

SOFTWARE ENGINEERING DEPARTMENT

FALL 2016



SAN JOSÉ STATE UNIVERSITY

**CMPE 273 – Enterprise Distributed Systems
Airbnb Simulation With Auction**

Team Member	SJSU ID
Disha Sheth	011498240
Harsh Mehta	011429912
Manu Barsainyan	011457472
Bhavan Pandya	011456991
Aishwarya Keerty	011091574

CONTENTS

- 1) About Airbnb
- 2) Individual Member Contributions
- 3) Object Management Policy
- 4) Goal of Project
- 5) Architecture
- 6) Screenshots: Various UI pages
 - 6.1 User Module
 - 6.2 Host Module
 - 6.3 Admin
 - 6.3.1 Analytics
 - 6.4 Bidding
- 7) Salient Features
- 8) Testing Graphs
- 9) Comparison Graphs
- 10) Bibliography

About Project

The project is a prototype of Airbnb, a web portal for booking a house on-the-go.



About Airbnb

Airbnb is an online marketplace that enables people to list, find, then rent vacation homes for a processing fee. It has over 2,000,000 listings in 34,000 cities and 191 countries. Founded in August 2008 and headquartered in San Francisco California.

Shortly after moving to San Francisco in October 2007, Brian Chesky and Joe Gebbia Created the initial concept for AirBnb & Breakfast during the Industrial Design Conference held by Industrial Designers Society of America. The original site offered short-term living quarters, breakfast, and a unique business networking opportunity for those who were unable to book a hotel in the saturated market.

Airbnb is a peer-to-peer accommodation marketplace that connects hosts (vendors of rooms/accommodations) and travelers via its website. Airbnb enables transactions between these two entities by charging a 'service fee' without directly owning any room by itself.

Users of the site must register and create a personal online profile before using the site. Every property is associated with a host whose profile includes recommendations by other users, reviews by previous guests, as well as a response rating.

Individual Member Contributions:

1. Disha Sheth

- Front end and back end development of user dashboard, listings and trips.
- Home page
- Search implementation
- Back end development of Trips module/services.
- Implementation of RabbitMQ.
- Designing of database.
- JMeter Testing.

2. Harsh Mehta

- Implementation of UI Routing.
- Front end and back end development of property listing.
- Implementation of checkout.
- Front end development of Trips module/services.
- Integration of different modules/services.
- Implementation of RabbitMQ.
- Implementation of Redis.
- Addition of multiple users, hosts and billing.

3. Manu Barsainyan :

- Front end and back end development of billing module/services.
- Front end and back end development of host dashboard, property listing, image and video upload module/services.
- Front end and back end development of bidding module/services.
- Front end and back end development of review module/services.
- Implementation of RabbitMQ.
- Mocha Testing.

4. Bhavan Pandya:

- Front end and back end design and development of Admin module.
- Front end and back end design and development of Admin and Host analytics module.
- Generating logic for log analysis.
- Implementation of RabbitMQ.
- Designing of database.

5. Aishwarya Keerty:

- Front end and back end design and development of login and register modules/services.
- Generation of Project Report.

Object Management Policy

1. Requirement Analysis:

Depending on the analysis of the requirements of the project, there are 3 tiers. They are:

- a) **The Client tier:** In this tier, the user will interact with the system.
- b) **The Middle tier/middleware/messaging system:** In this tier majority of the processing takes place. This tier supports the sending and receiving of messages between distributed systems. Some of the modules in this tier have been implemented using RabbitMQ which is an open source message broker software written in Erlang programming language that implements Advanced Message Queuing Protocol (AMQP).
- c) **The Third tier:** This tier comprises of a database to store the state of the entity objects of the system. Some modules in this tier use MySQL as the relational database while others use MongoDB as the NoSQL database to store data.

2. Functional Requirements Considerations:

In this project different modules such as user module, host module, billing module, admin module, trips module have been implemented. Each of the modules has an associated schema which shows how data is stored in the database.

3. Development:

Some of the key features in the project like add property helps user select property in their location and within required price range. Other features like ratings and reviews allows the host and client to rate their experiences. RabbitMQ helps to increase the sustainability when the load is high.

4. Testing the System:

a) Handling “Heavy Weight” resources:

Loading heavyweight resources like images and videos takes a significant amount of time. Hence, these heavyweight resources are stored in MongoDB. Retrieving images and videos from MongoDB is much more time efficient than retrieving them from MySQL.

b) Policy to decide when to write data into the database:

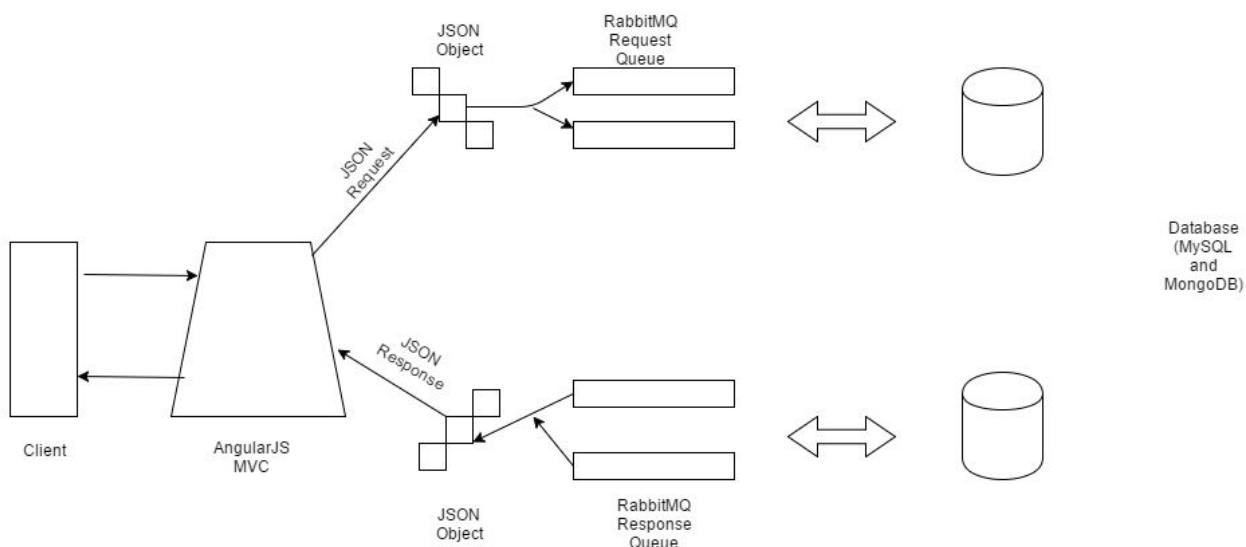
Data has been inserted into database optimally. Mongo has been used to store data like ratings, reviews, property details and sessions. Data relating to User, host, admin and billing details have been stored in MySQL. Introduction videos and host images have also been stored in MongoDB for faster query access.

Goal of Project

- 1) We have to understand the functionality of the REST API and implement Group Project using the nodejs, angularjs, HTML5 and bootstrap.
- 2) Develop an attitude of working in team and developing a scalable enterprise level application.
- 3) Understand, optimise and handle the functioning of the actual enterprise application.

Architecture of Application

Entire n-tier application is developed using MEAN(MongoDB, Express, AngularJS and NodeJS) which is further optimised using MySQL, connection pooling, RabbitMQ and Redis Caching.

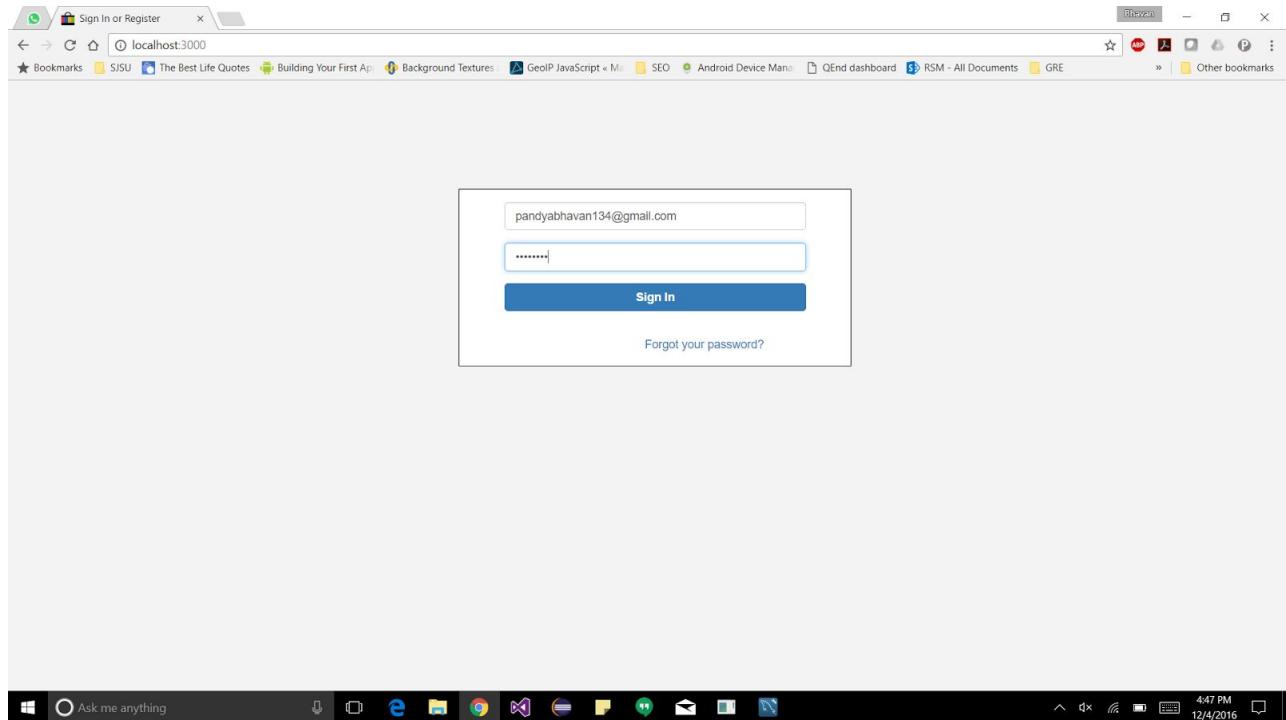


express

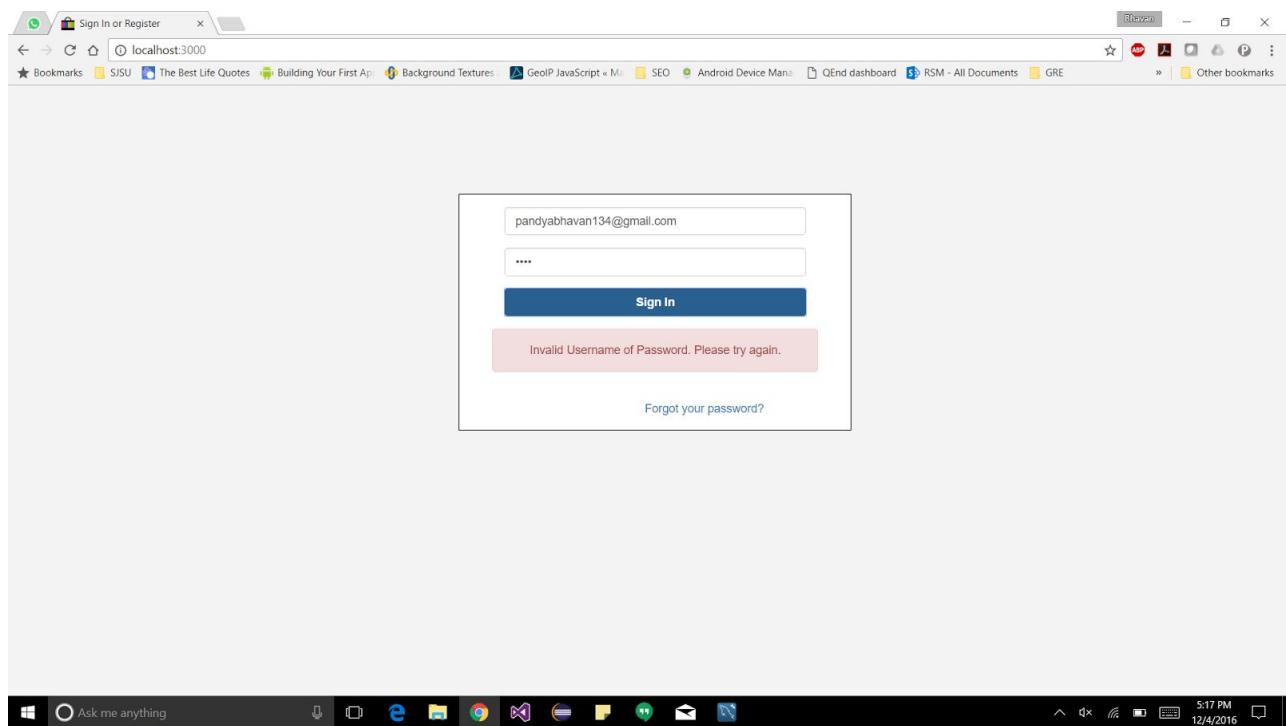


Screen Captures: Various UI Pages

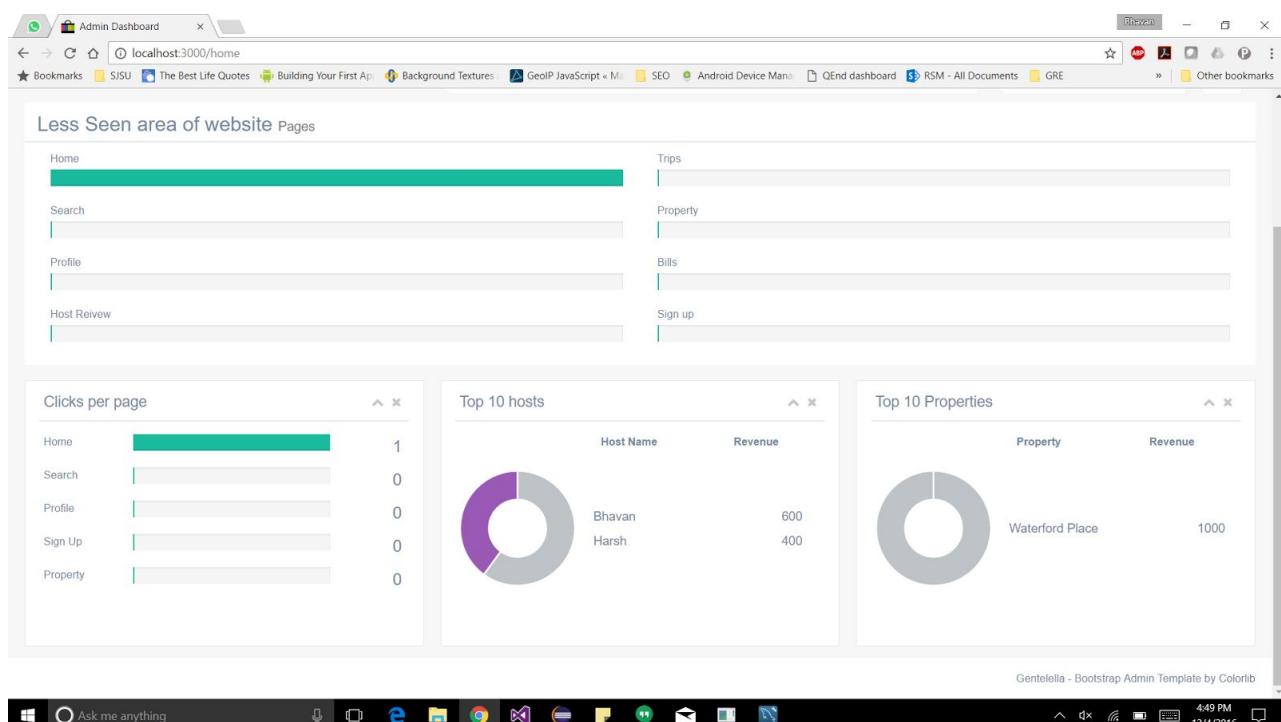
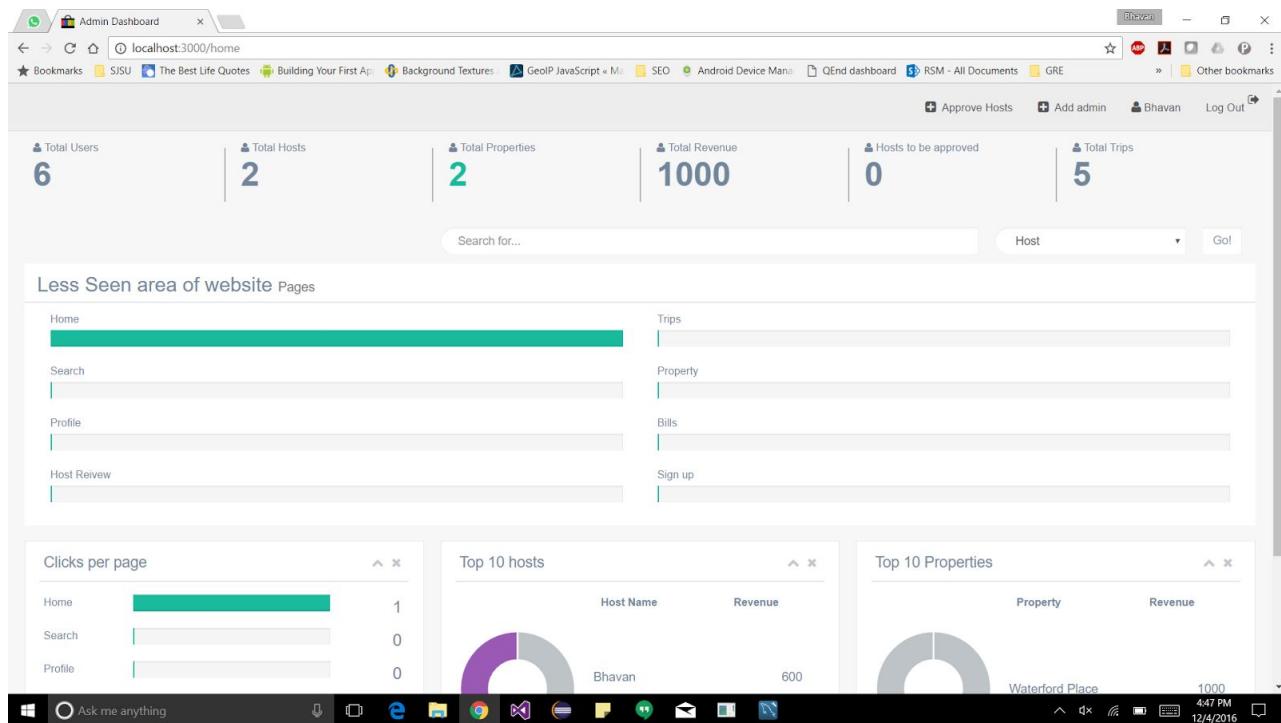
1. Admin Login Page: The admin(s) can log in using their email id and password.



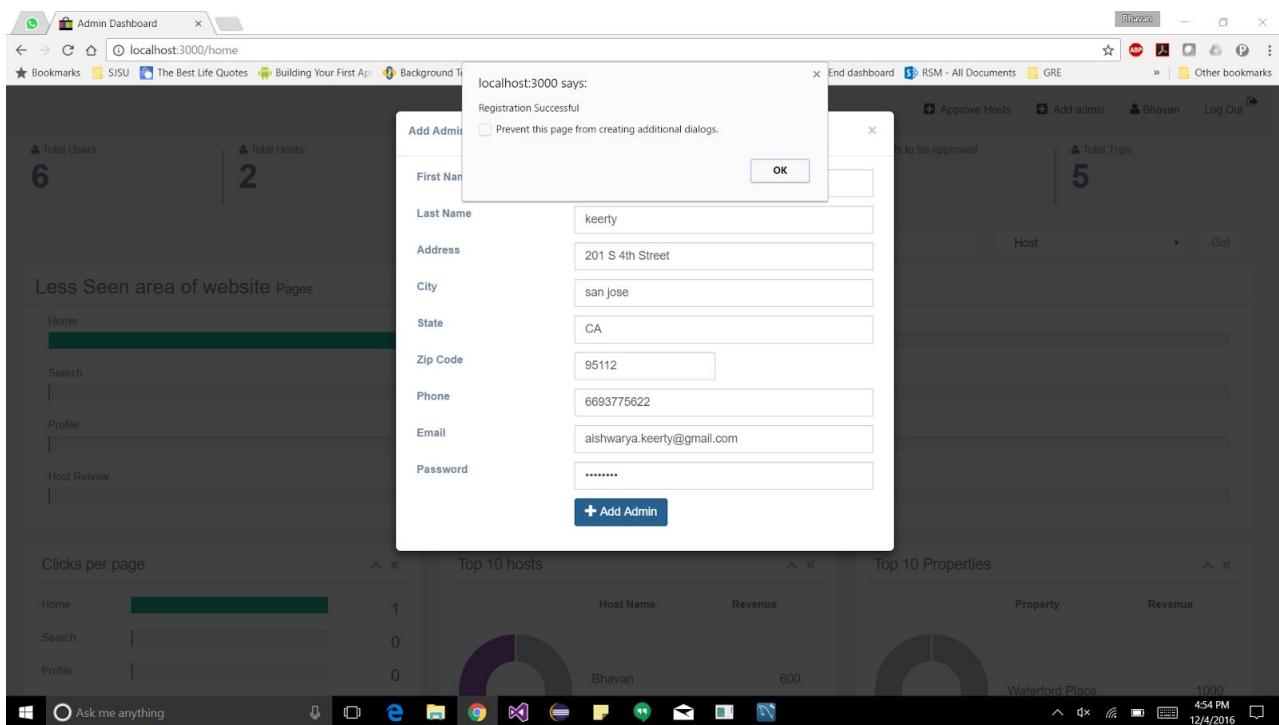
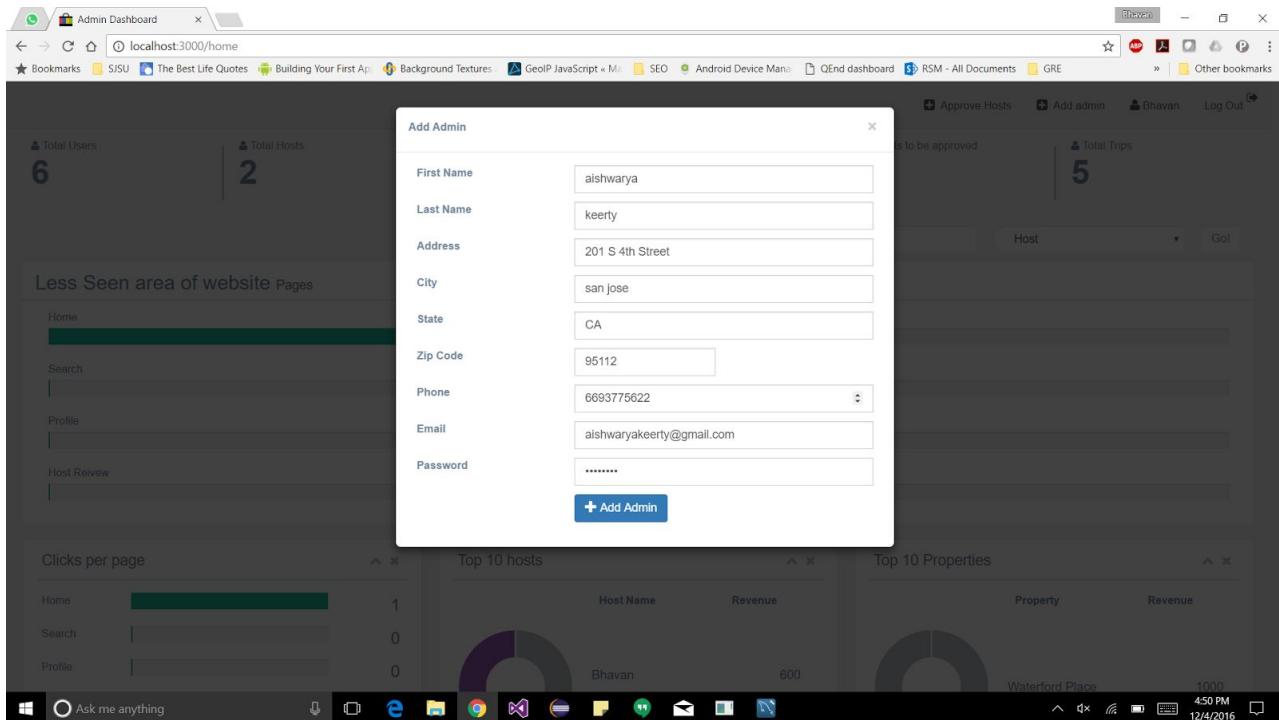
2. Invalid admin validation: Users with invalid details will be displayed error message.



3. Admin's Dashboard: The admin dashboard allows admin to approve hosts, add other admins, check the total revenue, trips, properties, users, clicks per page etc.



4. Add Admin: An admin can add one or more admins for the website.

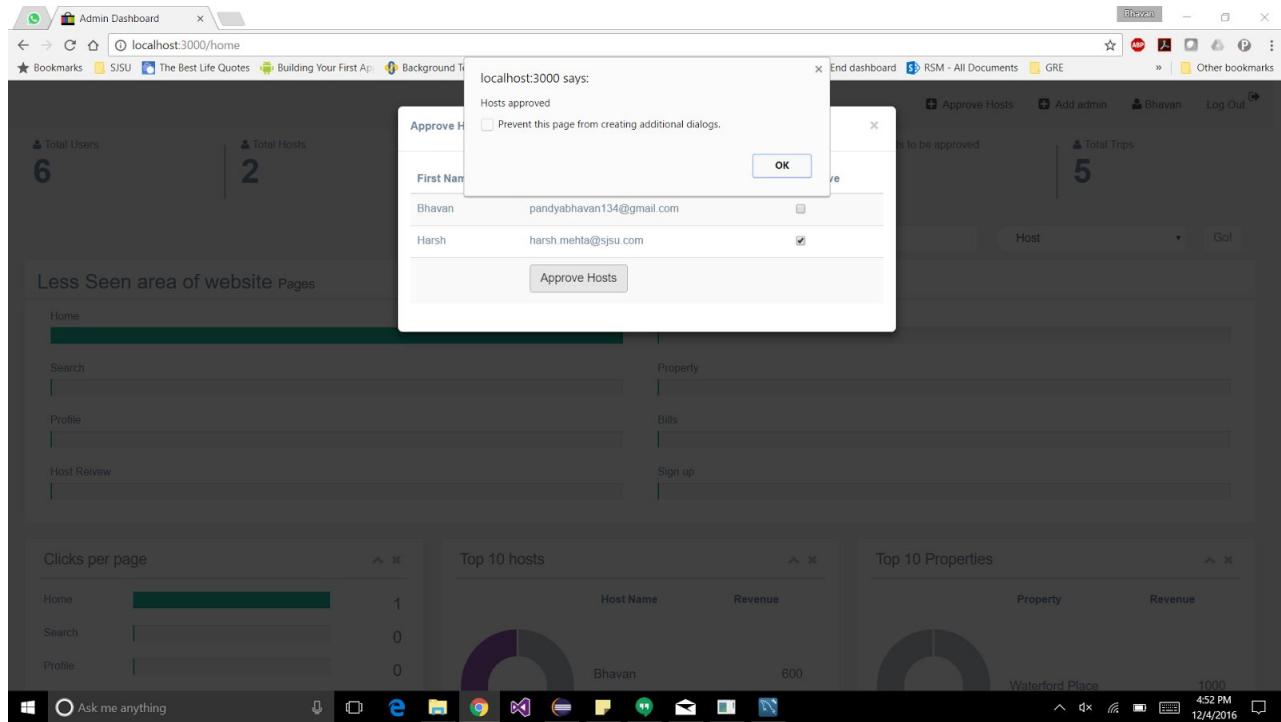


5. Update Profile: An admin can update his profile.

The screenshot shows a dark-themed Admin Dashboard. In the center, a modal window titled "Update Profile" is open, containing fields for First Name (Bhavan), Last Name (Pandy2), Address (101 E San Fernando), City (San Jose), State (CA), Zip Code (95112), Phone (6692715179), and Email (pandyabhanv134@gmail.com). A blue "Update Profile" button is at the bottom right of the modal. The background dashboard features sections for "Total Users" (6), "Total Hosts" (2), and "Less Seen area of website Pages" (Home, Search, Profile, Host Review). Below these are three charts: "Clicks per page" (Home: 1, Search: 0, Profile: 0), "Top 10 hosts" (Bhavan, Revenue: 600), and "Top 10 Properties" (Waterford Place, Revenue: 1000). The top navigation bar includes links for "Approve Hosts", "Add admin", "Bhavan", and "Log Out". The system status bar at the bottom shows "4:51 PM 12/4/2016".

6.Approve Hosts: The admin can approve pending Host requests.

This screenshot shows the same Admin Dashboard as the previous one, but with a different modal. The "Approve Hosts" modal lists two entries: Bhavan (Email: pandyabhanv134@gmail.com) and Harsh (Email: harsh.mehta@sjsu.com), each with an "Approve" checkbox checked. A blue "Approve Hosts" button is at the bottom right of the modal. The background dashboard elements are identical to the first screenshot, including the "Total Users" (6), "Total Hosts" (2), and various charts. The top navigation bar and system status bar are also present.



7. Admin Database:

Result Grid:

ID	frame	name	address	city	state	zipcode	phone	email	password	profile_image
12345678	Bhavan	Pandya2	101 E San Fernando	San Jose	CA	95112	6692715179	pandyabhavan134@gmail.com	\$2a\$10\$JwrxthMWhXmktaIyshzOJWS/sWLV...	blob
12345679	Harsh	Mehtha	101 E San Fernando	San Jose	CA	95112	6692734179	harsh.mehta@sjsu.com	\$2a\$10\$JwrxthMWhXmktaIyshzOJWS/sWLV...	blob
12345680	ashwarya	keerty	2015 4th Street	san jose	CA	95112	6693775622	ashwarya.keerty@gmail.com	\$2a\$10\$JwrxthMWhXmktaIyshzOJWS/sWLV...	blob

8. Search: The admin can search for hosts and bills in the admin dashboard.

The screenshot shows a custom admin dashboard interface. At the top, there are six summary cards with the following data:

- Total Users: 13
- Total Hosts: 2
- Total Properties: 2
- Total Revenue: 1000
- Hosts to be approved: 0
- Total Trips: 5

Below these cards is a search bar with the placeholder "harsh" and a dropdown menu set to "Host". A "Go!" button is located to the right of the search bar.

The main content area is titled "Less Seen area of website Pages" and contains two sections of horizontal bar charts:

- Left Section:**
 - Home: Long teal bar (~80%)
 - Search: Short grey bar (~10%)
 - Profile: Very short grey bar (~5%)
 - Host Review: Very short grey bar (~5%)
- Right Section:**
 - Trips: Short grey bar (~10%)
 - Property: Short grey bar (~10%)
 - Bills: Short grey bar (~10%)
 - Sign up: Very short grey bar (~5%)

Below these sections are three data visualizations:

- Clicks per page:**

Page	Clicks
Home	1
Search	0
Profile	0
- Top 10 hosts:**

Host Name	Revenue
Bhavan	600
- Top 10 Properties:**

Property	Revenue
Waterford Place	1000

The taskbar at the bottom shows various icons and the date/time: 12/4/2016, 6:02 PM.

9. Home Page: The home page allows the login and signup of users and hosts.

The screenshot shows the Airbnb homepage. The top navigation bar includes the Airbnb logo, a search bar with the placeholder "Where to?", and account-related links: "Become a Host", "Help", "Messages", and "Hi, harsh".

The main headline reads: "Live there. Book unique homes and experience a city like a local."

Below the headline are three input fields for search parameters:

- Where:** Destination, city, address
- When:** mm/dd/yyyy, mm/dd/yyyy
- Guests:** 1 Guest

A red "Search" button is positioned to the right of these fields.

The "Explore the world" section features five thumbnail images:

- A large image of the Eiffel Tower at sunset.
- A street scene with people and colorful balloons.
- A night sky with fireworks over a beach.
- A city skyline with a prominent dome.
- A lush green landscape with trees.

The taskbar at the bottom shows various icons and the date/time: 12/4/2016, 4:38 PM.

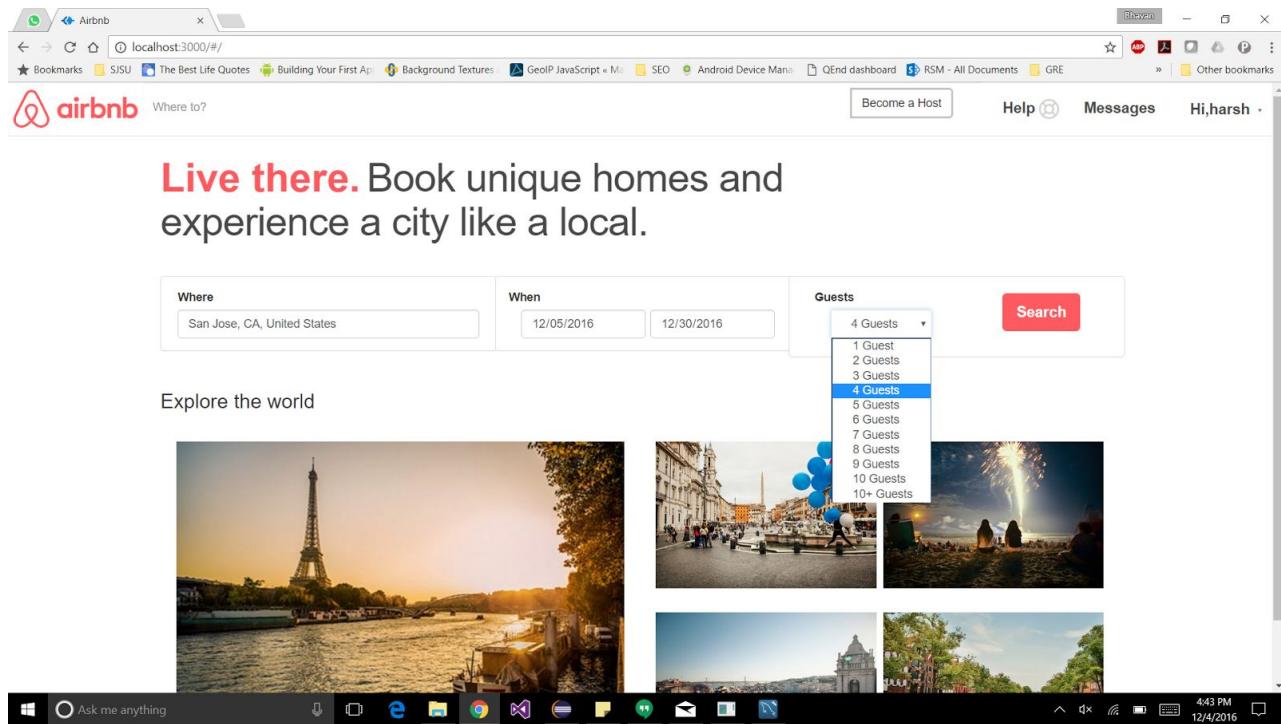
Screenshot of a web-based application dashboard titled "Admin Dashboard". The dashboard shows statistics: Total Users (13), Total Hosts (2), and Total Trips (5). A search modal is open, displaying a single result: ID 234567891, First Name Harsh, and Email harsh.mehta@sjsu.com. Below the stats, there's a section titled "Less Seen area of website Pages" with links to Home, Search, Profile, and Host Review. On the right, there are sections for Trips, Property, Bills, and Sign up. At the bottom, three charts show Clicks per page, Top 10 hosts, and Top 10 Properties. The taskbar at the bottom shows various icons and the date/time as 12/4/2016 6:03 PM.

10. Home page validations:

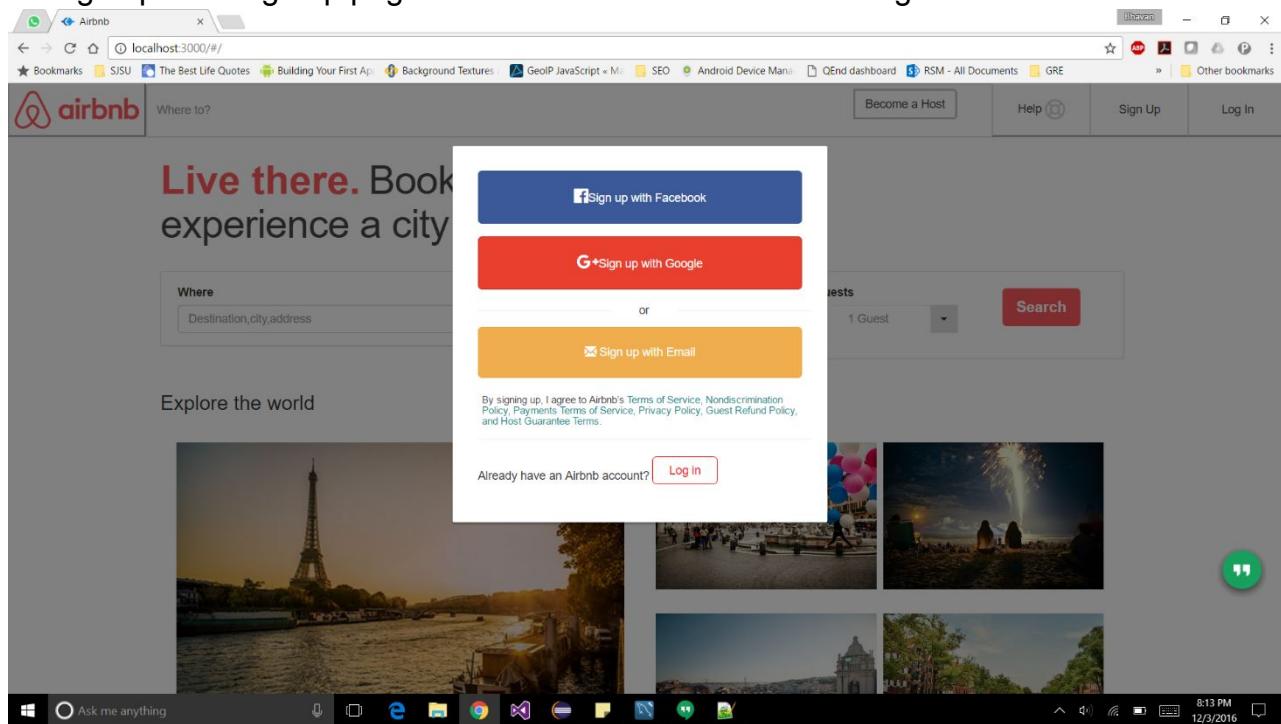
Screenshot of the Airbnb homepage. The header features the Airbnb logo and the tagline "Where to?". Below the header, a large call-to-action button says "Live there. Book unique homes and experience a city like a local." The search form includes fields for "Where" (with a dropdown menu showing locations like San Jose, CA, United States; Santa Clara, CA, United States; Sunnyvale, CA, United States; San Jose International Airport Long Term Parking, Airport, CA, United States; Santana Row, San Jose, CA, United States), "When" (date pickers), "Guests" (dropdown), and a red "Search" button. Below the search form, there are several thumbnail images of travel destinations: a sunset over the Eiffel Tower in Paris, a street in Rome with balloons, fireworks over a beach, a building in a city, and a view of a park. The taskbar at the bottom shows various icons and the date/time as 12/4/2016 4:42 PM.

Screenshot of the Airbnb homepage on a Windows desktop. The browser title bar says "Airbnb". The address bar shows "localhost:3000/#/". The search bar has "Where" set to "San Jose, CA, United States", "When" set to "12/03/2016" (invalid date), and "Guests" set to 1. A red "Search" button is visible. The main headline reads "Live there. Book unique homes and experience a city like a local." Below it is a search bar with fields for location, date, and guests. A "Explore the world" section features five thumbnail images: Eiffel Tower at sunset, a street with balloons, fireworks over water, a building with greenery, and a forest scene. The taskbar at the bottom shows various pinned icons and the date/time as 12/4/2016 4:42 PM.

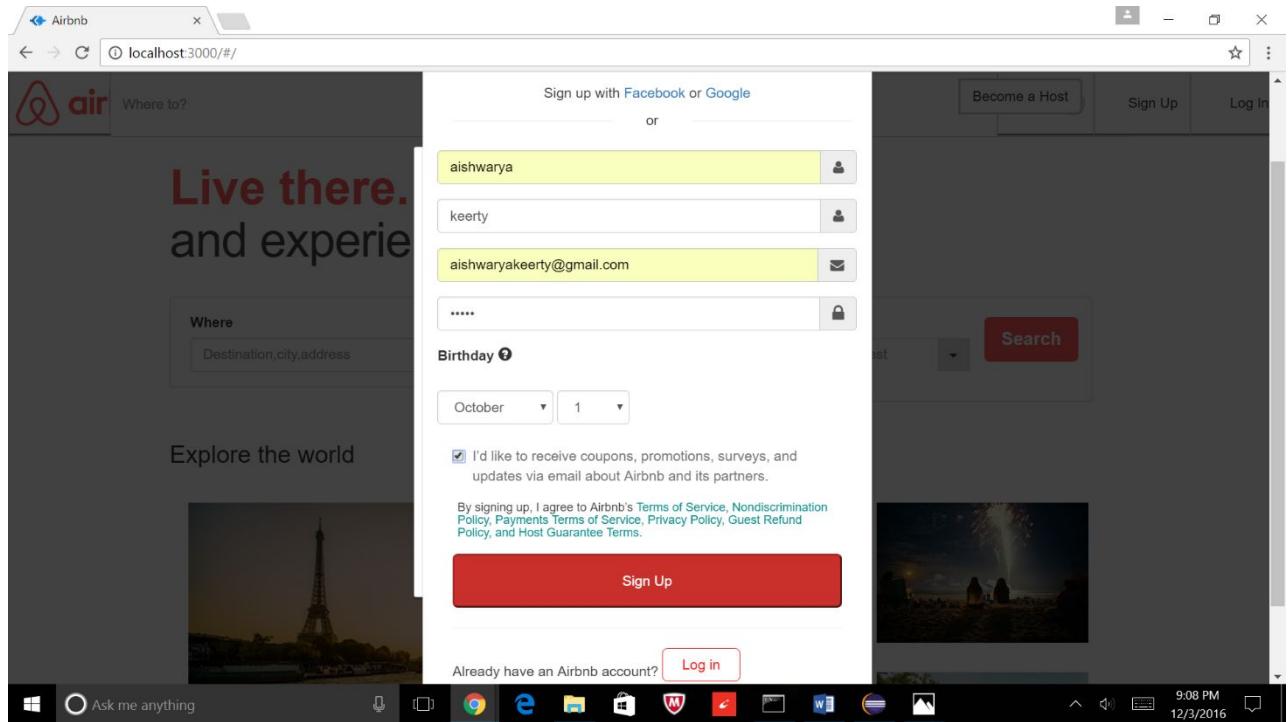
Screenshot of the Airbnb homepage on a Windows desktop. The browser title bar says "Airbnb". The address bar shows "localhost:3000/#/". The search bar has "Where" set to "San Jose, CA, United States", "When" set to "12/05/2016" (valid date) and "12/30/2016" (valid date), and "Guests" set to 1. A red "Search" button is visible. The main headline reads "Live there. Book unique homes and experience a city like a local." Below it is a search bar with fields for location, date, and guests. A "Explore the world" section features five thumbnail images: Eiffel Tower at sunset, a street with balloons, fireworks over water, a building with greenery, and a forest scene. The taskbar at the bottom shows various pinned icons and the date/time as 12/4/2016 4:43 PM.



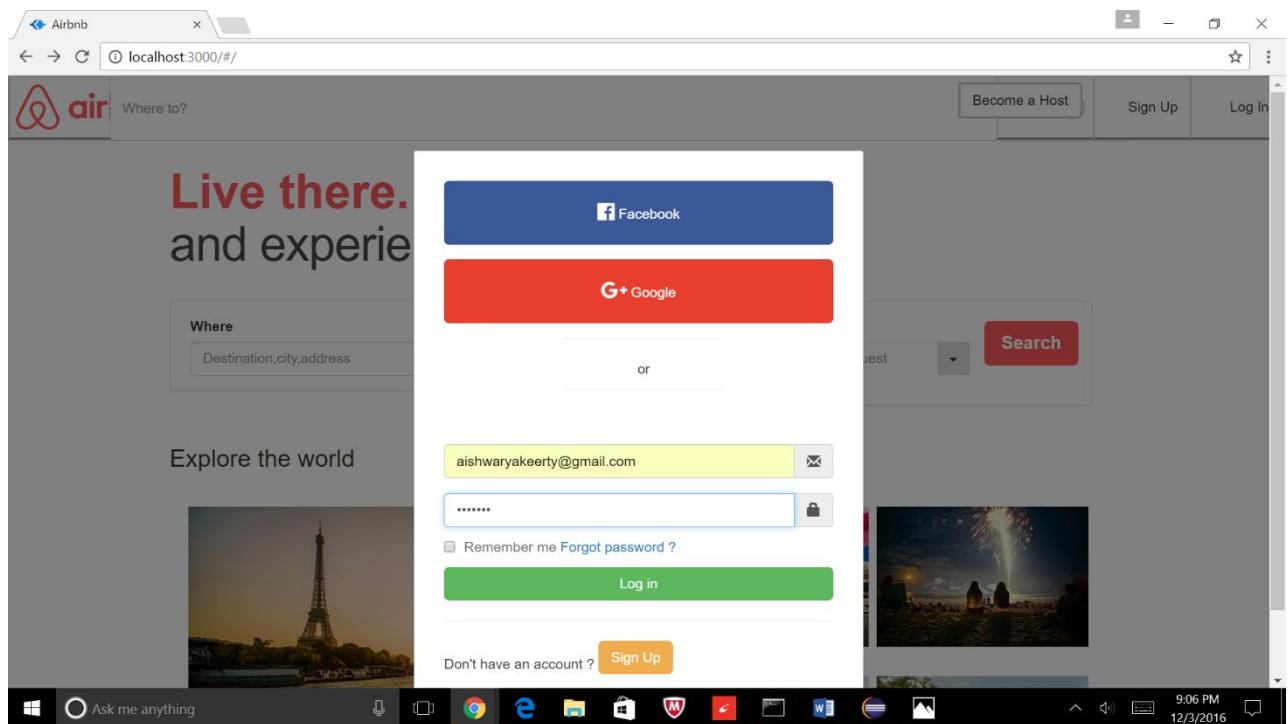
11 Sign up: The sign up page allows new hosts and users to register.

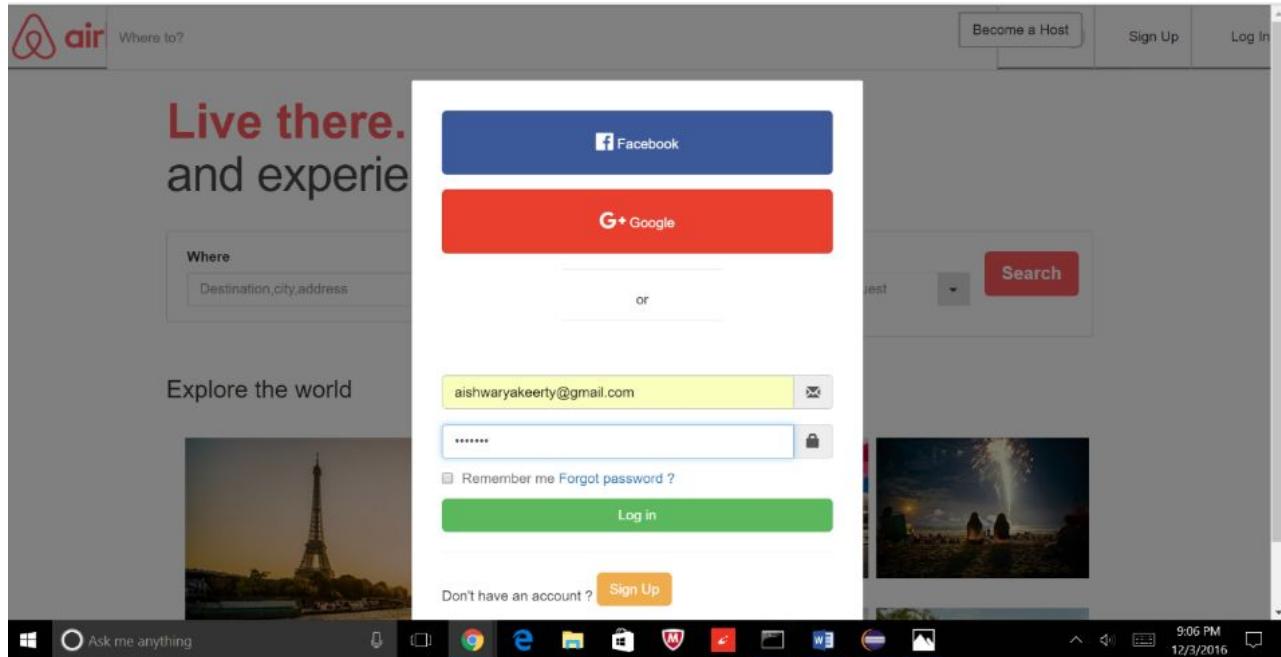


12.Sign up with Email: This allows the user/ host to sign up using Email.

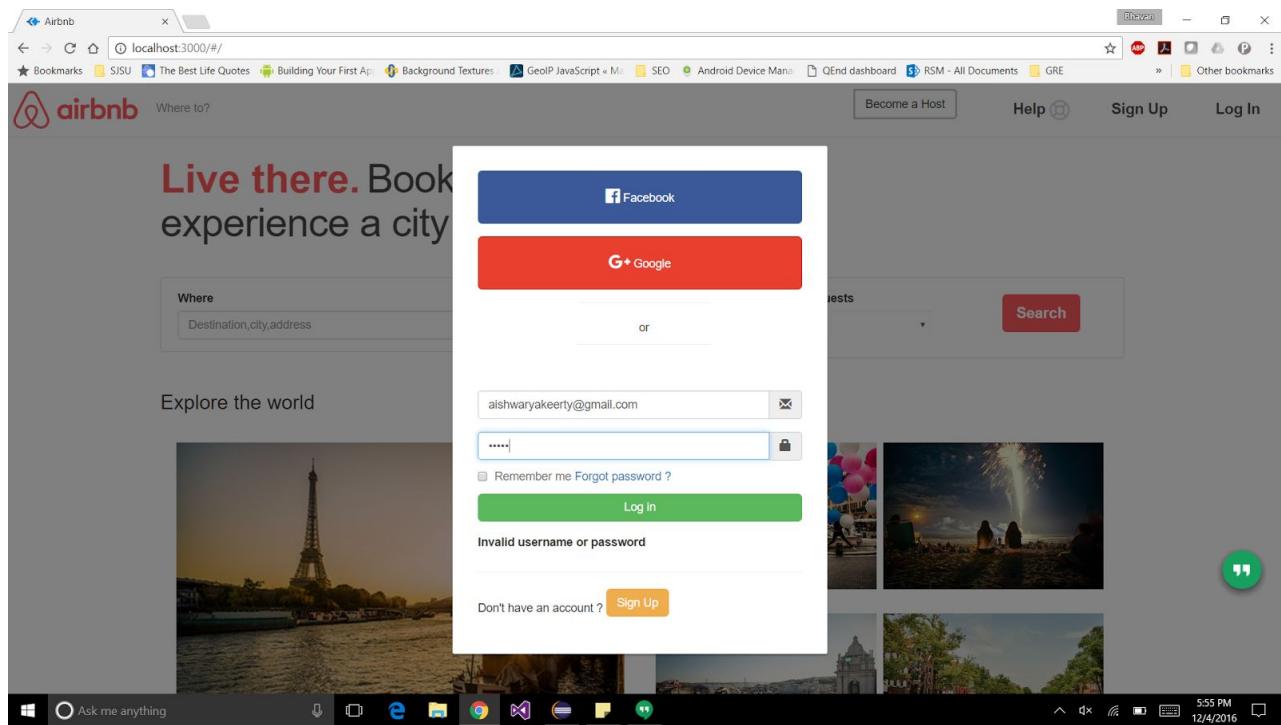


13.Login: The log in page allows existing user and hosts to log in.





14.Log in validation:



15.User Dashboard: The user dashboard allows the user to update profile, view listings, reservations.

Inbox (7,360) - dsheth18@ | Airbnb report.docx - Invit | Airbnb report.docx - Goo | Assignments: FA16: CMPE | Airbnb report.docx - Goo | Airbnb

localhost:3000/#/user_profile/123456780

Profile Your listings Your Trips

Edit Profile

View Profile

Required

First Name Harsh

Last Name Mehta

This is only shared once you have a confirmed booking with another Airbnb user.

I am Male

We use this data for analysis and never share it with other users.

Birth Date 12/4/2016

The magical day you were dropped from the sky by a stork. We use this data for analysis and never share it with other users.

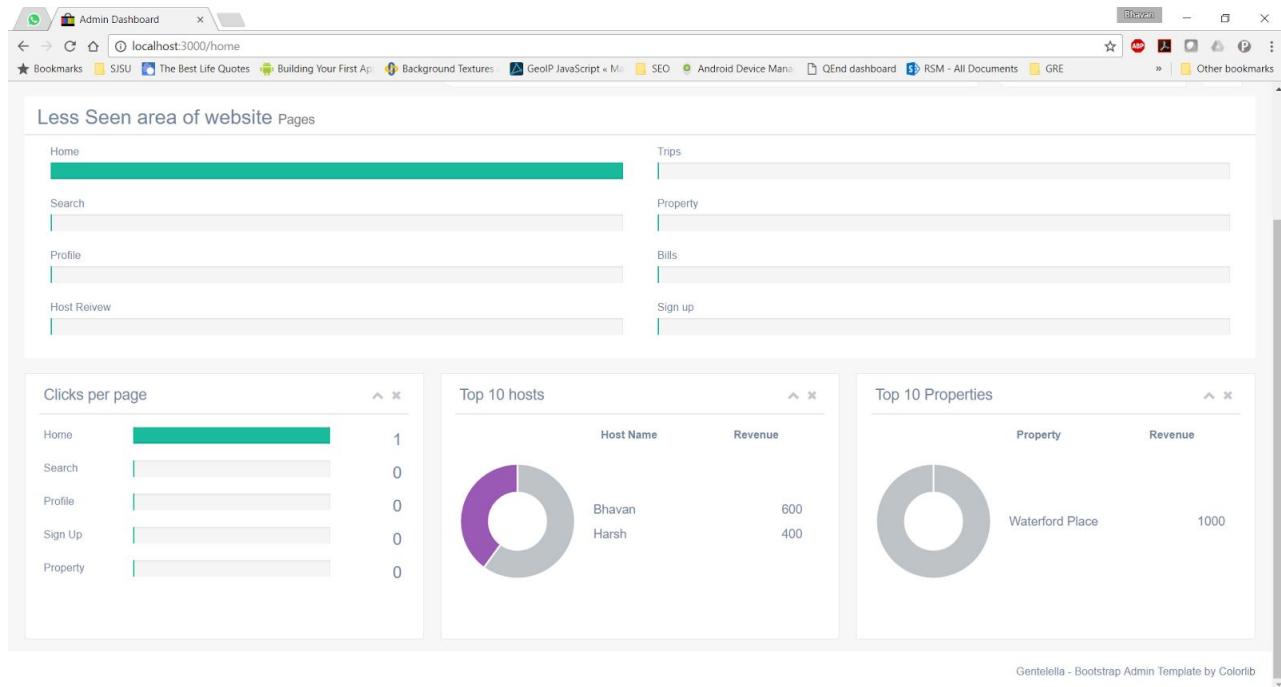
Email Address harsh.mehta@sjsu.com

Phone Number 6692734179

Where You Live 101 E San Fernanc undefined CA

Describe Yourself Traveler

16. Host Dashboard: The host dashboard allows the user to maintain stats, view listings, reservations, profile and account.



localhost:3000

Dashboard Stats Your Listing Your Reservations Profile Account

Your listing Summary ▾

Everyone's hosting style is different but we have some tips to help your listing standout.



The serene ideal property for family and working professionals
1700 N 10th St

Price Calendar

\$100 82 days open in next 3 months

Your minimum price

At your current price 1700 N 10th St your listing is most likely to be booked

[Update price](#)

More Media

Add more photos to your listing. Listings with 7 or more photos usually get more bookings

[Add more photos](#)

Add detailed description

Users that added a more detailed description of their properties get more interest in their listings.

[Add Detailed description](#)

Dashboard Stats Your Listing Your Reservations Profile Account

Your images :



[Update image](#)



[Update image](#)



[Update image](#)

Upload more images :

Choose File No file chosen

[Upload image](#)

Your videos :



▶ 0:00 / 1:13 🔍 🔊 [Full screen]

[Update video](#)



▶ 0:00 / 1:13 🔍 🔊 [Full screen]

[Update video](#)

Upload more videos :

No file chosen

▶ 0:00

[Upload video](#)

[Your Listing](#)
[Your Reservations](#)
[Your Bookings](#)

WaterFord Place



The serene ideal property for family and working professionals

1700 N 10th St

Price :100

[Your Listing](#)
[Your Reservations](#)
[Your Bookings](#)

WaterFord Place

Bill No # 12345678

Reservation requested by:



Harsh Mehta

From: 12/15/2016

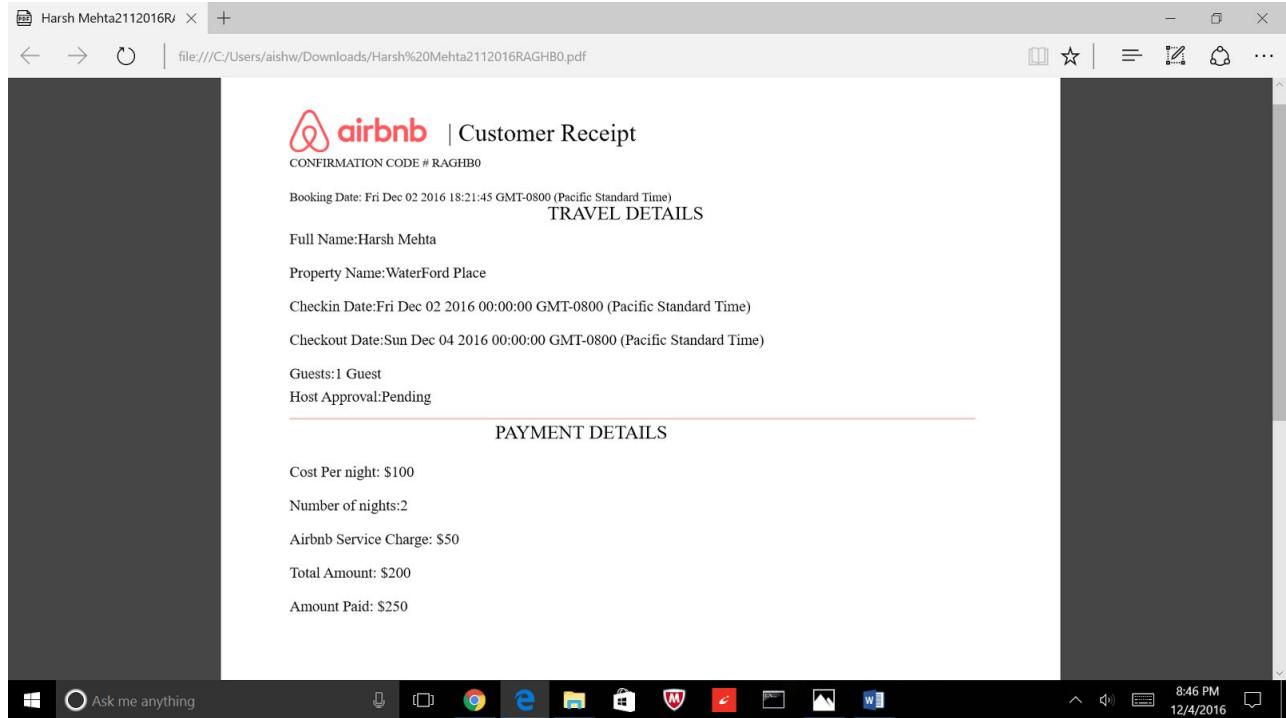
4.333333333333333 out of 5

To: 12/17/2016

[Approve request](#)

[Reject request](#)

16. Bill Details:



17. Host/User reviews:

Bhavan Pandya

3

Nan out of 5

★★★★★

Accuracy	★★★★★
Accuracy	★★★★★
Check in	★★★★★
Cleanliness	★★★★★
Communication	★★★★★
Location	★★★★★
Value	★★★★★

Harsh Mehta
12/13/2016

awesome experience

18. Write a review :

17. MySQL admin table:

	id	fname	lname	address	city	state	zipcode	phone	email	password	profile_image
1	12345678	Bhavani	Pandya	2 101 E San Fernando	San Jose	CA	95112	6692715179	pandyabhan134@gmail.com	password	URL
2	12345679	Harsh	Mehta	101 E San Fernando	San Jose	CA	95112	6692734179	harsh.mehta@spju.com	password	URL
3	12345680	ashwarya	keerty	2015 4th Street	san jose	CA	95112	6693775622	ashwarya.keerty@gmail.com	2a\$10\$J8wxrthMWhXmkIAyshzOJWS/sWLV...	URL

SQL History:

- 1 21:05:25 SELECT * FROM airbnb11.admin LIMIT 0, 1000
- 2 21:05:28 SELECT * FROM airbnb11.billing LIMIT 0, 1000
- 3 21:05:31 SELECT * FROM airbnb11.host LIMIT 0, 1000
- 4 21:05:33 SELECT * FROM airbnb11.property LIMIT 0, 1000
- 5 21:05:35 SELECT * FROM airbnb11.trip LIMIT 0, 1000
- 6 21:05:37 SELECT * FROM airbnb11.user LIMIT 0, 1000

18. MySQL host database:

The screenshot shows the MySQL Workbench interface with the 'host' table selected in the query editor. The results grid displays two rows of data:

id	name	Iname	description	address	city	state	zipcode	phone	email	password	cc_number	cvv	expiry
234567890	Bhavan	Pandya		101 E San Fernando	San Jose	CA	95112	6692715179	pandybhavan134@gmail.com	HULL	1234567890123456	123	06/2019
234567891	Harsh	Mehta	My name is Harsh and I live in Amsterdam. In as...	101 E San Fernando	San Jose	CA	95112	6692734179	harsh.mehta@sjtu.com	HULL	1234567890123456	123	06/2019

The 'Action Output' pane shows the history of SQL queries run on this table.

19. MySQL property Database:

The screenshot shows the MySQL Workbench interface with the 'property' table selected in the query editor. The results grid displays three rows of data:

id	host_id	ptitle	category	address	city	state	zipcode	guests	description	available_from	available_to	bidding	price	weekly	monthly	season
1	234567891	Waterford Place	Private room	1700 N First street	San Jose	California	95112	3	Private room with all required amenities	HULL	HULL	HULL	100	100	90	150
2	234567891	San Fernando	Private room	101 san fernando	San Jose	California	95112	1	Private room with all required amenities	HULL	HULL	HULL	200	200	180	300

The 'Action Output' pane shows the history of SQL queries run on this table.

20. MySQL trip table:

MySQL Workbench

Local instance MySQL57 (2) ×

File Edit View Query Database Server Tools Scripting Help

Navigator Schemas SQL File 3* admin billing host property trip user

SCHEMAS Filter objects

airbnb airbnb11 ebay sys test

Tables: admin billing host property trip user

SQL File 3* admin billing host property trip user

1 • SELECT * FROM airbnb11.trip;

Result Grid | Filter Rows: [] | Edit: [] | Export/Import: [] | Wrap Cell Content: []

Management Schemas Information

Table: user

Columns:

id	user_id	host_id	property_id	days	priceday	totalprice	from_date	to_date	bill_path
10000000	123456789	234567890	1	2	100	200	Fri Dec 02 2016 00:00:00 GMT-0800 (Pacific Sta... undefined	Sun Dec 04 2016 00:00:00 GMT-0800 (Pacific Sta... undefined	
10000001	123456789	234567890	1	2	100	200	Fri Dec 02 2016 00:00:00 GMT-0800 (Pacific Sta... Sun Dec 04 2016 00:00:00 GMT-0800 (Pacific Sta... undefined		
10000002	123456789	234567890	1	2	100	200	Fri Dec 02 2016 00:00:00 GMT-0800 (Pacific Sta... Sun Dec 04 2016 00:00:00 GMT-0800 (Pacific Sta... ./public/bills/Harsh Mehta2112016G8JR85.pdf		
10000003	123456789	234567891	1	2	100	200	Fri Dec 02 2016 00:00:00 GMT-0800 (Pacific Sta... Sun Dec 04 2016 00:00:00 GMT-0800 (Pacific Sta... ./public/bills/Harsh Mehta2112016SAVEY.pdf		

Action Output

#	Time	Action	Message	Duration
1	21:05:25	SELECT * FROM airbnb11.admin LIMIT 0, 1000	3 row(s) returned	0.000 sec / 0.000 sec
2	21:05:28	SELECT * FROM airbnb11.billing LIMIT 0, 1000	0 row(s) returned	0.000 sec / 0.000 sec
3	21:05:31	SELECT * FROM airbnb11.host LIMIT 0, 1000	2 row(s) returned	0.000 sec / 0.000 sec
4	21:05:33	SELECT * FROM airbnb11.property LIMIT 0, 1000	2 row(s) returned	0.000 sec / 0.000 sec
5	21:05:35	SELECT * FROM airbnb11.trip LIMIT 0, 1000	5 row(s) returned	0.000 sec / 0.000 sec
6	21:05:37	SELECT * FROM airbnb11.user LIMIT 0, 1000	13 row(s) returned	0.000 sec / 0.000 sec

Object Info Session

Ask me anything

9:06 PM 12/4/2016

21. MySQL user table:

MySQL Workbench

Local instance MySQL57 (2) ×

File Edit View Query Database Server Tools Scripting Help

Navigator Schemas SQL File 3* admin billing host property trip user

SCHEMAS Filter objects

airbnb airbnb11 ebay sys test

Tables: admin billing host property trip user

SQL File 3* admin billing host property trip user

1 • SELECT * FROM airbnb11.user;

Result Grid | Filter Rows: [] | Edit: [] | Export/Import: [] | Wrap Cell Content: []

Management Schemas Information

Table: user

Columns:

id	name	address	city	state	zipcode	phone	email	cc_number	cvv	expiry	avg_rating	num_rating	profile_image	password
123456780	Hersh Mehta	101 E San Fernando	San Jose	CA	95112	6692734179	harsh.mehta@sjtu.com	1234567890123456	123	06/2019	NULL	NULL/images/profile.JPG	NULL
123456789	Bhavan Pandya	101 E San Fernando	San Jose	CA	95112	6692719179	pandiyabhavan134@gmail.com	1234567890123456	123	06/2019	NULL	NULL	NULL	NULL
123456790	Bhavan Pandya	NULL	NULL	NULL	NULL	NULL	test@t.com	NULL	NULL	NULL	NULL	NULL	NULL	\$2a\$10\$g3

Action Output

#	Time	Action	Message	Duration
1	21:05:25	SELECT * FROM airbnb11.admin LIMIT 0, 1000	3 row(s) returned	0.000 sec / 0.000 sec
2	21:05:28	SELECT * FROM airbnb11.billing LIMIT 0, 1000	0 row(s) returned	0.000 sec / 0.000 sec
3	21:05:31	SELECT * FROM airbnb11.host LIMIT 0, 1000	2 row(s) returned	0.000 sec / 0.000 sec
4	21:05:33	SELECT * FROM airbnb11.property LIMIT 0, 1000	2 row(s) returned	0.000 sec / 0.000 sec
5	21:05:35	SELECT * FROM airbnb11.trip LIMIT 0, 1000	5 row(s) returned	0.000 sec / 0.000 sec
6	21:05:37	SELECT * FROM airbnb11.user LIMIT 0, 1000	13 row(s) returned	0.000 sec / 0.000 sec

Object Info Session

Ask me anything

9:06 PM 12/4/2016

22. Add property:

AirBNB

Hi, Aishwarya! Lets get you ready to become a host.

What kind of place do you have?

Entire Place for 1 guest

Address: 201 S 4th Street

City: San Jose

State: CA

Zip code: 95112

Continue

Windows taskbar: Ask me anything, 9:59 PM, 12/4/2016

AirBNB

Give your place a title

The Colonnade Apartments

Examples from similar listings near you

The Cypress: Nimkish Forest R and R

Image here

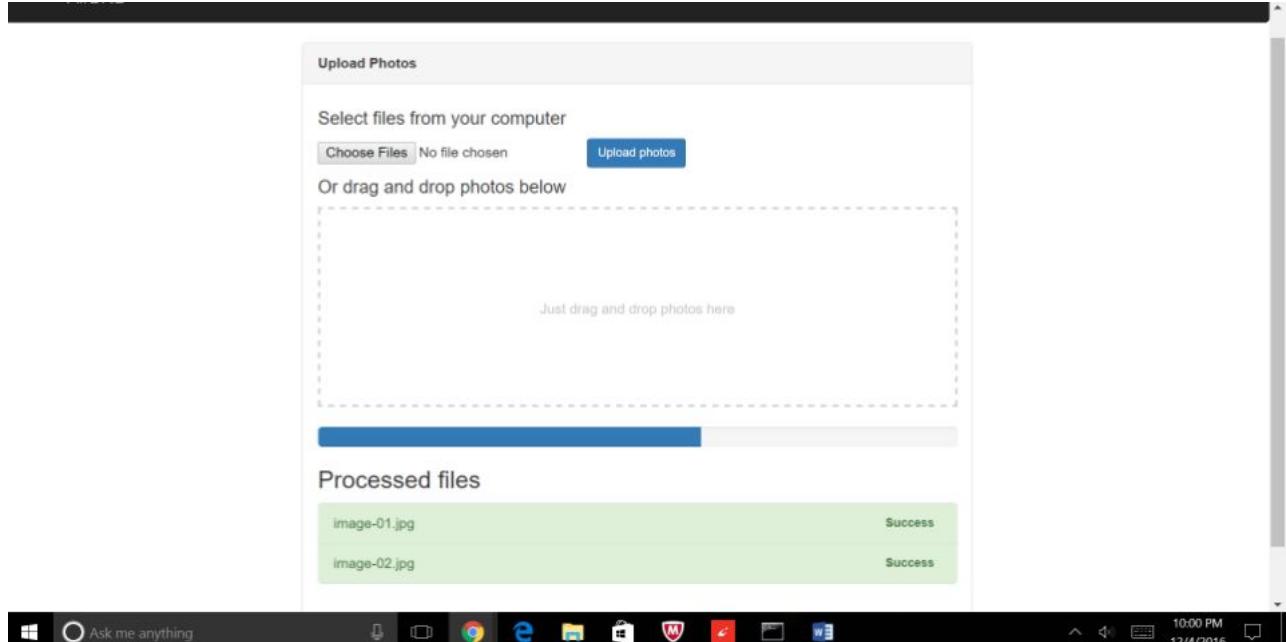
How travelers use titles

Travelers skim over the title when searching for places on Airbnb. Make titles catchy and short.

The best titles mention what's unique about your place—where it's close to, if it's big or cosy, if you have a special amenity—and what style it is—like "bright", "hipster", "family-style".

Back Next

Windows taskbar: Ask me anything, 9:59 PM, 12/4/2016



AirBNB

How do you want to set your price?

Price adapts to demand

You tell Smart Pricing to automatically adjust your price to match demand, but only within a price range that you set.

Price is fixed

Set a base price. Airbnb gives you price tips that you can accept or ignore.

Image here
The right price can change over time. Whatever you choose, you'll get personalized price tips.

Back Next



AirBNB

Set a price range
Your price will automatically adjust within the range you set.

Minimum
 Tip: \$33 CAD · Set

Maximum
 Tip: \$141 CAD · Set

Image here
How often you'd like to host
Your price can adjust to help you reach your occupancy goals. Adjusting your price doesn't change your availability settings or block dates on your calendar.

Back Next



AirBNB

Set a default nightly price
If you choose to turn off Smart Pricing, your nightly price will default to this amount.

Base Price
 Tip: \$47 CAD · Set

Currency

Image here
How often you'd like to host
Your price can adjust to help you reach your occupancy goals. Adjusting your price doesn't change your availability settings or block dates on your calendar.

Back Next



AirBNB

Update your calendar

Available from

Available Till

Season start date

Season end date

Guests will see your calendar and book available days.

Back
Submit



22. Log file:

event_log - Notepad

File Edit Format View Help

```
2016-10-24T20:34:21.124Z - info: User with id pb clicked at cart in file header
2016-10-24T20:34:21.375Z - info: User with id pb clicked at cart in file header
2016-10-24T20:34:25.571Z - info: User with id pb clicked at myebay in file header
2016-10-24T20:37:13.381Z - info: Guest user clicked at profile_link in file header
2016-10-24T20:37:14.983Z - info: Guest user clicked at register_tab in file login
2016-10-24T20:37:35.973Z - info: Guest user clicked at slideshow in file home
2016-10-24T20:40:11.679Z - info: Guest user clicked at profile_link in file header
2016-10-24T20:40:13.986Z - info: Guest user clicked at register_tab in file login
2016-10-24T20:40:15.680Z - info: Guest user clicked at login_tab in file login
2016-10-24T20:40:15.445Z - info: Guest user clicked at register_tab in file login
2016-10-24T20:41:05.532Z - info: Guest user clicked at register_tab in file login
2016-10-24T20:41:06.551Z - info: Guest user clicked at login_tab in file login
2016-10-24T20:41:07.414Z - info: Guest user clicked at register_tab in file login
2016-10-24T20:41:08.893Z - info: Guest user clicked at login_tab in file login
2016-10-24T20:41:09.558Z - info: Guest user clicked at register_tab in file login
2016-10-24T20:43:49.059Z - info: Guest user clicked at login_tab in file login
2016-10-24T20:43:51.241Z - info: Guest user clicked at sign_in in file login
2016-10-24T20:43:55.648Z - info: Guest user clicked at sign_in in file login
2016-10-24T20:44:00.332Z - info: Guest user clicked at profile_link in file header
2016-10-24T20:44:02.782Z - info: Guest user clicked at sign_in in file login
2016-10-24T20:44:23.421Z - info: Guest user clicked at profile_link in file header
2016-10-24T20:44:23.605Z - info: Guest user clicked at profile_link in file header
2016-10-24T20:44:24.350Z - info: Guest user clicked at profile_link in file header
2016-10-24T20:44:24.537Z - info: Guest user clicked at profile_link in file header
2016-10-24T20:44:25.768Z - info: Guest user clicked at profile_link in file header
2016-10-24T20:44:28.059Z - info: Guest user clicked at sign_in in file login
2016-10-24T20:44:34.731Z - info: Guest user clicked at sign_in in file login
2016-10-24T20:45:43.977Z - info: Guest user clicked at sign_in in file login
2016-10-24T20:46:45.267Z - info: Guest user clicked at sign_in in file login
2016-10-24T20:46:51.246Z - info: User with id pb clicked at profile_link in file header
2016-10-24T21:43:49.238Z - info: Guest user clicked at myebay in file header
2016-10-24T21:43:44.561Z - info: Guest user clicked at sign_in in file login
2016-10-24T21:43:47.716Z - info: User with id pb clicked at myebay in file header
2016-10-24T21:48:14.484Z - info: User with id pb clicked at myebay in file header
2016-10-24T22:06:57.592Z - info: User with id pb clicked at myebay in file header
2016-10-24T22:08:15.793Z - info: User with id pb clicked at myebay in file header
```

23. Testing: mocha.js

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

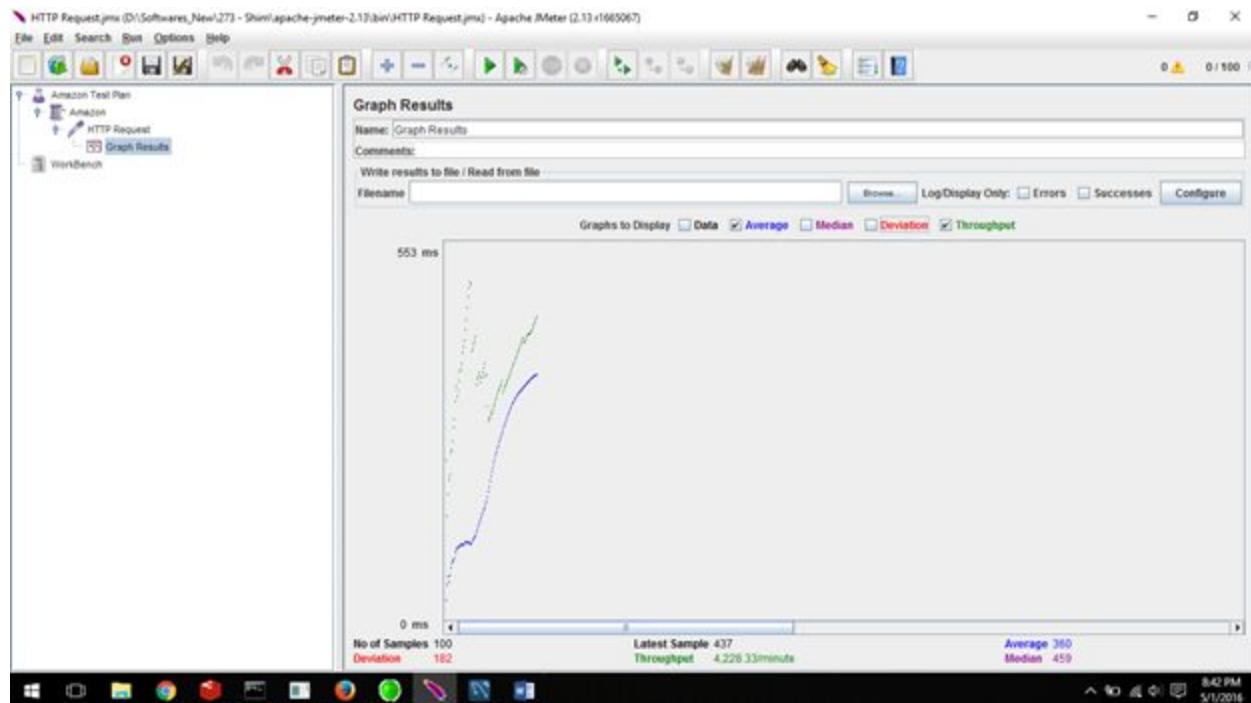
```

describe('http tests', function(){
  it('Test case for Login', function(done) {
    request.post( 'http://localhost:3000/login',{ form: {
      'username':'pandyabhavan134@gmail.com','password':'password' } },
      function (error, response, body) {
        assert.equal(200, response.statusCode);
        done();
      }
    );
  });
  it('Test case Get Header values for admin dashboard', function(done) {
    request.get('http://localhost:3000/getHeaderValues',function (error, response, body) {
      assert.equal(200, response.statusCode);
      done();
    }
  );
});
});

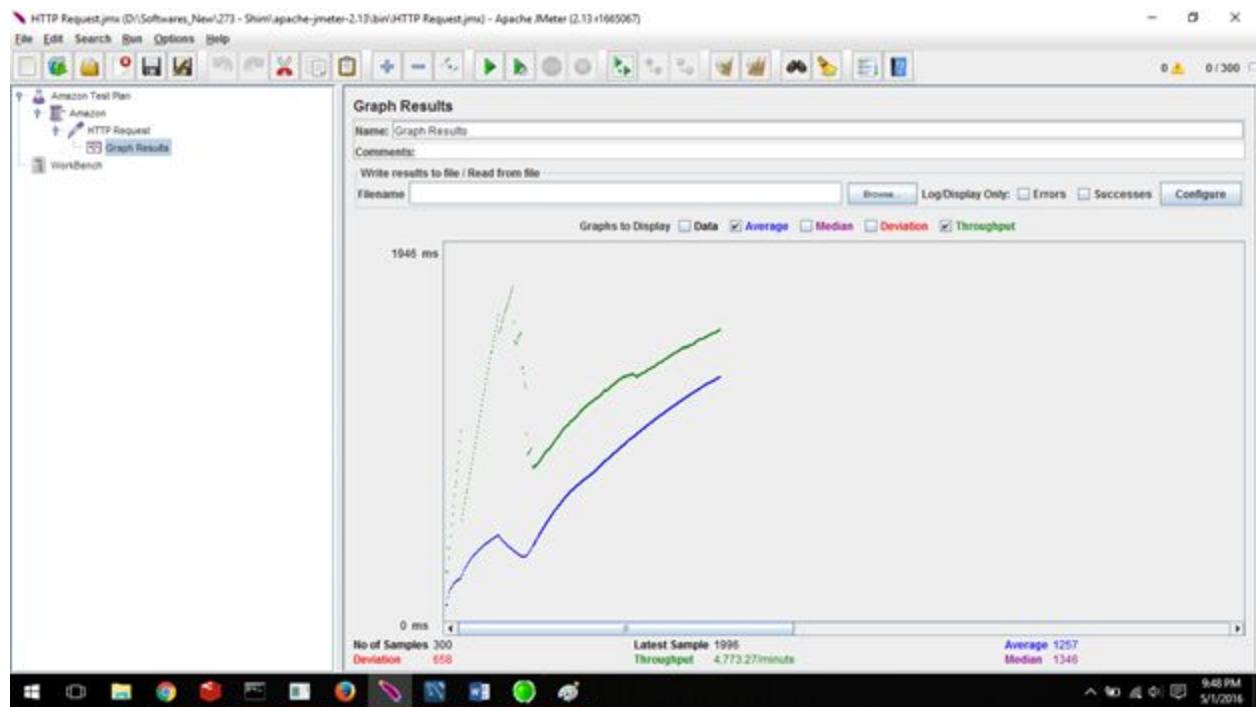
```

JMETER TESTING:

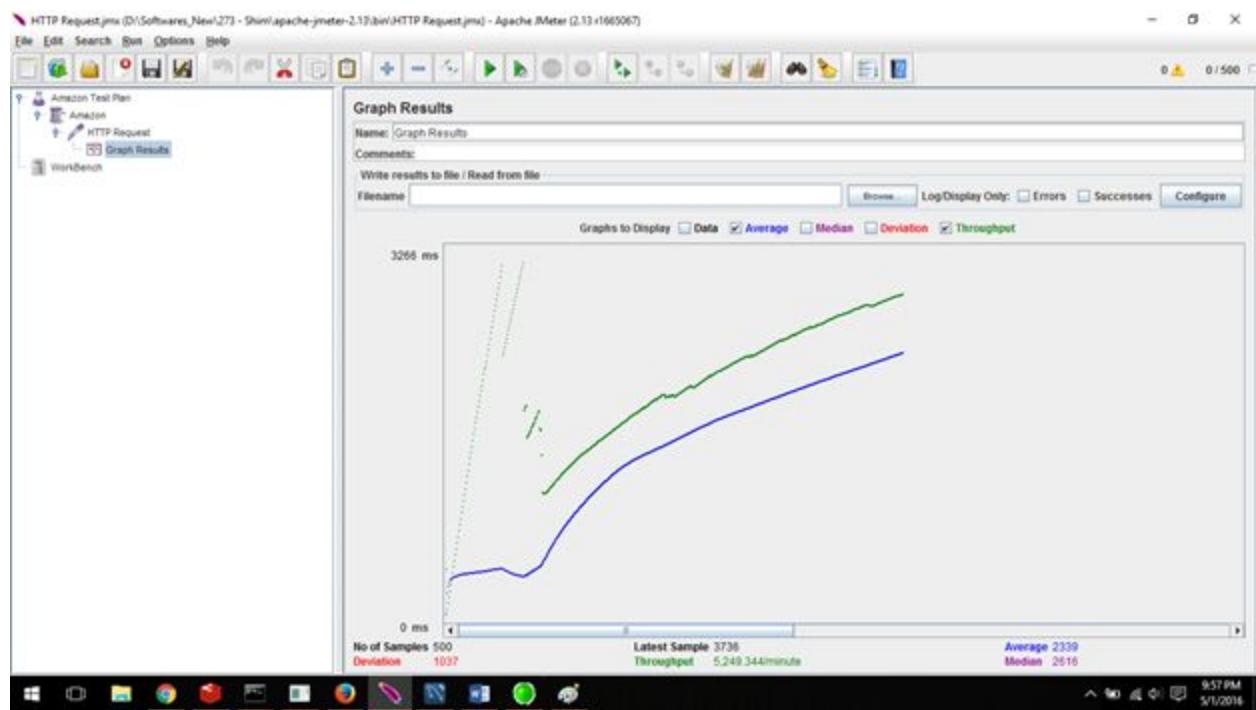
- 100 concurrent users:



- 300 concurrent users:



- 500 concurrent users:



CODE LISTING OF CLIENT APPLICATION:

- ejs file:

```
<!DOCTYPE html>
```

```

<html>
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1">
<title>Airbnb</title>
<link rel='stylesheet' href='/stylesheets/style.css' />
<link rel="stylesheet" href="../stylesheets/bootstrap/css/bootstrap.css">
<link rel="stylesheet" href="https://unpkg.com/purecss@0.6.0/build/pure-min.css">
<link rel="stylesheet"
      href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.6.3/css/font-awesome.min.css">
      <script type="text/javascript"
src="http://maps.googleapis.com/maps/api/js?libraries=places&sensor=false"></script>
      <script src="../javascripts/angular.js"></script>
      <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.1.1/jquery.min.js"></script>
      <script src="../stylesheets/bootstrap/js/bootstrap.js"></script>
      <script src="../angularjs/clientScript.js"></script>
      <script src="../angularjs/ngAutocomplete.js"></script>

      <!--//UI Router script-->
      <script
src="https://cdnjs.cloudflare.com/ajax/libs/angular-ui-router/0.3.2/angular-ui-router.js"></script>
</head>
<body ng-app="clientModule">
  <div ui-view="index"></div>
</body>
</html>

```

- html form snippet:

```

<div class="form-group">
  <div class="cols-sm-10">
    <div class="input-group">
      <input type="email" class="form-control" name="email" id="regemail" placeholder="Email
address" ng-model="user.email" required />
      <span class="input-group-addon"><i class="fa fa-envelope fa" aria-hidden="true"></i></span>
    </div>
  </div>
</div>
<div class="form-group">
  <div class="cols-sm-10">
    <div class="input-group">
      <input type="password" class="form-control" name="password" id="password" placeholder="New
Password" ng-model="user.password" required />
      <span class="input-group-addon"><i class="fa fa-lock fa-lg" aria-hidden="true"></i></span>
    </div>
  </div>
</div>

```

CONFIGURATION FILES:

- app.js

```
var express = require('express')
, routes = require('./routes')
, user = require('./routes/user')
, http = require('http')
, path = require('path')
, host = require('./routes/host')
, property = require('./routes/property');

var app = express();
var passport = require('passport');
require('./routes/passport')(passport);

var mysql=require('mysql');
var bcrypt = require('bcryptjs');

var expressSession = require("express-session");
var mongoStore = require("connect-mongo/es5")(expressSession);

var mongoURL="mongodb://localhost:27017/airbnbv10";
var login=require('./routes/login');
var room=require('./routes/room');
var profile=require('./routes/profile');
//Session configuration
app.use(expressSession({ secret: 'Airbnb',
  resave: false,
  //don't save session if unmodified // don't create session until something stored
  saveUninitialized: false,
  duration: 30 * 60 * 1000,
  activeDuration: 5 * 60 * 1000,
  store: new mongoStore({ url: mongoURL })
}));
//uncomment below middleware once passport startegy is defined
app.use(passport.initialize());
app.set('port', process.env.PORT || 3000);
app.set('views', __dirname + '/views');
app.set('view engine', 'ejs');
app.use(express.favicon());
app.use(express.logger('dev'));
app.use(express.bodyParser());
app.use(express.methodOverride());
app.use(app.router);
app.use(express.static(path.join(__dirname, 'public')));

// development only
```

```

if ('development' == app.get('env')) {
  app.use(express.errorHandler());
}

app.post('/loginUser', function(req, res, next) {
  passport.authenticate('login', function(err, user, info) {
    if(err) {
      response = {"statusCode":403,"data":null};
      res.send(JSON.stringify(response));
    }

    if(!user) {
      response = {"statusCode":401,"data":null};
      res.send(JSON.stringify(response));
    }

    req.login(user, {session:false}, function(err) {
      if(err) {
        response = {"statusCode":401,"data":null};
        res.send(JSON.stringify(response));
      }

      req.session.login = user;
      console.log("inside passport"+req.session.login);
      response = {"statusCode":200,"data":user};
      res.send(JSON.stringify(response));
    })
  })(req, res, next);
});

app.post('/getLoginSession',login.getLoginSession);

app.get('/loginUser', isAuthenticated, function(req, res) {
  res.render('successLogin', {user:{username: req.session.user}});
});

function isAuthenticated(req, res, next) {
  console.log('inside isauth');
  if(req.session.user) {
    console.log('in inside isauth');
    console.log(req.session.user);
    return next();
  }

  res.redirect('/');
};


```

```

// development only
app.post('/registerUser',login.register);

app.get('/',routes.index);
app.get('/users', user.list);
//calls to routes for airbnb

//app.post('/register',login.register);

//api to get all the room details
app.post('/getRoom',room.getDetails);
app.post('/getRoomd',room.getRoomDetails);

app.post('/gethostprofile',profile.getHostProfile);
app.post('/checkout',room.checkout);

app.post('/bookproperty',room.bookRoom);

app.post('/logout', function(req,res) {
  req.session.destroy();
  res.redirect('/');
});

var fs = require('fs');
app.post('/getPropertyListing',property.getPropertyListing);
app.post('/getBilling',property.getBilling);
app.post('/uploadimage', uploadimage);
app.post('/uploadvideo', uploadvideo);
//app.get('/rate_host', host.rate_host);
app.post('/review_host', host.review_host);
app.get('/reviewhost', host.reviewhost);
app.post('/getcurrentrating', host.getcurrentrating);
function uploadimage(req,res) {
  sleep(5000);
  var inStr = fs.createReadStream("/Users/mbars/Downloads/"+req.param("image"));
  var outStr =
    fs.createWriteStream("/airbnb/RabbitMQApplication/RabbitMQApplication/RabbitMQ-Client/public/i
mages/"+req.param("image"));

  inStr.pipe(outStr);
  property.addImage(req,res);
}

function uploadvideo(req,res) {
  sleep(5000);
  var inStr = fs.createReadStream("/Users/mbars/Downloads/"+req.param("image"));
  var outStr =

```

```

fs.createWriteStream("/airbnb/RabbitMQApplication/RabbitMQApplication/RabbitMQ-Client/public/images/" + req.param("image"));

inStr.pipe(outStr);
property.addVideo(req,res);

}

http.createServer(app).listen(app.get('port'), function(){
  console.log('Express server listening on port ' + app.get('port'));
});

● Server.js
//super simple rpc server example
var amqp = require('amqp')
, util = require('util');

var login = require('./services/login');

var property = require('./services/property');

var host = require('./services/host');
var cnn = amqp.createConnection({host:'127.0.0.1'});

cnn.on('ready', function(){
  console.log("listening on login_queue");

  cnn.queue('login_queue', function(q){

    q.subscribe(function(message, headers, deliveryInfo, m){
      util.log(util.format( deliveryInfo.routingKey, message));
      util.log("Message: " + JSON.stringify(message));
      util.log("DeliveryInfo: " + JSON.stringify(deliveryInfo));
      login.handle_request(message, function(err,res){

        //return index sent
        cnn.publish(m.replyTo, res, {
          contentType:'application/json',
          contentEncoding:'utf-8',
          correlationId:m.correlationId
        });
      });
    });
  });
}

cnn.queue('propertyListing_queue', function(q){
  q.subscribe(function(message, headers, deliveryInfo, m){

```

```

util.log(util.format( deliveryInfo.routingKey, message));
util.log("Message: "+JSON.stringify(message));
util.log("DeliveryInfo: "+JSON.stringify(deliveryInfo));
property.handle_request(message, function(err,res){

    //return index sent
    cnn.publish(m.replyTo, res, {
        contentType:'application/json',
        contentEncoding:'utf-8',
        correlationId:m.correlationId
    });
});
});
});
cnn.queue('billing_queue', function(q){
    q.subscribe(function(message, headers, deliveryInfo, m){
        util.log(util.format( deliveryInfo.routingKey, message));
        util.log("Message: "+JSON.stringify(message));
        util.log("DeliveryInfo: "+JSON.stringify(deliveryInfo));
        property.getBilling(message, function(err,res){

            //return index sent
            cnn.publish(m.replyTo, res, {
                contentType:'application/json',
                contentEncoding:'utf-8',
                correlationId:m.correlationId
            });
        });
    });
});
cnn.queue('approvem_bill_queue', function(q){
    q.subscribe(function(message, headers, deliveryInfo, m){
        util.log(util.format( deliveryInfo.routingKey, message));
        util.log("Message: "+JSON.stringify(message));
        util.log("DeliveryInfo: "+JSON.stringify(deliveryInfo));
        property.approveBill(message, function(err,res){

            //return index sent
            cnn.publish(m.replyTo, res, {
                contentType:'application/json',
                contentEncoding:'utf-8',
                correlationId:m.correlationId
            });
        });
    });
});
cnn.queue('addImage_queue', function(q){
    q.subscribe(function(message, headers, deliveryInfo, m){

```

```

util.log(util.format( deliveryInfo.routingKey, message));
util.log("Message: "+JSON.stringify(message));
util.log("DeliveryInfo: "+JSON.stringify(deliveryInfo));
property.addImage(message, function(err,res){

    //return index sent
    cnn.publish(m.replyTo, res, {
        contentType:'application/json',
        contentEncoding:'utf-8',
        correlationId:m.correlationId
    });
});
});
});
});

cnn.queue('addVideo_queue', function(q){
q.subscribe(function(message, headers, deliveryInfo, m){
    util.log(util.format( deliveryInfo.routingKey, message));
    util.log("Message: "+JSON.stringify(message));
    util.log("DeliveryInfo: "+JSON.stringify(deliveryInfo));
    property.addVideo(message, function(err,res){

        //return index sent
        cnn.publish(m.replyTo, res, {
            contentType:'application/json',
            contentEncoding:'utf-8',
            correlationId:m.correlationId
        });
    });
});
});
});

cnn.queue('gethostreviews_queue', function(q){
q.subscribe(function(message, headers, deliveryInfo, m){
    util.log(util.format( deliveryInfo.routingKey, message));
    util.log("Message: "+JSON.stringify(message));
    util.log("DeliveryInfo: "+JSON.stringify(deliveryInfo));
    host.getcurrentreviews(message, function(err,res){

        //return index sent
        cnn.publish(m.replyTo, res, {
            contentType:'application/json',
            contentEncoding:'utf-8',
            correlationId:m.correlationId
        });
    });
});
});
});
});

```

```

cnn.queue('review_host_queue', function(q){
    q.subscribe(function(message, headers, deliveryInfo, m){
        util.log(util.format( deliveryInfo.routingKey, message));
        util.log("Message: "+JSON.stringify(message));
        util.log("DeliveryInfo: "+JSON.stringify(deliveryInfo));
        host.review_host(message, function(err,res){

            //return index sent
            cnn.publish(m.replyTo, res, {
                contentType:'application/json',
                contentEncoding:'utf-8',
                correlationId:m.correlationId
            });
        });
    });
});
});
```

EXPRESS SESSION:

```

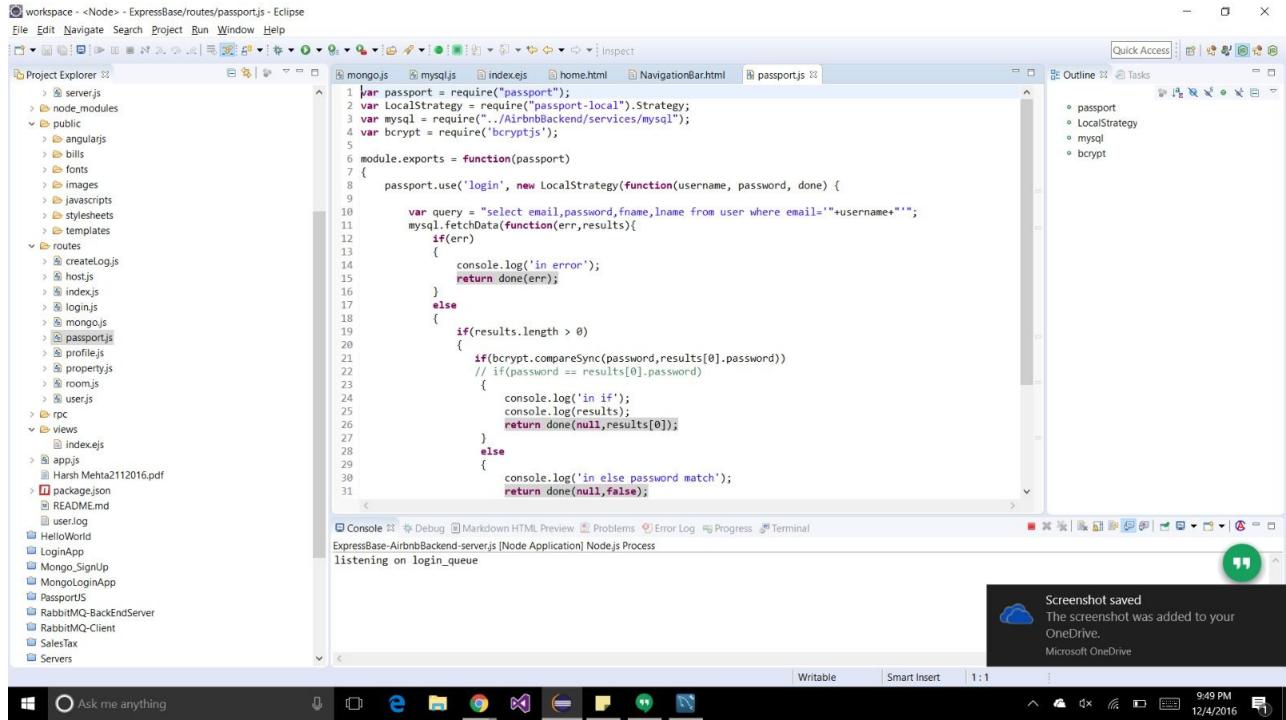
var mysql=require('mysql');
var bcrypt = require('bcryptjs');

var expressSession = require('express-session');
var mongoStore = require("connect-mongo/es5")(expressSession);
var mongoURL="mongodb://localhost:27017/airbnb10";
var login=require('./routes/login');
var room=require('./routes/room');
var profile=require('./routes/profile');

//Session configuration
app.use(expressSession({ secret: 'Airbnb',
    resave: false,
    //don't save session if unmodified // don't create session until something stored
    saveUninitialized: false,
    duration: 30 * 60 * 1000,
    activeDuration: 5 * 60 * 1000,
    store: new mongoStore({ url: mongoURL })
}));

//uncomment below middleware once passport strategy is defined
app.use(passport.initialize());
app.use(passport.session());
```

PASSPORT JS:



Optimisation Techniques for Web Application

1) Redis Caching:

Redis caching is implemented on the server side to reduce the overhead of opening and closing DB connections of the frequently visited property.

app.js

```
app.post('/getRoomId', function (req, res) {
  if (!req.param('data'))
    {res.status(400).send("Please send a proper title");
     console.log("in if of app redis details"+req);
    }
  else {
    room.getRoomDetailsCached(redis, req.param('data'), function (room) {
      if (!room) res.status(500).send("Server error");
      else res.status(200).send(room);
    });
  }
});
```

room.js

```

var getRoomDetails=function (req,callback)      //changes in exports after redis
{
  //console.log("In room details"+req.title);
  console.log("in room details"+req);
  var msg_payload={
    proname: req, /////changes in exports after redis
    queue:"room_details"
  }

  mq_client.make_request('login_queue',msg_payload,function(err,results){

    console.log(results);
    if(err){
      throw err;
    }
    else
    {
      if(results.statusCode === 200){
        console.log("valid Login");
        //req.session.fname=results.fname;
        res={"room":results.room, "statusCode":200, "image":results.image};
        //res.send({"room":results.room, "statusCode":200, "image":results.image});
        callback(null,res);

        console.log("success");
      }
      else {
        console.log("Invalid Login");
        //res.send({"login":"Fail"});
        res={"login":"Fail"};
        callback(null,res);
      }
    }
  });
};

exports.getRoomDetailsCached=function (redis, title, callback) {
  console.log("In redis cachd"+title);
  redis.get(title, function (err, reply) {
    if(err) callback(null);
    else if(reply) //room exists in cache
      callback(JSON.parse(reply));
    else {
      console.log("In redis cachd"+title);
      getRoomDetails(title,function (err,room) {
        if(err || !room)
        {
          callback(null);
        }
      });
    }
  });
}

```

```

        else {
            redis.set(title, JSON.stringify(room), function () {
                console.log("in set redis"+redis.get(title));
                callback(room);
            });
        }
    });
}
);
}
);

```

2) Connection Pooling:

Connection pooling is implemented for both the databases, MySQL and MongoDB.

MongoDB Pooling:

```

var MongoClient = require('mongodb').MongoClient;
var db;
var connected = false;
var ejs = require('ejs');//importing module ejs
var mysql = require('mysql');//importing module mysql
var pool = [], free = [], req = [];
var current_connection;
var mongoURL = "mongodb://localhost:27017/login";

/**
 * Connects to the MongoDB Database with the provided URL
 */
exports.connect = function(url, callback){
    var _db = p.get(url);

    db = _db;
    connected = true;
    console.log(connected +" is connected?");
    callback(db);
    p.release(db)
};

/**
 * Returns the collection on the selected database
 */
exports.collection = function(name){
//    var number = p.get(name);
    if(!connected) {
        throw new Error('Must connect to Mongo before calling "collection"');
    }
    var coll = db.collection(name);
    return coll;
};

```

```

function getConnection(pool){

    MongoClient.connect(mongoURL, function(err, _db){
        if(err) { throw new Error('Could not connect: '+err); }
        db = _db;
        connected = true;
        pool.push(db);
    });
}

function Pool()
{

    for(var i=0; i < 50; ++i)
    {
        getConnection(pool);
        free.push(i);
    }

}

Pool.prototype.get = function(request_name)
{
    if(free.length > 0)
    {
        var db = pool[free.length-1];
        free.pop();
        current_db = db;
        return current_db;
    }
    else
    {
        req.push(request_name);
        return null;
    }
}

Pool.prototype.release = function(number)
{
    free.push(number);
    if(req.length > 0)
    {
        Pool.get(req[0]);
        req.slice(0,1);
    }
}

```

```
var p=new Pool();
```

MySQL:

- MySQL connection pool:

```
var MongoClient = require('mongodb').MongoClient;
var dbConnection;
var connected = false;
var optionvalues = {
    db : {
        numberOfRetries : 5
    },
    server : {
        auto_reconnect : true,
        poolSize : 400,
        socketOptions : {
            connectTimeoutMS : 500
        }
    },
    replSet : {},
    mongos : {}
};

var dbConnection;
function initiatePool(url, callback) {
    MongoClient.connect(url, optionvalues, function(err, db) {
        if (err)
            throw err;

        dbConnection = db;
        connected = true;
        callback(dbConnection);
    });
}

exports.connect = function(url, callback) {
    if(!dbConnection){
        initiatePool(url, callback)
    }
    else{
        callback(dbConnection);
    }
};

exports.collection = function(name) {
    if (!connected) {
        throw new Error('Must connect to Mongo before calling "collection"');
    }
}
```

```

        return dbConnection.collection(name);
   };

● MySQL Connection Pooling
var mysql = require('mysql');
var listOfConnections = require("collections/list")
var connectionPool;

function getConnection(){
    var connection = mysql.createConnection({
        host : 'localhost',
        user : 'root',
        password : 'password',
        database : 'airbnb',
        port : 3306
    });
    return connection;
}

exports.createConnectionPool = function(noOfConnections){
    connectionPool = new listOfConnections();
    for(var i=0;i<noOfConnections;i++){
        connectionPool.push(getConnection());
    }
};

var getConnectionFromConnectionPool = function (){
    if(connectionPool.length == 0){
        return getConnection();
    }else{
        return connectionPool.pop();
    }
};

releaseConnection = function(connection){
    connectionPool.push(connection);
};

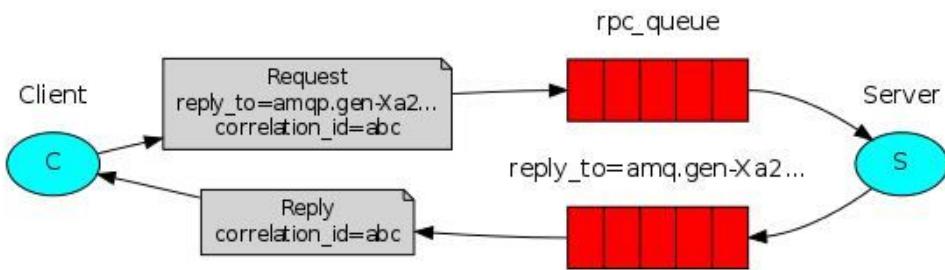
exports.fetchData = function(callback,sqlQuery){
    var connection=getConnectionFromConnectionPool();
    connection.query(sqlQuery, function(err, rows, fields) {
        if(err){
            console.log("ERROR: " + err.message);
        }
        callback(err, rows);
    });
    releaseConnection(connection);
};

```

3) RabbitMQ:

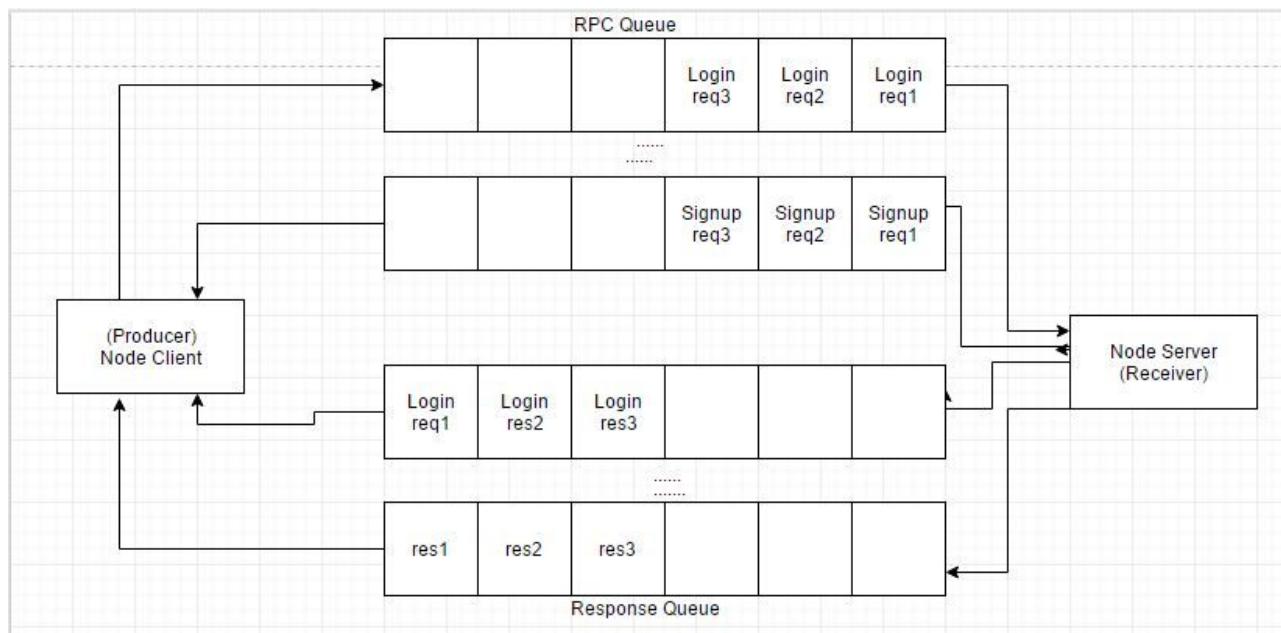
Decoupling of node backend using RPC Message Queues.

Summary



Our RPC will work like this:

- › When the Client starts up, it creates an anonymous exclusive callback queue.
- › For an RPC request, the Client sends a message with two properties: `reply_to`, which is set to the callback queue and `correlation_id`, which is set to a unique value for every request.
- › The request is sent to an `rpc_queue` queue.
- › The RPC worker (aka: server) is waiting for requests on that queue. When a request appears, it does the job and sends a message with the result back to the Client, using the queue from the `reply_to` field.
- › The client waits for data on the callback queue. When a message appears, it checks the `correlation_id` property. If it matches the value from the request it returns the response to the application.



4) Prepared Statement:

This technique is used to prevent the web application from sql injection.

```
/*/var clean_user = sanitizer.sanitize(username);  
  
var post = {Username: clean_user, Password: hash};  
  
var query = connection.query('INSERT INTO users SET ?', post, function(err, results)  
{ // Can a Sql injection happen here? });*/
```