f4237e9f". The page is titled "HomeAway" and displays "Recent Trips". There are two entries for "Timber Leaf Apartments" located in San Jose. Both entries are labeled "Newly Renovated". The first entry has a booking from Tuesday, December 4, 2018, to Thursday, December 6, 2018, for 3 guests at a base nightly rate of \$339. The second entry has a booking from Thursday, November 29, 2018, to Friday, November 30, 2018, for 4 guests at a base nightly rate of \$339. At the top of the page, there are input fields for "Timber" and "mm/dd" with a "Clear Filter" button. Below the input fields are two yellow buttons: "Filter Place by Name" and "Filter By Date". A navigation bar at the bottom includes "previous", "1", and "next" buttons. The browser's taskbar at the bottom shows various pinned icons and the system tray on the right.

Explanation

- We filtered to see only the bookings I have done for Timber Leaf Apartments. There are only 2 bookings, hence 2 results.

Logging Off

The screenshot shows the HomeAway homepage. At the top right, there is a user account dropdown menu with options: Account Information, Inbox, and Logout. Below the menu, there is a search bar with fields for 'Select a destination', two date pickers ('mm/dd/yyyy'), 'No of Guests', and a 'Search' button. The background features a photograph of a large, modern house at sunset. Three promotional banners are visible: 'Your whole vacation starts here' (Choose a rental from the world's best selection), 'Book and stay with confidence' (Secure payments, peace of mind), and 'Your vacation your way' (More space, more privacy, no compromises).

The screenshot shows the HomeAway login page. The title is 'Log in to HomeAway'. Below it, there is a link 'Need an account? [Sign Up](#)'. The main form is titled 'Travel Account login' and contains two input fields: 'Email address' and 'Password'. There is also a 'Forgot password?' link and a 'Keep me signed in' checkbox. A yellow 'Log in' button is at the bottom. The background is white.

Explanation:

- Demonstrates a simple functioning of logging off.

Answering to the question asked by the traveler

Logging into d@k owner to answer the question asked on his property

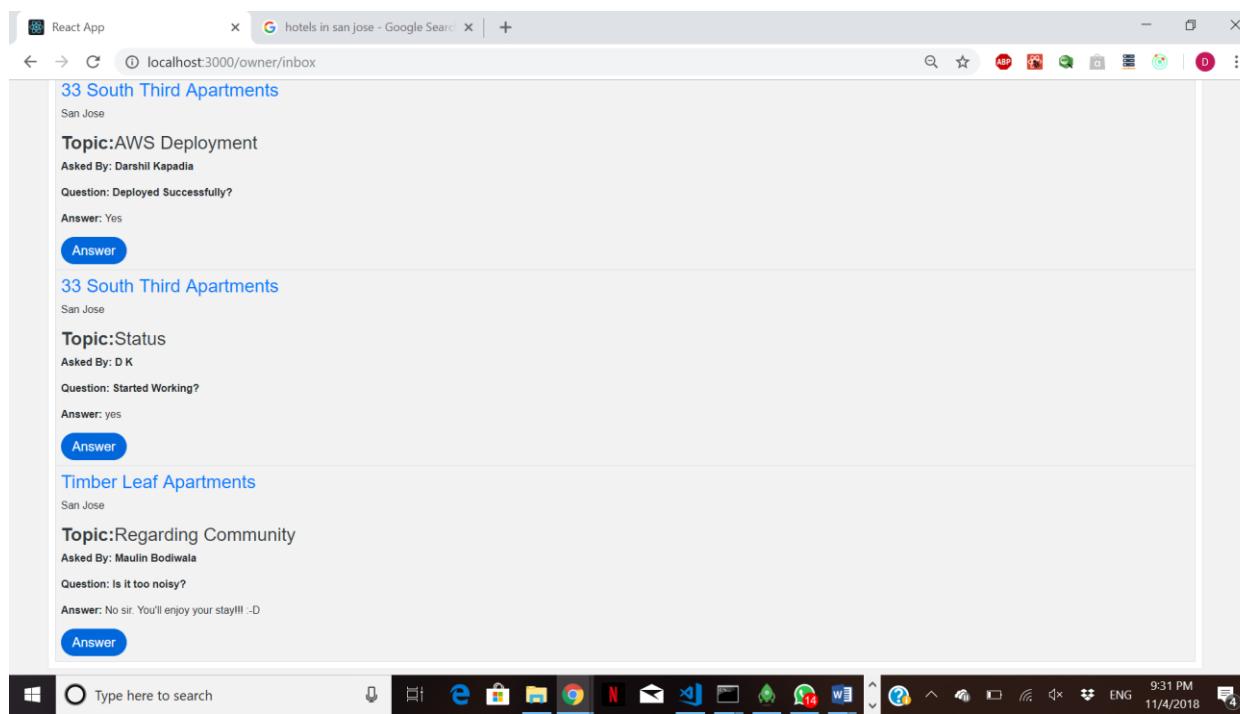
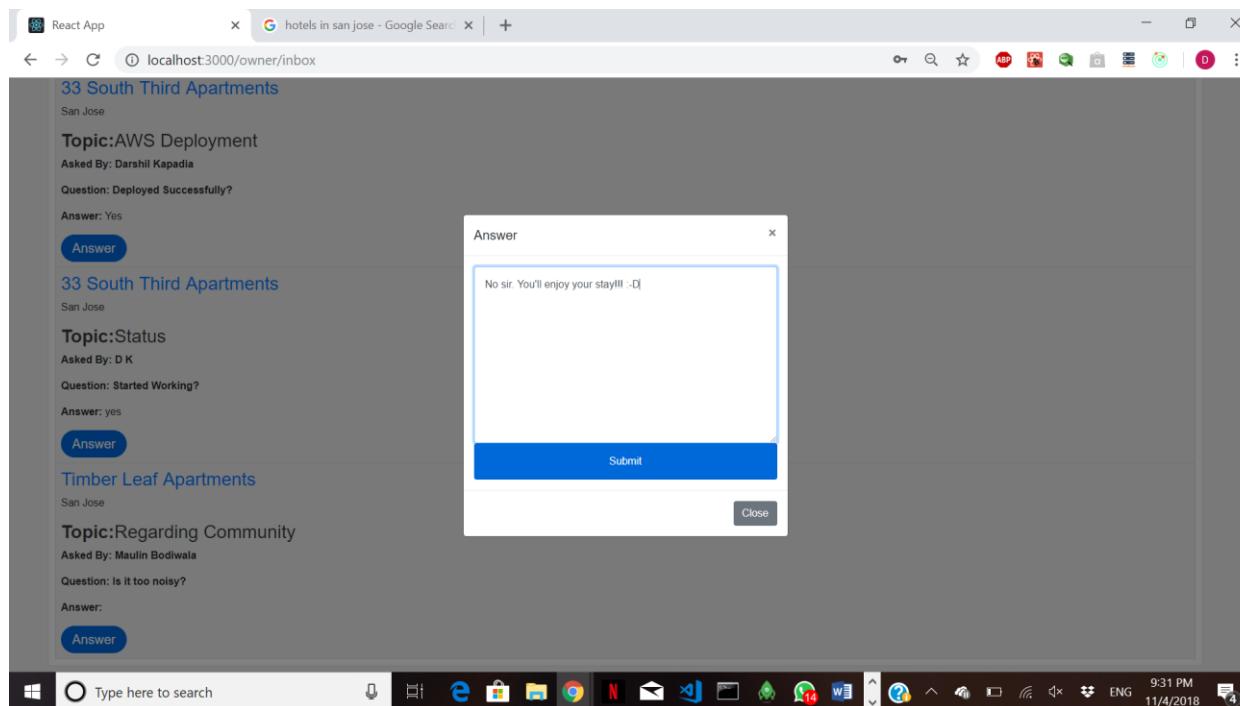
The screenshot shows a web browser window with the URL localhost:3000/owner/login. The page is titled "Owner Login" and has a "Welcome back!" message. It features a photograph of a wooden deck with lounge chairs and a fire pit at sunset. There are input fields for "Email" (containing "d@k") and "Password" (containing "..."). Below the fields are links for "Forgot password?" and "Keep me signed in". A large yellow "Log in" button is at the bottom. The browser's address bar shows "React App" and "hotels in san jose - Google Search". The taskbar at the bottom includes icons for File Explorer, Edge, Store, Photos, Google Chrome, Netflix, Mail, and others.

The screenshot shows a web browser window with the URL localhost:3000/owner/inbox. The page title is "33 South Third Apartments" and "San Jose". It lists three questions:

- Topic: AWS Deployment**
Asked By: Darshil Kapadia
Question: Deployed Successfully?
Answer: Yes
[Answer](#)
- Topic: Status**
Asked By: D K
Question: Started Working?
Answer: yes
[Answer](#)
- Topic: Regarding Community**
Asked By: Maulin Bodiwala
Question: Is it too noisy?
Answer:
[Answer](#)

The browser's address bar shows "React App" and "hotels in san jose - Google Search". The taskbar at the bottom includes icons for File Explorer, Edge, Store, Photos, Google Chrome, Netflix, Mail, and others.

3rd Question is the one asked by our new traveler. Let's answer it



The screenshot shows the Visual Studio Code interface. The top bar includes File, Edit, Selection, View, Go, Debug, Terminal, and Help. The title bar says "OwnerLoginPage.js - Lab2 - Visual Studio Code". Below the title bar, there are tabs for Details.js, OwnerHomePage.js, Inbox.js, EditProfile.js, index.js, OwnerLoginPage.js (highlighted in yellow), LoginPage.js, Main.js, and ListPlaces.js. The main editor area contains code for the OwnerLoginPage.js component. The code includes logic for handling props, logging owner info to localStorage, and pushing a history entry. The terminal below shows several API requests and responses related to user authentication and question posting.

```

105     componentWillReceiveProps(nextProps) {
106       console.log(nextProps.ownerInfo)
107       if (nextProps.ownerInfo) {
108         localStorage.setItem("ownerlogin",nextProps.ownerInfo._id)
109         localStorage.setItem("owneremail",nextProps.ownerInfo.email)
110         localStorage.setItem("token",nextProps.ownerInfo.token)
111         this.props.history.push(['owner/home'])
112     }else if(nextProps.error){

```

TERMINAL

```

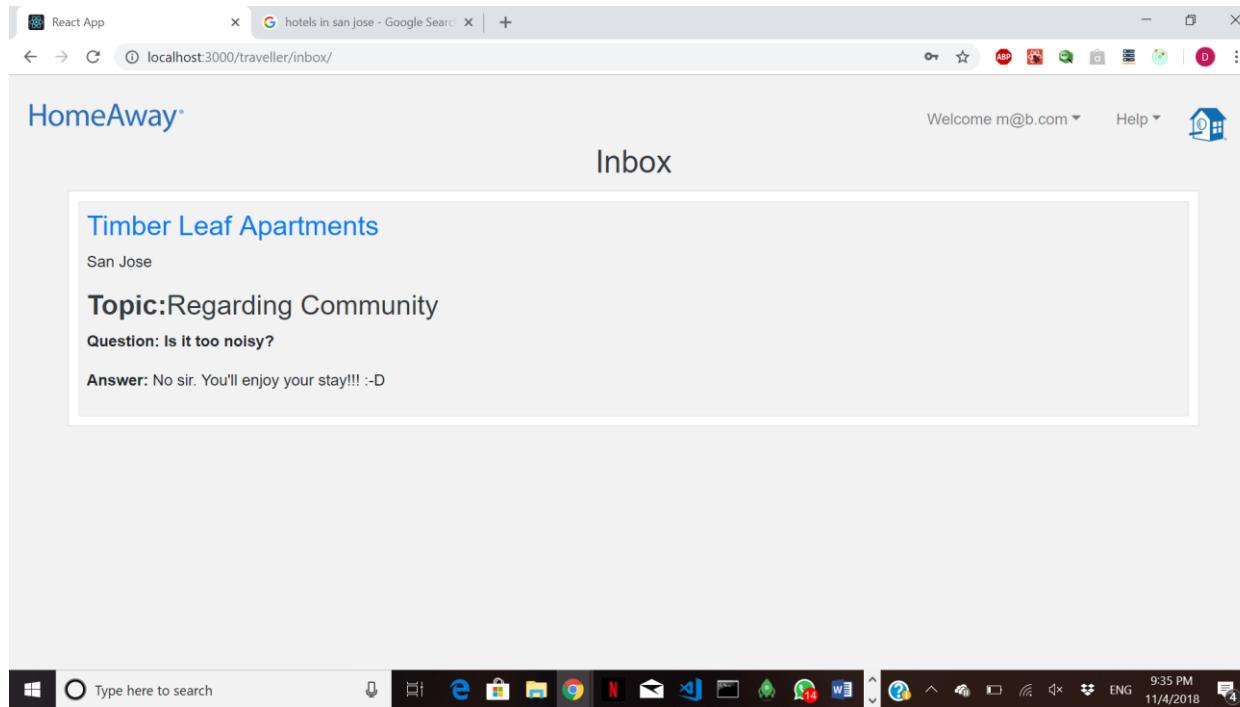
GET /owner/5bdf33bbf8d8102258c9dd39/question 200 201.909 ms -
OPTIONS /owner/5bdf33bbf8d8102258c9dd39/question 204 0.282 ms - 0
Asking a question
in make request
{ _id: '5bdf33bbf8d8102258c9dd39',
  answer: 'No sir. You\'ll enjoy your stay!!! :-D',
  ownerId: '5bdf33bbf8d8102258c9dd39' }
in response: postOwnerAnswer
in response1
true
in response2
{ postOwnerAnswer: { '0': 1 } }
msg received
POST /owner/5bdf33bbf8d8102258c9dd39/question 200 103.385 ms -
getting questions

```

Explanation:

- As shown in the screenshots, we successfully answered the traveler's question. Afterwards, we'll verify whether traveler can see the answer or not?

Checking the answer on traveler's side



Looking at Owner's Dashboard

The screenshot shows a web browser window titled "React App" with the URL "localhost:3000/owner/home". The page displays "Recent Bookings" for two properties:

- 33 South Third Apartments**
Newly Renovated
Booked By: Darshil Kapadia
From: Wed Nov 07 2018
To: Sat Nov 10 2018
Guests: 3
Base Nightly Rate: \$109
- 33 South Third Apartments**
Newly Renovated
Booked By: D K
From: Thu Nov 22 2018
To: Sat Nov 24 2018
Guests: 3
Base Nightly Rate: \$109

At the bottom of the page, there is a navigation bar with buttons for "previous", "1", "2", "3", "4", and "next". The status bar at the bottom of the screen shows "1st Page".

The screenshot shows a web browser window titled "React App" with the URL "localhost:3000/owner/home". The page displays "Recent Bookings" for two properties:

- 33 South Third Apartments**
Newly Renovated
Booked By: Darshil Kapadia
From: Sun Nov 25 2018
To: Mon Nov 26 2018
Guests: 1
Base Nightly Rate: \$109
- Timber Leaf Apartments**
Newly Renovated
Booked By: Darshil Kapadia
From: Tue Nov 20 2018
To: Thu Nov 22 2018
Guests: 3
Base Nightly Rate: \$339

At the bottom of the page, there is a navigation bar with buttons for "previous", "1", "2", "3", "4", and "next". The status bar at the bottom of the screen shows "2nd page".

Recent Bookings

Timber Leaf Apartments
Newly Renovated
Booked By: Maulin Bodiwala
From: Tue Dec 04 2018
To: Thu Dec 06 2018
Guests: 3
Base Nightly Rate: \$339

33 South Third Apartments
Newly Renovated
Booked By: Maulin Bodiwala
From: Thu Dec 13 2018
To: Sat Dec 15 2018
Guests: 1
Base Nightly Rate: \$109

previous 1 2 3 4 next

9:36 PM 11/4/2018

Recent Bookings

Timber Leaf Apartments
Newly Renovated
Booked By: Maulin Bodiwala
From: Thu Nov 29 2018
To: Fri Nov 30 2018
Guests: 4
Base Nightly Rate: \$339

previous 1 2 3 4 next

9:36 PM 11/4/2018

Filtering through recent bookings

A screenshot of a web browser window titled "React App" showing a search result for "hotels in san jose - Google Search". The URL is "localhost:3000/owner/home". The search term "33" is entered in the search bar. Below the search bar are four buttons: "Filter Place by Name", "mm/dd/yyyy", "Filter By Date", and "Clear Filter". The main content area is titled "Recent Bookings" and displays two entries for "33 South Third Apartments". Both entries are labeled "Newly Renovated". The first entry shows bookings from Darshil Kapadia (From: Wed Nov 07 2018, To: Sat Nov 10 2018, Guests: 3, Base Nightly Rate: \$109). The second entry shows bookings from D K (From: Thu Nov 22 2018, To: Sat Nov 24 2018, Guests: 3, Base Nightly Rate: \$109). A navigation bar at the bottom shows "previous" and "next" buttons, with the number "1" highlighted.

Results having only 33 South Third Apartment Bookings

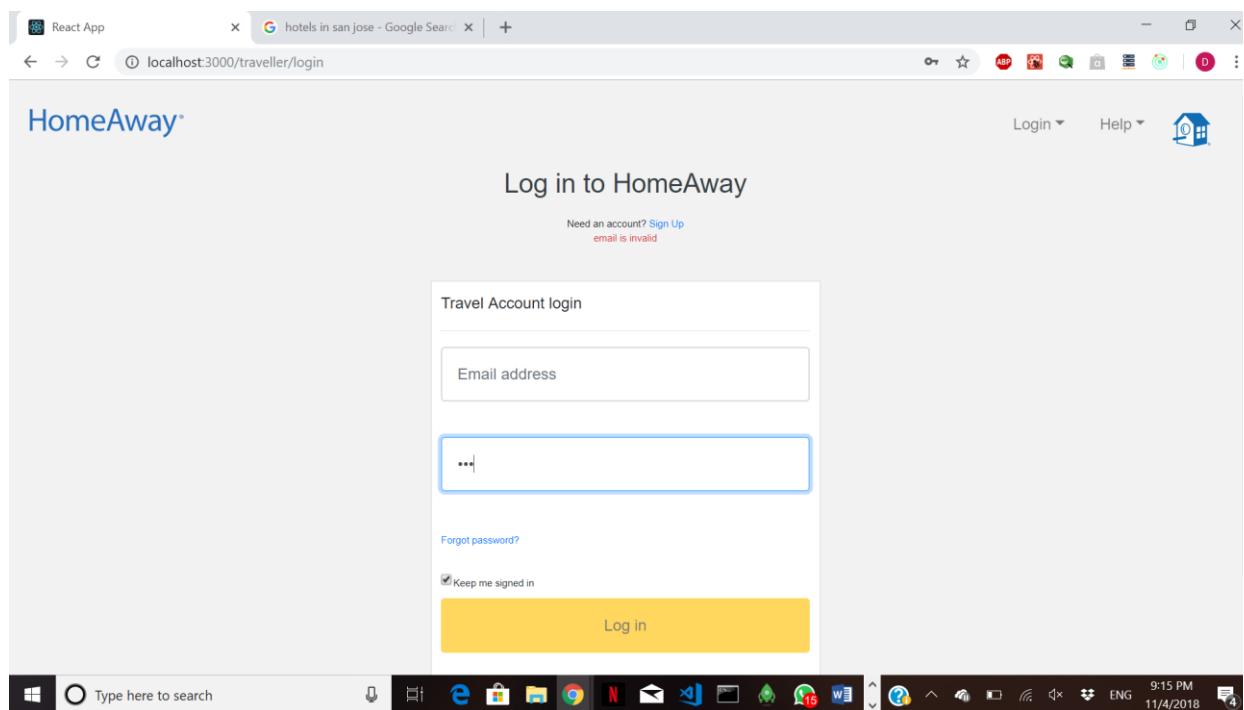
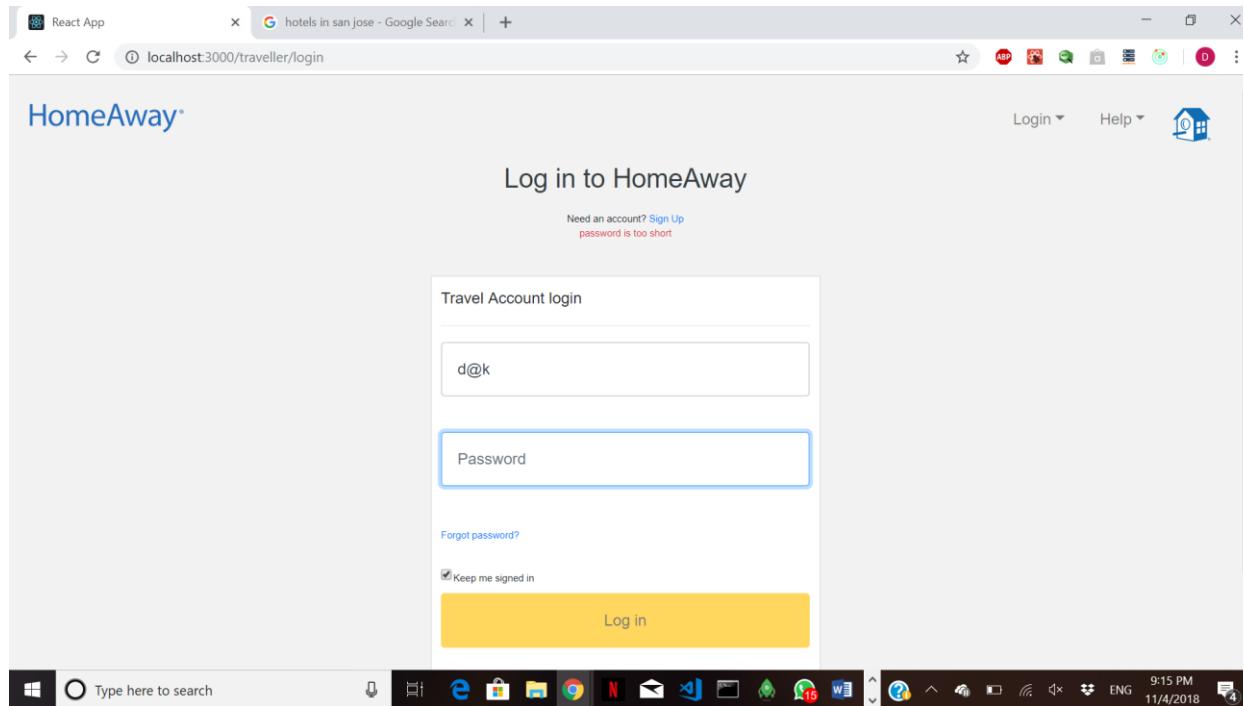
A screenshot of a web browser window titled "React App" showing a search result for "hotels in san jose - Google Search". The URL is "localhost:3000/owner/home". The search term "33" is entered in the search bar. Below the search bar are four buttons: "Filter Place by Name", "mm/dd/yyyy", "Filter By Date", and "Clear Filter". The main content area is titled "Recent Bookings" and displays two entries for "33 South Third Apartments". Both entries are labeled "Newly Renovated". The first entry shows bookings from Darshil Kapadia (From: Sun Nov 25 2018, To: Mon Nov 26 2018, Guests: 1, Base Nightly Rate: \$109). The second entry shows bookings from Maulin Bodiwala (From: Thu Dec 13 2018, To: Sat Dec 15 2018, Guests: 1, Base Nightly Rate: \$109). A navigation bar at the bottom shows "previous" and "next" buttons, with the number "1" highlighted.

Explanation:

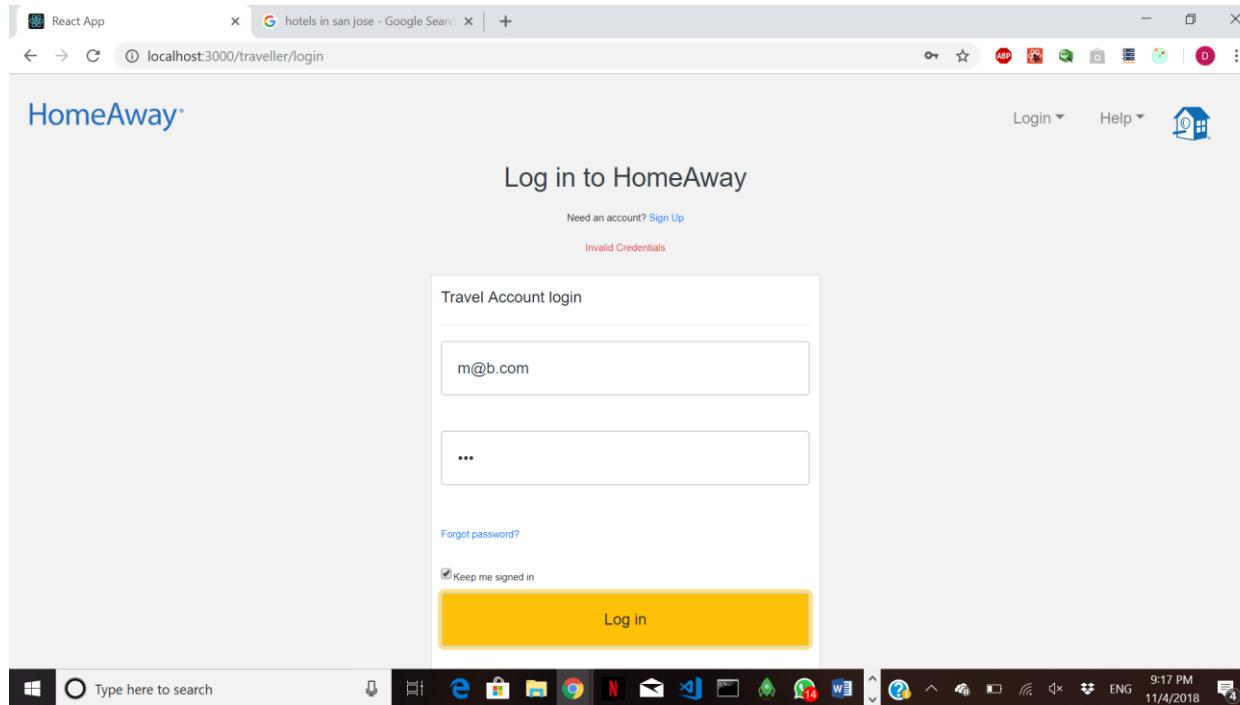
- We filtered to get only the bookings made on property named “33 South Third Apartments”

Validation Test-Cases' Results:

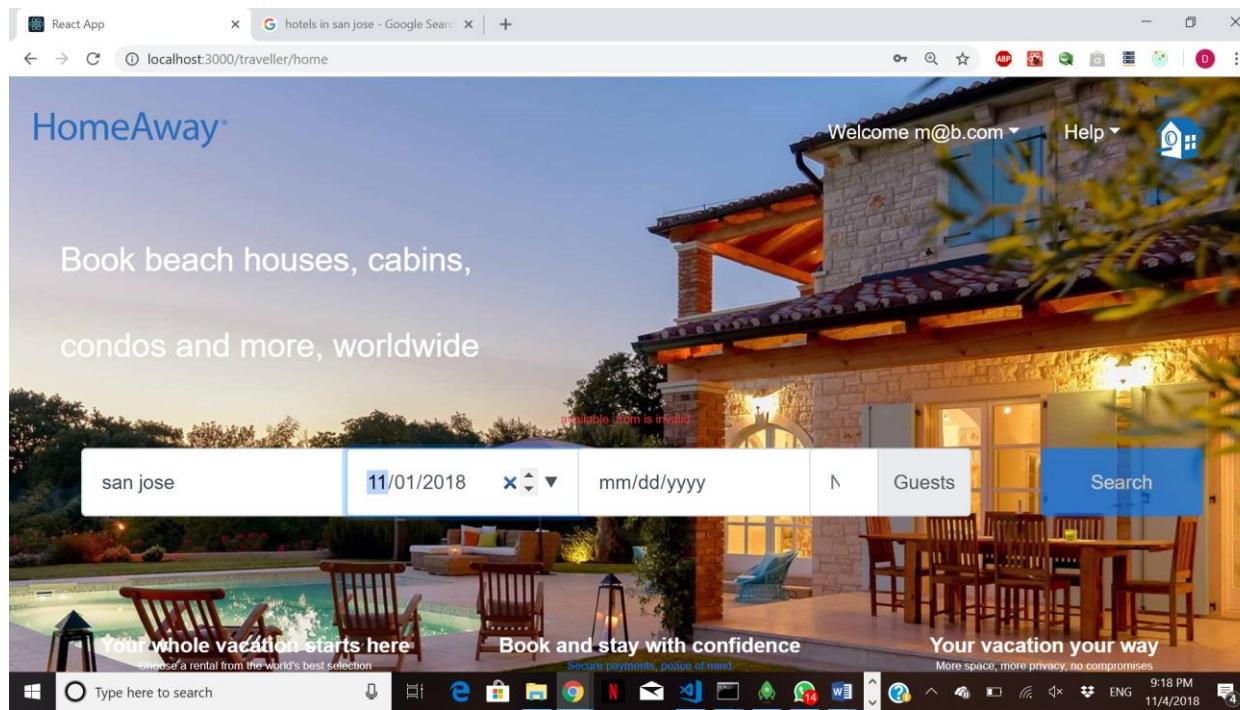
Keeping the email/password field empty

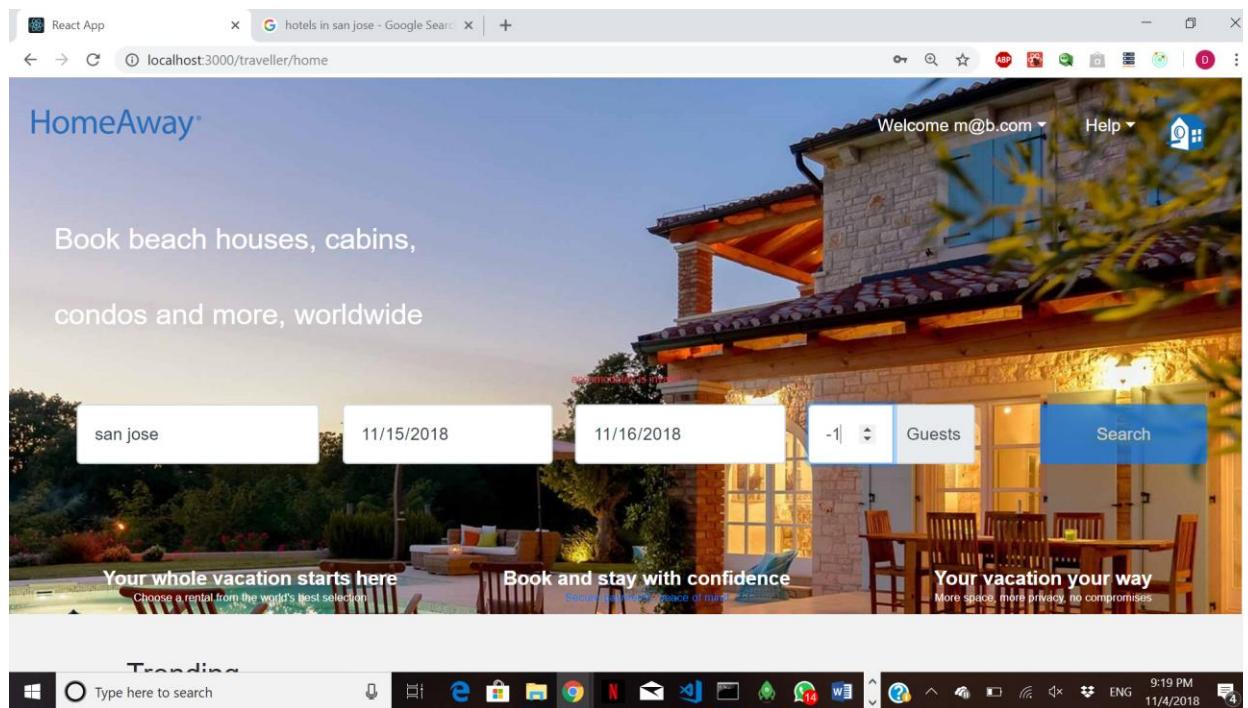


Logging in with incorrect credentials



Searching the property without location city or arrival date or departure date or guests





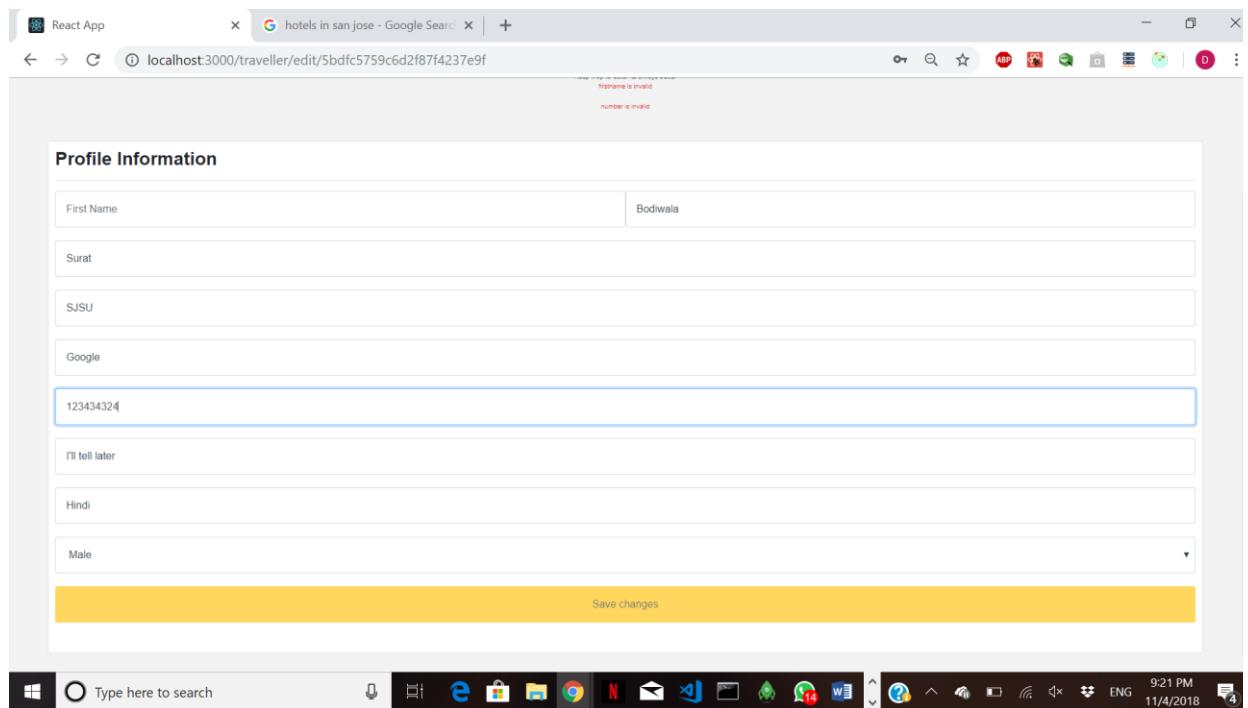
Explanation:

- First screenshot shows that the arrival date is before today's date and will show an error to the user saying "available_from is invalid".
- Second screenshot shows that the guests cannot be negative. We selected -1 as our guests which is never possible and thus it gets reflected.
- In both the cases, please notice the search button. It is disabled throughout. It will only get enabled if all the details are passing the validations.

Editing the information and keeping required fields like name blank and contact no not of 10 digits

The screenshot shows a web browser window with the URL `localhost:3000/traveller/edit/5bdfc5759c6d2f87f4237e9f`. The page is titled "Edit Profile" and displays "Profile Information". It includes fields for First Name (containing "Bodiwala" and an error message "Please fill out this field."), Middle Name (containing "Surat"), and Last Name (containing "SJSU"). The browser's status bar at the bottom shows the date and time as 9:21 PM 11/4/2018.

The screenshot shows a web browser window with the URL `localhost:3000/traveller/edit/5bdfc5759c6d2f87f4237e9f`. The page is titled "Edit Profile" and displays "Profile Information". It includes fields for First Name (containing "Bodiwala" and an error message "Please fill out this field."), Middle Name (containing "Surat"), Last Name (containing "SJSU"), and a Contact Number field (containing "123434324" and an error message "number is invalid"). The browser's status bar at the bottom shows the date and time as 9:21 PM 11/4/2018.



Explanation:

- As it shows, one cannot have empty name in the database and one cannot have more than or less than 10 digits in the contact no.
- Because these validations were proved wrong, the “Save Changes” button is disabled and the user is shown the error above the form.

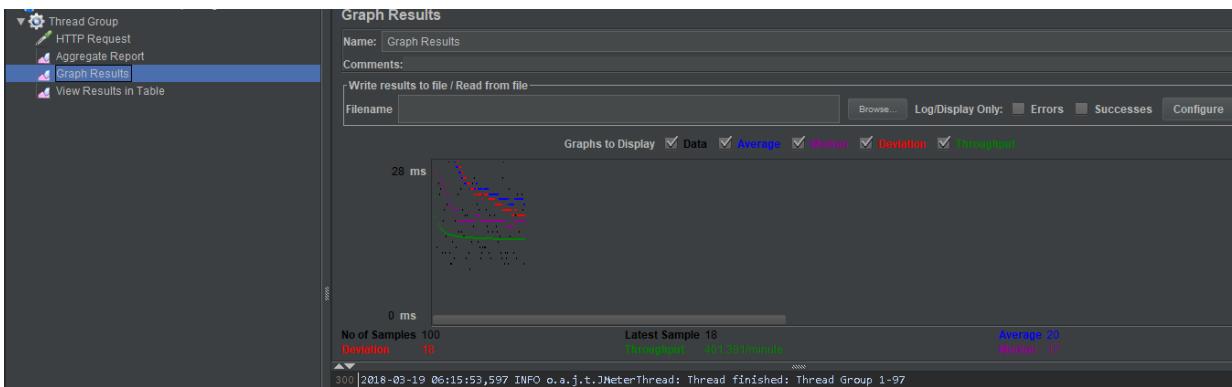
Performance

J-Meter (With Connection Pooling)

On Invoking 100 Concurrent Users

The image displays three screenshots of the JMeter application interface:

- Thread Group Configuration:** Shows a "Thread Group" node selected in the left tree. The main panel shows settings for 100 threads, a ramp-up period of 15 seconds, and a loop count of 1. It also includes scheduler and duration configurations.
- HTTP Request Configuration:** Shows an "HTTP Request" node selected. The configuration panel details a GET request to "http://localhost:3001/travel/5bde9480b05d126cd0691f7e".
- Aggregate Report Results:** Shows an "Aggregate Report" node selected. The results table provides a summary of 100 requests, with a total average response time of 20 ms and a maximum of 139 ms.



View Results in Table

Name: View Results in Table
Comments:
Write results to file / Read from file
Filename:

Sample #	Start Time	Thread Name	Label	Sample Time(...)	Status	Bytes	Sent Bytes	Latency	Connect Time(...)
1	06:15:39.110	Thread Group ...	HTTP Request	139	✓	712	133	128	23
2	06:15:39.211	Thread Group ...	HTTP Request	87	✓	714	133	87	1
3	06:15:39.383	Thread Group ...	HTTP Request	71	✓	710	133	71	1
4	06:15:39.523	Thread Group ...	HTTP Request	12	✓	710	133	12	1
5	06:15:39.680	Thread Group ...	HTTP Request	48	✓	712	133	47	3
6	06:15:39.821	Thread Group ...	HTTP Request	24	✓	716	133	24	2
7	06:15:39.978	Thread Group ...	HTTP Request	22	✓	716	133	22	2
8	06:15:40.121	Thread Group ...	HTTP Request	23	✓	716	133	23	2
9	06:15:40.272	Thread Group ...	HTTP Request	25	✓	714	133	24	2
10	06:15:40.422	Thread Group ...	HTTP Request	11	✓	714	133	11	1
11	06:15:40.571	Thread Group ...	HTTP Request	17	✓	710	133	17	1
12	06:15:40.722	Thread Group ...	HTTP Request	19	✓	710	133	18	1
13	06:15:40.871	Thread Group ...	HTTP Request	11	✓	712	133	11	1
14	06:15:41.022	Thread Group ...	HTTP Request	14	✓	714	133	14	1
15	06:15:41.172	Thread Group ...	HTTP Request	20	✓	714	133	20	2
16	06:15:41.363	Thread Group ...	HTTP Request	20	✓	714	133	20	1
17	06:15:41.471	Thread Group ...	HTTP Request	28	✓	710	133	28	1
18	06:15:41.621	Thread Group ...	HTTP Request	11	✓	716	133	11	1
19	06:15:41.772	Thread Group ...	HTTP Request	9	✓	714	133	9	0
20	06:15:41.921	Thread Group ...	HTTP Request	9	✓	710	133	9	1
21	06:15:42.072	Thread Group ...	HTTP Request	13	✓	710	133	13	1
22	06:15:42.224	Thread Group ...	HTTP Request	19	✓	716	133	19	2

300 | 2018-03-19 06:15:53,597 INFO o.a.j.t.JMeterThread: Thread finished: Thread Group 1-97

View Results in Table

Name: View Results in Table
Comments:
Write results to file / Read from file
Filename:

Sample #	Start Time	Thread Name	Label	Sample Time(...)	Status	Bytes	Sent Bytes	Latency	Connect Time(...)
76	06:15:50.379	Thread Group ...	HTTP Request	25	✓	712	133	24	2
77	06:15:50.529	Thread Group ...	HTTP Request	28	✓	712	133	28	2
78	06:15:50.679	Thread Group ...	HTTP Request	9	✓	710	133	9	1
79	06:15:50.829	Thread Group ...	HTTP Request	11	✓	712	133	11	1
80	06:15:50.979	Thread Group ...	HTTP Request	10	✓	718	133	10	1
81	06:15:51.129	Thread Group ...	HTTP Request	11	✓	714	133	11	1
82	06:15:51.279	Thread Group ...	HTTP Request	9	✓	716	133	8	0
83	06:15:51.429	Thread Group ...	HTTP Request	23	✓	712	133	22	2
84	06:15:51.579	Thread Group ...	HTTP Request	14	✓	710	133	14	2
85	06:15:51.729	Thread Group ...	HTTP Request	20	✓	710	133	20	3
86	06:15:51.879	Thread Group ...	HTTP Request	20	✓	714	133	19	2
87	06:15:52.029	Thread Group ...	HTTP Request	18	✓	710	133	18	2
88	06:15:52.185	Thread Group ...	HTTP Request	11	✓	716	133	11	2
89	06:15:52.328	Thread Group ...	HTTP Request	63	✓	714	133	63	1
90	06:15:52.527	Thread Group ...	HTTP Request	18	✓	716	133	18	2
91	06:15:52.628	Thread Group ...	HTTP Request	10	✓	710	133	10	1
92	06:15:52.779	Thread Group ...	HTTP Request	22	✓	712	133	22	2
93	06:15:52.930	Thread Group ...	HTTP Request	21	✓	714	133	21	1
94	06:15:53.127	Thread Group ...	HTTP Request	15	✓	710	133	15	1
95	06:15:53.229	Thread Group ...	HTTP Request	30	✓	710	133	30	0
96	06:15:53.427	Thread Group ...	HTTP Request	21	✓	712	133	21	2
97	06:15:53.576	Thread Group ...	HTTP Request	22	✓	720	133	22	2
98	06:15:53.710	Thread Group ...	HTTP Request	19	✓	712	133	19	2
99	06:15:53.859	Thread Group ...	HTTP Request	9	✓	714	133	9	1
100	06:15:54.040	Thread Group ...	HTTP Request	18	✓	710	133	18	2

300 | 2018-03-19 06:15:53,597 INFO o.a.j.t.JMeterThread: Thread finished: Thread Group 1-97

On invoking 200 concurrent users

Thread Group

- HTTP Request
- Aggregate Report
- Graph Results
- View Results in Table
- Thread Group**
- HTTP Request
- Aggregate Report
- View Results in Table
- Graph Results
- Thread Group
- HTTP Request
- Aggregate Report
- View Results in Table
- Graph Results
- Thread Group
- HTTP Request
- View Results in Table
- Aggregate Report

Thread Group

Name: Thread Group
Comments:
Action to be taken after a Sampler error:

- Continue
- Start Next Thread Loop
- Stop Thread
- Stop Test
- Stop Test Now

Thread Properties

Number of Threads (users): 200
Ramp-Up Period (in seconds): 16
Loop Count: Forever 1
 Delay Thread creation until needed
 Scheduler

Scheduler Configuration

Duration (seconds):

Aggregate Report

- HTTP Request
- Aggregate Report
- Graph Results
- View Results in Table
- Thread Group
- HTTP Request
- Aggregate Report
- View Results in Table
- Graph Results
- Thread Group
- HTTP Request
- Aggregate Report
- View Results in Table
- Graph Results
- Thread Group
- HTTP Request
- View Results in Table
- Aggregate Report

Aggregate Report

Name: Aggregate Report
Comments:
Write results to file / Read from file

Label	# Samples	Average	Median	90% Line	95% Line	99% Line	Min	Max	Error %	Throughput	Received K...	Sent KB/sec
HTTP Requ...	200	13	10	21	30	118	7	130	0.00%	12.2/sec	8.51	1.5
TOTAL	200	13	10	21	30	118	7	130	0.00%	12.2/sec	8.51	1.5

Aggregate Report

- HTTP Request
- Aggregate Report
- Graph Results
- View Results in Table
- Thread Group
- HTTP Request
- Aggregate Report
- View Results in Table
- Graph Results
- Thread Group
- HTTP Request
- Aggregate Report
- View Results in Table
- Graph Results
- Thread Group
- HTTP Request
- View Results in Table
- Aggregate Report

Aggregate Report

Name: Aggregate Report
Comments:
Write results to file / Read from file

Label	# Samples	Average	Median	90% Line	95% Line	99% Line	Min	Max	Error %	Throughput	Received K...	Sent KB/sec
HTTP Requ...	200	13	10	21	30	118	7	130	0.00%	12.2/sec	8.51	1.5
TOTAL	200	13	10	21	30	118	7	130	0.00%	12.2/sec	8.51	1.5

View Results in Table

- HTTP Request
- Aggregate Report
- Graph Results
- View Results in Table
- Thread Group
- HTTP Request
- Aggregate Report
- View Results in Table
- Graph Results
- Thread Group
- HTTP Request
- Aggregate Report
- View Results in Table
- Graph Results
- Thread Group
- HTTP Request
- View Results in Table
- Aggregate Report

View Results in Table

Name: View Results in Table
Comments:
Write results to file / Read from file

Sample #	Start Time	Thread Name	Label	Sample Time...	Status	Bytes	Sent Bytes	Latency	Connect Time...
1	07:07:57.678	Thread Group ...	HTTP Request	130	✓	712	133	129	105
2	07:07:57.758	Thread Group ...	HTTP Request	57	✓	712	133	57	27
3	07:07:57.849	Thread Group ...	HTTP Request	118	✓	712	133	118	3
4	07:07:57.851	Thread Group ...	HTTP Request	126	✓	710	133	126	2
5	07:07:57.941	Thread Group ...	HTTP Request	45	✓	710	133	44	3
6	07:07:58.004	Thread Group ...	HTTP Request	22	✓	716	133	22	2
7	07:07:58.073	Thread Group ...	HTTP Request	19	✓	716	133	19	2
8	07:07:58.156	Thread Group ...	HTTP Request	16	✓	712	133	16	2
9	07:07:58.240	Thread Group ...	HTTP Request	13	✓	716	133	12	1
10	07:07:58.322	Thread Group ...	HTTP Request	8	✓	710	133	8	1
11	07:07:58.407	Thread Group ...	HTTP Request	20	✓	710	133	20	3
12	07:07:58.511	Thread Group ...	HTTP Request	21	✓	710	133	21	2
13	07:07:58.650	Thread Group ...	HTTP Request	10	✓	716	133	10	1
14	07:07:58.796	Thread Group ...	HTTP Request	19	✓	710	133	19	3
15	07:07:58.811	Thread Group ...	HTTP Request	24	✓	714	133	24	2
16	07:07:58.897	Thread Group ...	HTTP Request	18	✓	712	133	18	2
17	07:07:58.973	Thread Group ...	HTTP Request	26	✓	712	133	26	3
18	07:07:58.958	Thread Group ...	HTTP Request	56	✓	712	133	56	2
19	07:07:59.037	Thread Group ...	HTTP Request	26	✓	712	133	26	2
20	07:07:59.146	Thread Group ...	HTTP Request	9	✓	710	133	9	2
21	07:07:59.279	Thread Group ...	HTTP Request	9	✓	712	133	8	1
22	07:07:59.291	Thread Group ...	HTTP Request	8	✓	716	133	8	1

Test Plan Structure:

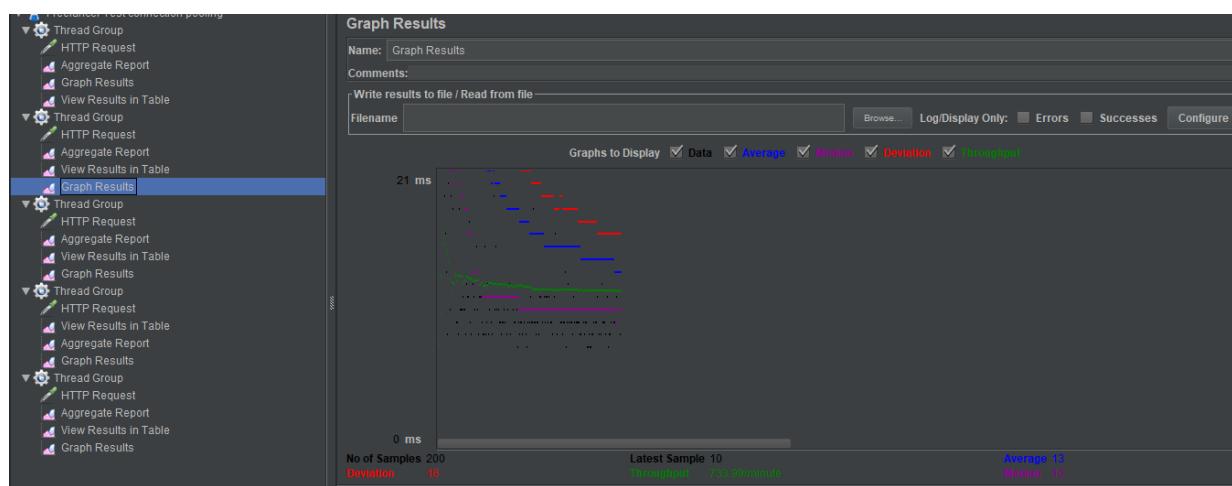
- Thread Group 1
 - HTTP Request
 - Aggregate Report
 - Graph Results
 - View Results in Table**
- Thread Group 2
 - HTTP Request
 - Aggregate Report
 - View Results in Table
- Thread Group 3
 - HTTP Request
 - Aggregate Report
 - Graph Results
- Thread Group 4
 - HTTP Request
 - Aggregate Report
 - View Results in Table
- Thread Group 5
 - HTTP Request
 - Aggregate Report
 - View Results in Table

View Results in Table Data:

Sample #	Start Time	Thread Name	Label	Sample Time(...)	Status	Bytes	Sent Bytes	Latency	Connect Time(...)
176	07:08:11.985	Thread Group ...	HTTP Request	8	✓	710	133	8	1
177	07:08:12.056	Thread Group ...	HTTP Request	30	✓	710	133	30	23
178	07:08:12.146	Thread Group ...	HTTP Request	8	✓	710	133	8	1
179	07:08:12.227	Thread Group ...	HTTP Request	12	✓	710	133	12	1
180	07:08:12.307	Thread Group ...	HTTP Request	9	✓	712	133	9	1
181	07:08:12.387	Thread Group ...	HTTP Request	9	✓	712	133	9	1
182	07:08:12.466	Thread Group ...	HTTP Request	9	✓	712	133	9	1
183	07:08:12.601	Thread Group ...	HTTP Request	8	✓	714	133	8	1
184	07:08:12.703	Thread Group ...	HTTP Request	24	✓	712	133	24	17
185	07:08:12.762	Thread Group ...	HTTP Request	11	✓	710	133	11	1
186	07:08:12.843	Thread Group ...	HTTP Request	7	✓	712	133	7	0
187	07:08:12.953	Thread Group ...	HTTP Request	8	✓	714	133	8	1
188	07:08:13.004	Thread Group ...	HTTP Request	8	✓	712	133	8	1
189	07:08:13.094	Thread Group ...	HTTP Request	9	✓	716	133	9	1
190	07:08:13.169	Thread Group ...	HTTP Request	10	✓	714	133	10	1
191	07:08:13.245	Thread Group ...	HTTP Request	10	✓	714	133	10	1
192	07:08:13.325	Thread Group ...	HTTP Request	9	✓	712	133	9	1
193	07:08:13.406	Thread Group ...	HTTP Request	9	✓	710	133	9	1
194	07:08:13.487	Thread Group ...	HTTP Request	9	✓	712	133	9	1
195	07:08:13.567	Thread Group ...	HTTP Request	11	✓	712	133	11	1
196	07:08:13.676	Thread Group ...	HTTP Request	9	✓	714	133	9	1
197	07:08:13.728	Thread Group ...	HTTP Request	10	✓	714	133	10	1
198	07:08:13.809	Thread Group ...	HTTP Request	29	✓	710	133	29	1
199	07:08:13.888	Thread Group ...	HTTP Request	8	✓	712	133	8	1
200	07:08:14.017	Thread Group ...	HTTP Request	10	✓	712	133	10	1

Log/Display Only: Errors Successes Configure

Scroll automatically: Child samples: No of Samples: 200 Latest Sample: 10 Average: 13 Deviation: 16



On Invoking 300 Concurrent Users

Test Plan Structure:

- Thread Group 1
 - HTTP Request
 - Aggregate Report
 - Graph Results
 - View Results in Table**
- Thread Group 2
 - HTTP Request
 - Aggregate Report
 - View Results in Table
- Thread Group 3
 - HTTP Request
 - Aggregate Report
 - Graph Results
- Thread Group 4
 - HTTP Request
 - Aggregate Report
 - View Results in Table
- Thread Group 5
 - HTTP Request
 - Aggregate Report
 - View Results in Table

Thread Group Configuration:

Name: Thread Group

Comments:

Action to be taken after a Sampler error:

Continue Start Next Thread Loop Stop Thread Stop Test Stop Test Now

Thread Properties:

Number of Threads (users): 300

Ramp-Up Period (in seconds): 19

Loop Count: Forever 1

Delay Thread creation until needed

Scheduler

Scheduler Configuration:

Duration (seconds):

Startup delay (seconds):

Aggregate Report

Name: Aggregate Report
Comments:
Write results to file / Read from file
Filename Browse... Log/Display Only: Errors Successes Configure

Label	# Samples	Average	Median	90% Line	95% Line	99% Line	Min	Max	Error %	Throughput	Received KB/sec	Sent KB/sec
HTTP Request	300	9	9	11	15	17	5	37	0.00%	15.8/sec	10.98	2.0
TOTAL	300	9	9	11	15	17	5	37	0.00%	15.8/sec	10.98	2.0

View Results In Table

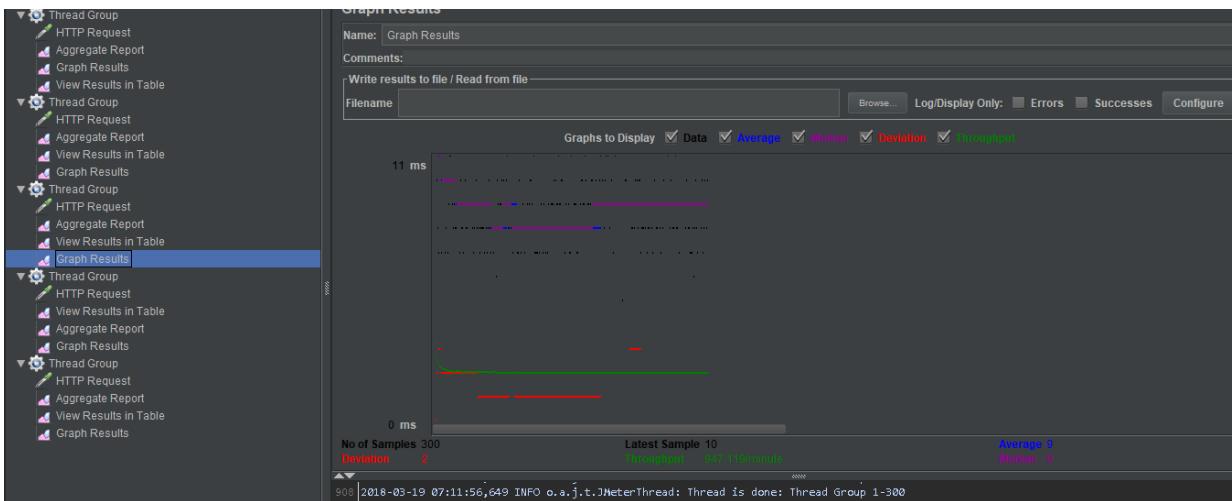
Name: View Results in Table
Comments:
Write results to file / Read from file
Filename Browse... Log/Display Only: Errors Successes Configure

Sample #	Start Time	Thread Name	Label	Sample Time(...)	Status	Bytes	Sent Bytes	Latency	Connect Time(...)
1	07:11:37.645	Thread Group ...	HTTP Request	15	✓	714	133	15	1
2	07:11:37.700	Thread Group ...	HTTP Request	14	✓	710	133	14	1
3	07:11:37.772	Thread Group ...	HTTP Request	15	✓	714	133	15	1
4	07:11:37.836	Thread Group ...	HTTP Request	10	✓	714	133	10	2
5	07:11:37.901	Thread Group ...	HTTP Request	8	✓	710	133	8	1
6	07:11:37.954	Thread Group ...	HTTP Request	7	✓	710	133	7	1
7	07:11:38.028	Thread Group ...	HTTP Request	11	✓	712	133	11	1
8	07:11:38.092	Thread Group ...	HTTP Request	11	✓	712	133	11	1
9	07:11:38.152	Thread Group ...	HTTP Request	7	✓	714	133	7	1
10	07:11:38.221	Thread Group ...	HTTP Request	10	✓	712	133	10	1
11	07:11:38.284	Thread Group ...	HTTP Request	10	✓	712	133	10	1
12	07:11:38.348	Thread Group ...	HTTP Request	10	✓	712	133	10	1
13	07:11:38.411	Thread Group ...	HTTP Request	7	✓	712	133	7	1
14	07:11:38.475	Thread Group ...	HTTP Request	8	✓	718	133	8	1
15	07:11:38.539	Thread Group ...	HTTP Request	15	✓	718	133	15	1
16	07:11:38.602	Thread Group ...	HTTP Request	7	✓	712	133	7	1
17	07:11:38.667	Thread Group ...	HTTP Request	9	✓	716	133	9	1
18	07:11:38.730	Thread Group ...	HTTP Request	11	✓	710	133	11	1
19	07:11:38.795	Thread Group ...	HTTP Request	11	✓	712	133	11	1
20	07:11:38.859	Thread Group ...	HTTP Request	10	✓	716	133	10	1
21	07:11:38.945	Thread Group ...	HTTP Request	9	✓	712	133	9	1
22	07:11:38.990	Thread Group ...	HTTP Request	8	✓	712	133	8	1

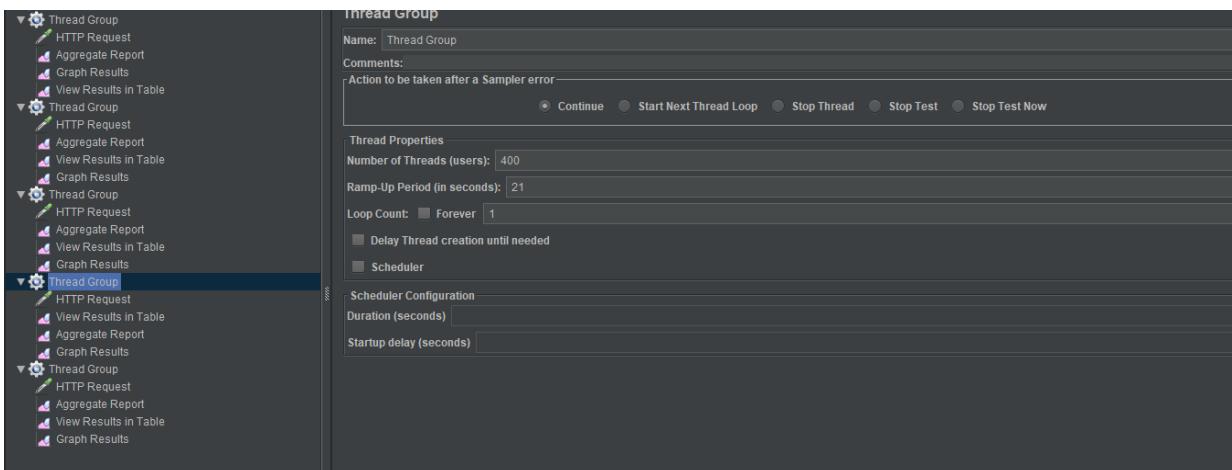
View Results In Table

Filename Browse... Log/Display Only: Errors Successes Configure

Sample #	Start Time	Thread Name	Label	Sample Time(...)	Status	Bytes	Sent Bytes	Latency	Connect Time(...)
276	07:11:55.120	Thread Group ...	HTTP Request	8	✓	714	133	8	1
277	07:11:55.183	Thread Group ...	HTTP Request	10	✓	714	133	10	1
278	07:11:55.246	Thread Group ...	HTTP Request	9	✓	712	133	9	1
279	07:11:55.309	Thread Group ...	HTTP Request	8	✓	714	133	8	1
280	07:11:55.374	Thread Group ...	HTTP Request	8	✓	712	133	8	1
281	07:11:55.437	Thread Group ...	HTTP Request	7	✓	712	133	7	1
282	07:11:55.501	Thread Group ...	HTTP Request	7	✓	716	133	7	1
283	07:11:55.565	Thread Group ...	HTTP Request	7	✓	710	133	7	1
284	07:11:55.627	Thread Group ...	HTTP Request	8	✓	712	133	8	1
285	07:11:55.690	Thread Group ...	HTTP Request	6	✓	712	133	6	1
286	07:11:55.754	Thread Group ...	HTTP Request	8	✓	712	133	8	1
287	07:11:55.817	Thread Group ...	HTTP Request	10	✓	712	133	10	1
288	07:11:55.879	Thread Group ...	HTTP Request	8	✓	710	133	8	1
289	07:11:55.944	Thread Group ...	HTTP Request	7	✓	714	133	7	1
290	07:11:56.008	Thread Group ...	HTTP Request	9	✓	710	133	9	1
291	07:11:56.071	Thread Group ...	HTTP Request	9	✓	714	133	9	1
292	07:11:56.135	Thread Group ...	HTTP Request	9	✓	710	133	9	1
293	07:11:56.198	Thread Group ...	HTTP Request	8	✓	712	133	8	1
294	07:11:56.260	Thread Group ...	HTTP Request	10	✓	710	133	10	1
295	07:11:56.325	Thread Group ...	HTTP Request	7	✓	714	133	7	1
296	07:11:56.388	Thread Group ...	HTTP Request	8	✓	710	133	8	1
297	07:11:56.451	Thread Group ...	HTTP Request	10	✓	714	133	10	1
298	07:11:56.515	Thread Group ...	HTTP Request	9	✓	712	133	9	2
299	07:11:56.578	Thread Group ...	HTTP Request	8	✓	718	133	8	2
300	07:11:56.640	Thread Group ...	HTTP Request	10	✓	710	133	10	2



On invoking 400 concurrent users



Sample #	Start Time	Thread Name	Label	Sample Time(μs)	Status	Bytes	Sent Bytes	Latency	Connect Time(μs)
1	07:15:48.832	Thread Group ...	HTTP Request	13	✓	718	133	13	1
2	07:15:48.884	Thread Group ...	HTTP Request	9	✓	714	133	9	1
3	07:15:49.937	Thread Group ...	HTTP Request	9	✓	710	133	9	1
4	07:15:49.946	Thread Group ...	HTTP Request	9	✓	710	133	9	1
5	07:15:49.046	Thread Group ...	HTTP Request	10	✓	718	133	10	2
6	07:15:49.100	Thread Group ...	HTTP Request	9	✓	714	133	9	1
7	07:15:49.153	Thread Group ...	HTTP Request	9	✓	718	133	9	1
8	07:15:49.206	Thread Group ...	HTTP Request	9	✓	710	133	9	1
9	07:15:49.258	Thread Group ...	HTTP Request	0	✓	710	133	9	1
10	07:15:49.310	Thread Group ...	HTTP Request	12	✓	712	133	12	1
11	07:15:49.358	Thread Group ...	HTTP Request	9	✓	710	133	9	1
12	07:15:49.410	Thread Group ...	HTTP Request	8	✓	712	133	8	2
13	07:15:49.465	Thread Group ...	HTTP Request	9	✓	710	133	9	1
14	07:15:49.517	Thread Group ...	HTTP Request	8	✓	712	133	8	1
15	07:15:49.574	Thread Group ...	HTTP Request	10	✓	716	133	10	1
16	07:15:49.626	Thread Group ...	HTTP Request	9	✓	710	133	9	1
17	07:15:49.680	Thread Group ...	HTTP Request	9	✓	714	133	9	1
18	07:15:49.747	Thread Group ...	HTTP Request	8	✓	712	133	8	1
19	07:15:49.786	Thread Group ...	HTTP Request	9	✓	718	133	9	1
20	07:15:49.839	Thread Group ...	HTTP Request	9	✓	714	133	9	1
21	07:15:49.895	Thread Group ...	HTTP Request	8	✓	712	133	8	1
22	07:15:49.948	Thread Group ...	HTTP Request	8	✓	714	133	8	1

Tree View:

- Thread Group
 - HTTP Request
 - Aggregate Report
 - Graph Results
 - View Results in Table
- Thread Group
 - HTTP Request
 - Aggregate Report
 - View Results in Table
 - Graph Results
- Thread Group
 - HTTP Request
 - Aggregate Report
 - View Results in Table
 - Graph Results
- Thread Group
 - HTTP Request
 - Aggregate Report
 - View Results in Table
 - Graph Results
- Thread Group
 - HTTP Request
 - Aggregate Report
 - View Results in Table
 - Graph Results
- Thread Group
 - HTTP Request
 - Aggregate Report
 - View Results in Table
 - Graph Results

Table View (Sample # 376 to 400):

Sample #	Start Time	Thread Name	Label	Sample Time(...)	Status	Bytes	Sent Bytes	Latency	Connect Time(...)
376	07:16:08.567	Thread Group ...	HTTP Request	11	✓	710	133	11	1
377	07:16:08.640	Thread Group ...	HTTP Request	10	✓	714	133	10	1
378	07:16:08.692	Thread Group ...	HTTP Request	11	✓	712	133	11	1
379	07:16:08.745	Thread Group ...	HTTP Request	10	✓	714	133	10	1
380	07:16:08.797	Thread Group ...	HTTP Request	10	✓	716	133	10	1
381	07:16:08.850	Thread Group ...	HTTP Request	9	✓	712	133	9	1
382	07:16:08.902	Thread Group ...	HTTP Request	9	✓	716	133	9	1
383	07:16:08.955	Thread Group ...	HTTP Request	9	✓	712	133	9	1
384	07:16:09.007	Thread Group ...	HTTP Request	9	✓	716	133	9	1
385	07:16:09.059	Thread Group ...	HTTP Request	10	✓	712	133	10	1
386	07:16:09.112	Thread Group ...	HTTP Request	11	✓	712	133	11	1
387	07:16:09.185	Thread Group ...	HTTP Request	11	✓	710	133	11	1
388	07:16:09.217	Thread Group ...	HTTP Request	11	✓	712	133	11	2
389	07:16:09.271	Thread Group ...	HTTP Request	11	✓	712	133	11	2
390	07:16:09.322	Thread Group ...	HTTP Request	10	✓	712	133	10	1
391	07:16:09.375	Thread Group ...	HTTP Request	8	✓	712	133	8	1
392	07:16:09.428	Thread Group ...	HTTP Request	10	✓	712	133	10	1
393	07:16:09.482	Thread Group ...	HTTP Request	10	✗	714	133	10	1
394	07:16:09.533	Thread Group ...	HTTP Request	8	✓	712	133	8	1
395	07:16:09.586	Thread Group ...	HTTP Request	9	✓	712	133	9	1
396	07:16:09.638	Thread Group ...	HTTP Request	8	✓	712	133	8	1
397	07:16:09.692	Thread Group ...	HTTP Request	8	✓	712	133	8	0
398	07:16:09.744	Thread Group ...	HTTP Request	9	✓	716	133	9	1
399	07:16:09.796	Thread Group ...	HTTP Request	12	✓	714	133	12	1
400	07:16:09.849	Thread Group ...	HTTP Request	11	✓	712	133	11	1

Log View:

```
996|2018-03-19 07:16:09,848 INFO o.a.j.t.JMeterThread: Thread started: Thread Group 1-400
997|2018-03-19 07:16:09,859 INFO o.a.j.t.JMeterThread: Thread is done: Thread Group 1-400
```

Tree View:

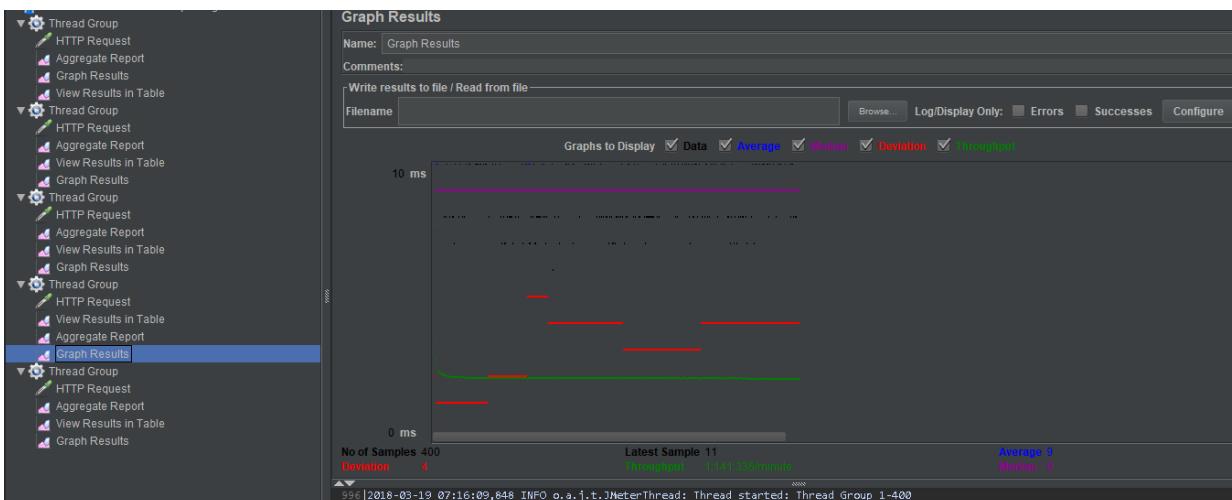
- Thread Group
 - HTTP Request
 - Aggregate Report
 - Graph Results
 - View Results in Table
- Thread Group
 - HTTP Request
 - Aggregate Report
 - View Results in Table
 - Graph Results
- Thread Group
 - HTTP Request
 - Aggregate Report
 - View Results in Table
 - Graph Results
- Thread Group
 - HTTP Request
 - Aggregate Report
 - View Results in Table
 - Graph Results
- Thread Group
 - HTTP Request
 - Aggregate Report
 - View Results in Table
 - Graph Results

Aggregate Report View:

Label	# Samples	Average	Median	90% Line	95% Line	99% Line	Min	Max	Error %	Throughput	Received K	Sent KB/s
HTTP Request	400	9	9	11	12	19	6	66	0.00%	19.0/sec	13.24	2.4
TOTAL	400	9	9	11	12	19	6	66	0.00%	19.0/sec	13.24	2.4

Save Options:

- Include group name in label?
- Save Table Data
- Save Table Header



On invoking 500 concurrent user

Thread Group

- HTTP Request
- Aggregate Report
- Graph Results
- View Results in Table
- Thread Group
- HTTP Request
- Aggregate Report
- Graph Results
- Thread Group
- HTTP Request
- Aggregate Report
- Graph Results
- Thread Group
- HTTP Request
- Aggregate Report
- Graph Results
- Thread Group
- HTTP Request
- Aggregate Report
- Graph Results
- Thread Group
- HTTP Request
- Aggregate Report
- Graph Results

Thread Group

Name: Thread Group
Comments:
Action to be taken after a Sampler error:

- Continue
- Start Next Thread Loop
- Stop Thread
- Stop Test
- Stop Test Now

Thread Properties

Number of Threads (users): 500
Ramp-Up Period (in seconds): 24
Loop Count: Forever 1
 Delay Thread creation until needed
 Scheduler

Scheduler Configuration

Duration (seconds):
Startup delay (seconds):

Aggregate Report

Name: Aggregate Report
Comments:
Write results to file / Read from file

Label	# Samples	Average	Median	90% Line	95% Line	99% Line	Min	Max	Error %	Throughput	Received K...	Sent KB/sec
HTTP Requ...	500	11	10	17	20	28	7	57	0.00%	20 B/sec	14.47	2.7
TOTAL	500	11	10	17	20	28	7	57	0.00%	20 B/sec	14.47	2.7

View Results In Table

Name: View Results In Table
Comments:
Write results to file / Read from file

Sample #	Start Time	Thread Name	Label	Sample Time(...)	Status	Bytes	Sent Bytes	Latency	Connect Time(...)
1	07:19:37.824	Thread Group ...	HTTP Request	17	✓	712	133	17	3
2	07:19:37.872	Thread Group ...	HTTP Request	11	✓	716	133	11	1
3	07:19:37.922	Thread Group ...	HTTP Request	10	✓	710	133	10	1
4	07:19:37.970	Thread Group ...	HTTP Request	9	✓	710	133	9	1
5	07:19:38.019	Thread Group ...	HTTP Request	10	✓	710	133	10	1
6	07:19:38.065	Thread Group ...	HTTP Request	9	✓	712	133	9	1
7	07:19:38.116	Thread Group ...	HTTP Request	9	✓	712	133	9	1
8	07:19:38.163	Thread Group ...	HTTP Request	11	✓	714	133	11	1
9	07:19:38.216	Thread Group ...	HTTP Request	9	✓	718	133	9	1
10	07:19:38.264	Thread Group ...	HTTP Request	8	✓	714	133	8	1
11	07:19:38.306	Thread Group ...	HTTP Request	10	✓	712	133	10	1
12	07:19:38.355	Thread Group ...	HTTP Request	10	✓	712	133	10	1
13	07:19:38.410	Thread Group ...	HTTP Request	9	✓	710	133	9	1
14	07:19:38.457	Thread Group ...	HTTP Request	9	✓	712	133	9	2
15	07:19:38.506	Thread Group ...	HTTP Request	8	✓	710	133	8	2
16	07:19:38.555	Thread Group ...	HTTP Request	8	✓	716	133	8	1
17	07:19:38.604	Thread Group ...	HTTP Request	9	✓	714	133	9	1
18	07:19:38.677	Thread Group ...	HTTP Request	9	✓	718	133	9	1
19	07:19:38.700	Thread Group ...	HTTP Request	10	✓	712	133	10	1
20	07:19:38.750	Thread Group ...	HTTP Request	9	✓	710	133	9	1
21	07:19:38.808	Thread Group ...	HTTP Request	12	✓	712	133	12	1
22	07:19:38.866	Thread Group ...	HTTP Request	8	✓	710	133	8	1

998 [2018-03-19 07:20:01,864 INFO o.a.j.t.JMeterThread: Thread started: Thread Group 1-500]

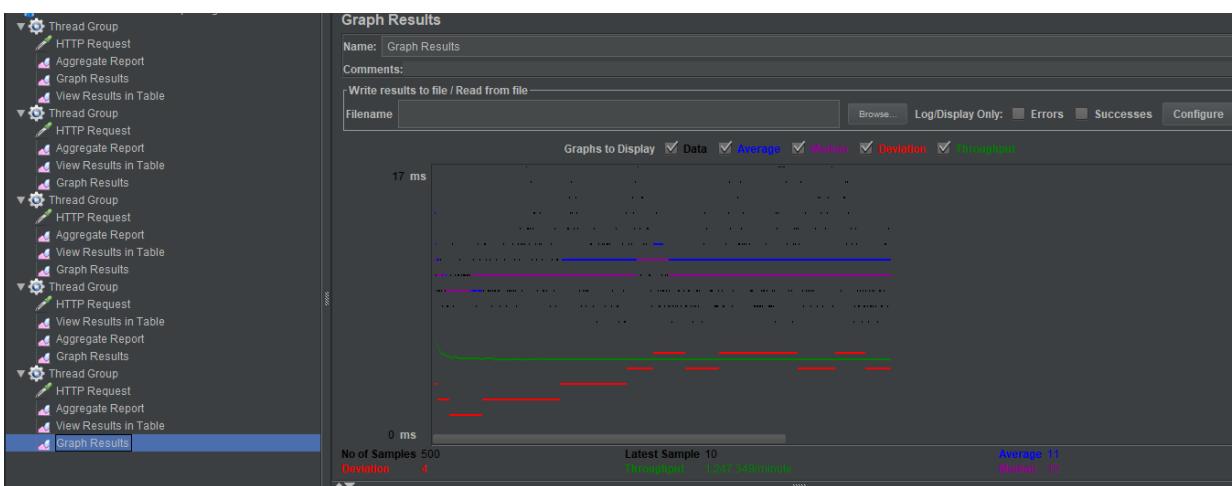
Tree View:

- Thread Group
 - HTTP Request
 - Aggregate Report
 - Graph Results
 - View Results in Table**
- Thread Group
 - HTTP Request
 - Aggregate Report
 - View Results in Table
 - Graph Results
- Thread Group
 - HTTP Request
 - Aggregate Report
 - View Results in Table
 - Graph Results
- Thread Group
 - HTTP Request
 - Aggregate Report
 - View Results in Table**
 - Graph Results

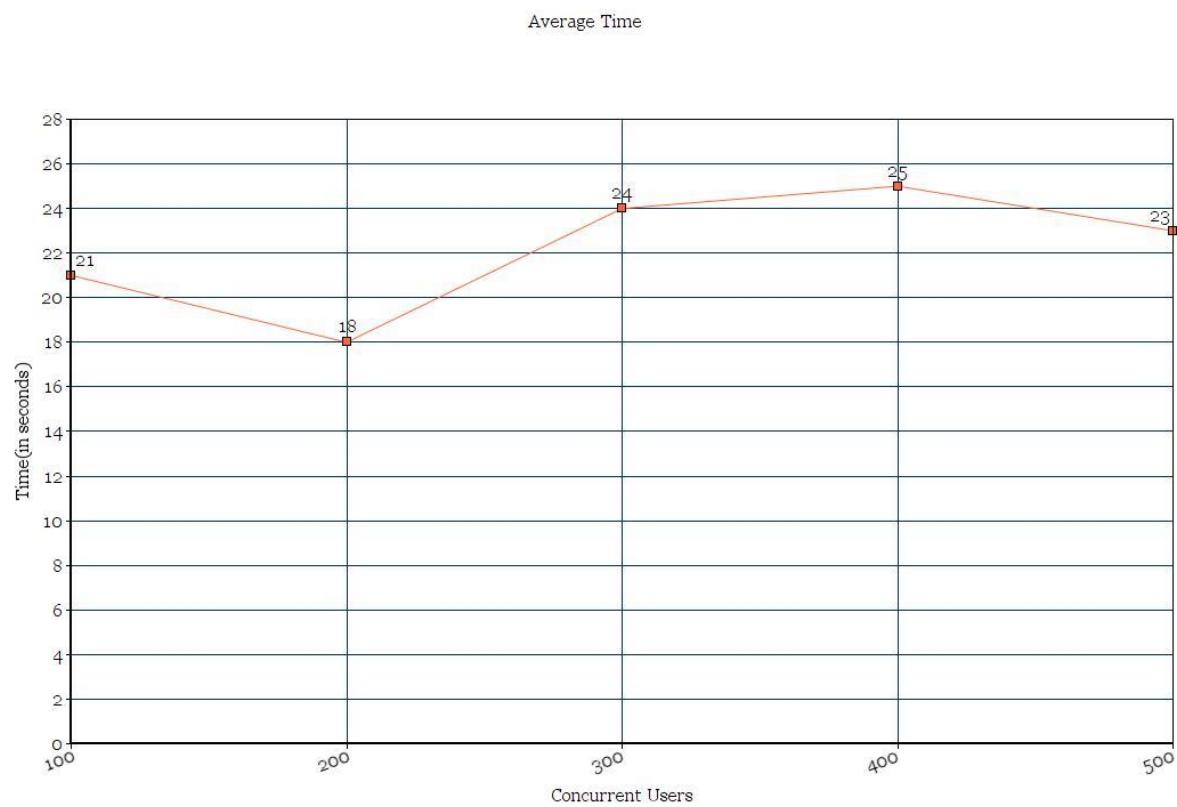
Table View (Sample # 476 to 500):

Sample #	Start Time	Thread Name	Label	Sample Time(...)	Status	Bytes	Sent Bytes	Latency	Connect Time(...)
476	07:20:00.709	Thread Group ...	HTTP Request	7	✓	712	133	7	1
477	07:20:00.757	Thread Group ...	HTTP Request	9	✓	712	133	9	1
478	07:20:00.805	Thread Group ...	HTTP Request	8	✓	712	133	8	1
479	07:20:00.852	Thread Group ...	HTTP Request	8	✓	710	133	8	1
480	07:20:00.900	Thread Group ...	HTTP Request	9	✓	710	133	9	1
481	07:20:00.911	Thread Group ...	HTTP Request	8	✓	716	133	8	1
482	07:20:00.997	Thread Group ...	HTTP Request	9	✓	712	133	9	1
483	07:20:01.048	Thread Group ...	HTTP Request	10	✓	710	133	10	1
484	07:20:01.093	Thread Group ...	HTTP Request	8	✓	714	133	8	1
485	07:20:01.140	Thread Group ...	HTTP Request	10	✓	712	133	10	1
486	07:20:01.187	Thread Group ...	HTTP Request	7	✓	714	133	7	1
487	07:20:01.235	Thread Group ...	HTTP Request	10	✓	714	133	10	1
488	07:20:01.283	Thread Group ...	HTTP Request	9	✓	710	133	9	1
489	07:20:01.331	Thread Group ...	HTTP Request	9	✓	710	133	9	1
490	07:20:01.380	Thread Group ...	HTTP Request	8	✓	710	133	8	1
491	07:20:01.428	Thread Group ...	HTTP Request	8	✓	710	133	8	1
492	07:20:01.477	Thread Group ...	HTTP Request	24	✓	716	133	24	4
493	07:20:01.523	Thread Group ...	HTTP Request	9	✓	714	133	9	1
494	07:20:01.576	Thread Group ...	HTTP Request	12	✓	716	133	12	1
495	07:20:01.624	Thread Group ...	HTTP Request	12	✓	710	133	12	2
496	07:20:01.672	Thread Group ...	HTTP Request	10	✓	712	133	10	1
497	07:20:01.721	Thread Group ...	HTTP Request	8	✓	710	133	8	1
498	07:20:01.769	Thread Group ...	HTTP Request	13	✓	712	133	13	1
499	07:20:01.818	Thread Group ...	HTTP Request	19	✓	710	133	19	1
500	07:20:01.885	Thread Group ...	HTTP Request	10	✓	710	133	10	1

Buttons: Scroll automatically? Child samples? No of Samples: 500 Latest Sample 10 Average: 11 Deviation: 4



Average Time



Mocha test results for 5 randomly selected calls

```
> mocha

✓ checking if profile is edited (126ms)
✓ checking if profile is fetched
✓ checking if properties are returned (171ms)
✓ checking if answer is posted (93ms)
✓ checking if pic is uploaded (98ms)

5 passing (519ms)
```

Question and Answers:

Q) Compare passport authentication process with the authentication process used in Lab1.

Ans:

In Lab1 , client sessions were used and nothing was used for storing sessions. Due to this session used to get destroyed as when the browser is closed. Even though this method was able to store user password and username in session however it is not scalable for many servers and all routes of REST API calls has to be authenticated again for user.

In Lab2, express sessions are used with passport local strategy with node js. Sessions are stored in Mongo DB. When user logs in, the session information is stored in mongo DB. After that it initializes passport with local strategy and authenticates the user. Now even if the tabs are closed and the application is open on other tab in same browser, user wont be redirected to login page. Instead he will be directly shown home page. Here Mongo DB is used to check saved username and password. Then it calls serialize method to attach user object which has user authentication information to session. Request is made for user session data with passport user data. Now for all other requests from browser, a session object with user is sent. Passport's initialize method is called every time for a request and user is verified. Passport authentication is independent of the server which receives request. Passport is very useful for achieving scalability.

Q: Compare performance with and without Kafka. Explain in detail the reason for difference in performance.

Ans:

The performance of the application is observed to be slower without kafka than with kafka. It is happening because kafka guarantees fast message delivery for scalable framework. Message load balancing help for fast retrieval of messages using kafka. Kafka framework has topics, brokers , producer and consumers and zookeeper to manage message delivery and scalability. Zookeeper is sharing the state and is highly available if all brokers fails.

Q: If given an option to implement MySQL and MongoDB both in your application, specify which data of the applications will you store in MongoDB and MySQL respectively

Ans:

Both the databases, MongoDB and MySQL have their pros and cons. MySQL is good for highly relational database. MongoDB is not an easy replacement for legacy systems that were built for relational databases.

Freelancer application requires multi-row transactions for several functionalities, thus better suited for a relational database.

On the other hand, MongoDB is suitable for environments with high write load because it supports high insert rate and doesn't have to worry about transaction safety, unlike MySQL which locks up the entire database and cause performance issues during insert operation. As the application contains multiple insertions and updates as user performs actions on mainly project data. Looking at this scenario, project, bid and transaction related data should be in MongoDB and User related data should be in MySQL, but this again creates problem when mapping of project and user data or bid and user data is to be done as data needs to be fetched from both, Mongo and MySQL.

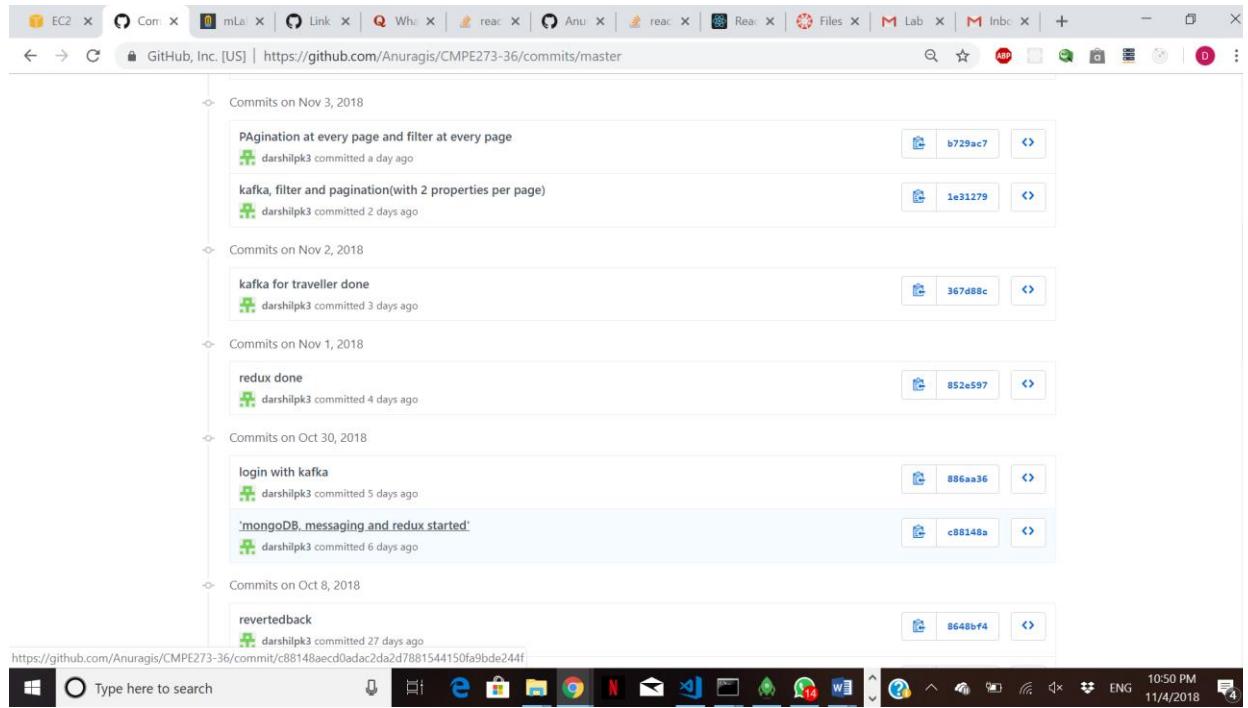
Third and one of the most important factor is of scaling. MongoDB supports built-in replication, sharding, and auto-elections. Using auto-elections, you can set up a secondary database to automatically take over if the primary database fails. Sharding allows for horizontal scaling, which is difficult to implement in MySQL.

But as the application developed was small and need for good relational database is prominent, even with all above advantages of MongoDB I would choose to use MySQL for storing all databases namely User, Project, Bid and Payment_history and only session data in MongoDB.

Commit History

The screenshot shows a GitHub commit history for the repository 'Anuragis / CMPE273-36'. The commits are listed in chronological order from Nov 4, 2018. The commits are:

- "Completed" by darshilpk3 committed 6 hours ago. SHA: b8c8f82
- "Changes made to run on deployment" by darshilpk3 committed 7 hours ago. SHA: 823c62c
- migrated to localStorage by darshilpk3 committed 21 hours ago. SHA: f20327c
- Create .gitignore by darshilpk3 committed 23 hours ago. Verified. SHA: 0463781
- Correcting few things by darshilpk3 committed 23 hours ago. SHA: 7415a90
- Create .gitignore by darshilpk3 committed a day ago. Verified. SHA: 0e8e7a8
- Create .gitignore by darshilpk3 committed a day ago. SHA: f4fe8c5



Github Link: <https://github.com/Anuragis/CMPE273-36.git>

Extra - Requirement:

Owner with more than 5 bookings: email: d@k, password: 123
Traveler with more than 5 bookings: email: d@k, password:123

AWS account information:

Email: darshilpareshbhai.kapadia@sjtu.edu
Password: Maruti_800