

SOFTWARE ENGINEERING

FALL 2016



SAN JOSÉ STATE UNIVERSITY

CMPE 273 – Enterprise Distributed Systems – Team Project

Airbnb Simulation with Auction – Report

Team Number 17

Team Member	SJSU ID
Maitray Shah	011432279
Parth Mukesh Upadhyay	011451219
Salmaan Pehlari	011409307
Shrey Bhatt	011489777
Vedant Shete	011412141

Contribution of Members

- **Maitray Shah:** Back End Development of Analytics Module, Mocha Testing, Front End Design of Analytics and Host Module
- **Parth Upadhyay:** Backend Development of Trips, Property Listing Module, Redis Cache Management, RabbitMQ Implementation, Integration of Back End and Front End, Google Maps API
- **Salmaan Pehlari:** Backend Development of Billing, Host, User Module, Dynamic Pricing Algorithm implementation, Mocha Testing, Integration of Back End and Front End, Redis Cache Management
- **Shrey Bhatt:** Database Design, Front End Design of Admin, User, Property, Trip and Profile Module, RabbitMQ implementation, Integration of Back End and Front End, JMeter Testing
- **Vedant Shete:** Backend Development of Admin Panel Module, Visualization of Analytics Graphs Using D3, JMeter Testing

Introduction

- **Goal**

The application is the prototype of the Airbnb. In this project we built Airbnb which enables people to list, find, then rent vacation homes.

Airbnb is a peer-to-peer accommodation market place that connects hosts (vendors of rooms/accommodations) and travelers via its website.

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.

Validation at every stage of the application is also implemented, and all exceptions have also been handled.

Server is also tested with several hundred concurrent users using JMeter, and the results have been compared and compiled in a graph to compare the effect of adding connection pooling in the system.

Testing of the Rest API's has also been done using Mocha

- **Purpose of System**

We learned and implemented the MEAN stack for web development and also the use of RabbitMQ using the AMQP protocol for scalability. We have used the Redis for cache management.

System Design

Technologies used

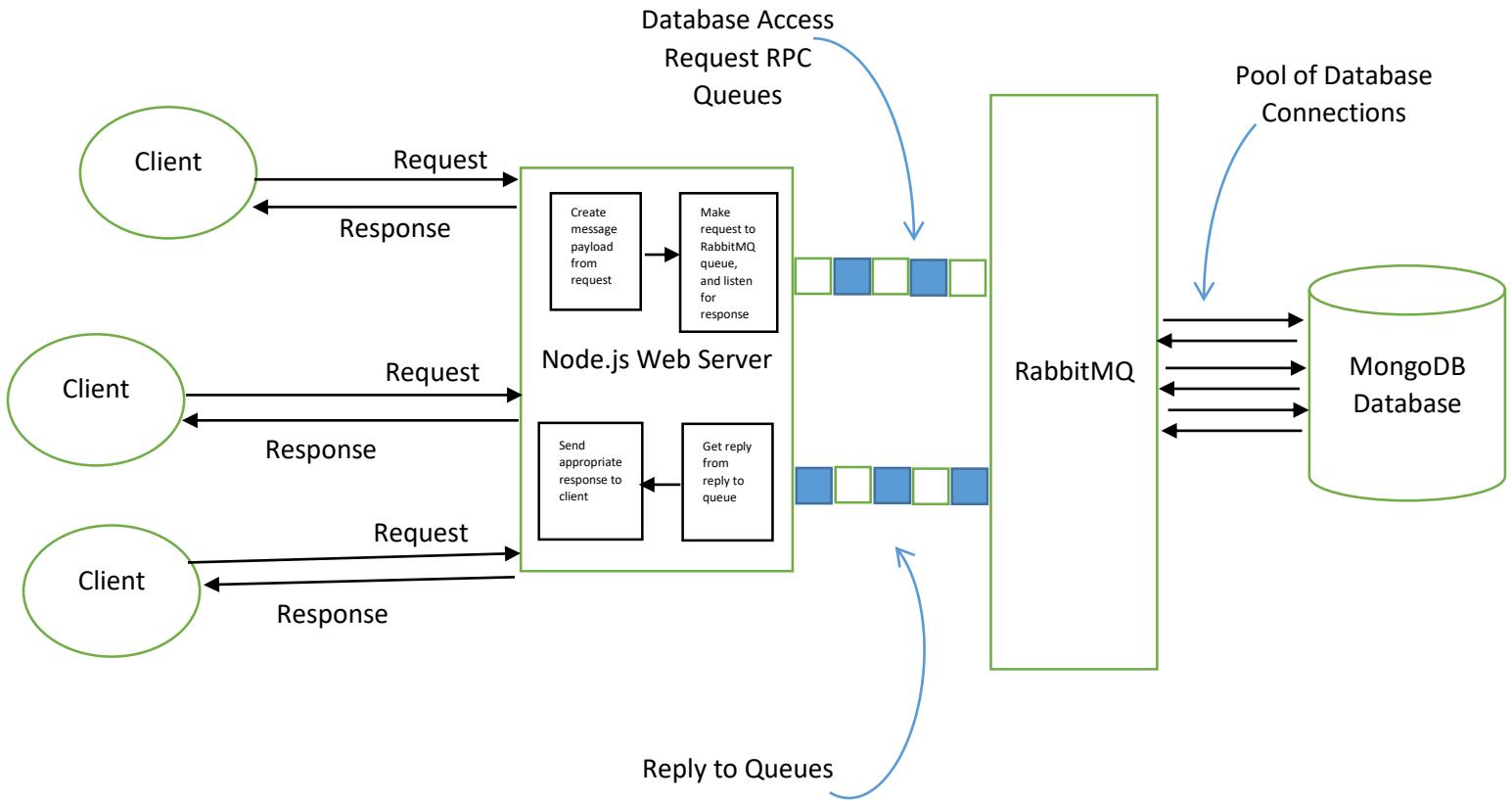
- Frontend : Angular.js
- Backend : Node.js
- UI/UX : CSS and Bootstrap
- Markup Language : Html

- Scripting Language : JavaScript
- Database : MongoDB
- Password Encryption : Bcrypt
- Authentication : Passport.js
- Messaging Protocol : AMQP
- Message broker Middleware : RabbitMQ
- SQL Caching : Redis
- Analytics : ELK stack
- Data Generation : Python , Faker

- The system is made using NodeJS, and express JS framework at the backend.
- MongoDB is used as a database for the application.
- Sessions have also been stored in MongoDB.
- RabbitMQ is used as a message passing mechanism for database access from client to server.
- Frameworks like bcrypt for password hashing, and Winston for log generation have also been used. For front-end, angularJS is used along with HTML/CSS.
- The user is prompted to login, or sign up on the front page. The user cannot access any pages without signing in. The user's password is encrypted and stored in the database. The user' log in time is also logged in the database.
- User's personal information is also stored in the database.
- User can then proceed to post and advertisement or buy from other user advertisements.
- User can also bid on other items, which takes place over a period of 4 days.
- Connection pooling has been implemented at the backend for the server to handle requests and send responses faster.
- The server tracks all activities of the user like which item seen, item added to cart, time of log in, etc.

- Server also takes care of validation at every stage. Incorrect login, negative values for stock, and item quantities have been taken care of as well.

ARCHITECTURE DIAGRAM



Object Management Policy

1. Requirement gathering and Requirement Analysis:

In this phase, we performed requirement gathering and requirement analysis. As per the requirements of the project the three tiers needed in order to accomplish the Airbnb application are as following:

I) Client tier: The user will interact with the Airbnb application using the User Interface.

II) Middle tier/ Message Oriented Middleware(MOM) : This tier supports sending and receiving messages between distributed systems. The services have been implemented using RabbitMQ message broker which implements the Advanced Message Queuing Protocol(AMQP).

III) Database tier: This tier comprises of the database to store the objects of the system. We are using two databases for the implementation of this application.

- i) MySQL : We are storing the data in MySQL if the application has high transactional data which requires ACID properties.
- ii) MongoDB: In MongoDB we are storing the data which requires flexible schema management and the size of the data is comparatively large for the processing.

2.Database Design:

In the Airbnb application; we implemented different modules namely User, Host, Admin, Analysis, Listing, Billing and Trips. All of the modules has an associated schema which is responsible for representing how particular data should be stored in database.

3.User Interface Design and Backend Development:

On the basis of Airbnb application requirements, we have developed the User Interface and implemented different functionalities. The implementation of the maps has been done using google maps API. We have used RabbitMQ to increase the reliability when the number of requests are more. We also have used Redis for SQL cache management.

4.Testing:

We have frequently tested all the APIs of the Airbnb application using Jmeter and Mocha and recorded results for the same. We have made changes accordingly if the performance was not up to the mark. We have designed Mocha test cases for each APIs.

How we handled the “Heavy Weight” resources.

The heavy weight resources in the project are images and videos of properties, photos of users and photos uploaded by user during property and trip review. We have all these heavy weight resources inside MongoDB as reading the same from the MySQL will take much more lookup time. As we have stored the images and videos in the MongoDB, retrieving them will be much more faster compared to MySQL. We also implemented redis cache to make sure the same object isn't processed many times by the server.

The policy we used to decide when to write the data into the database

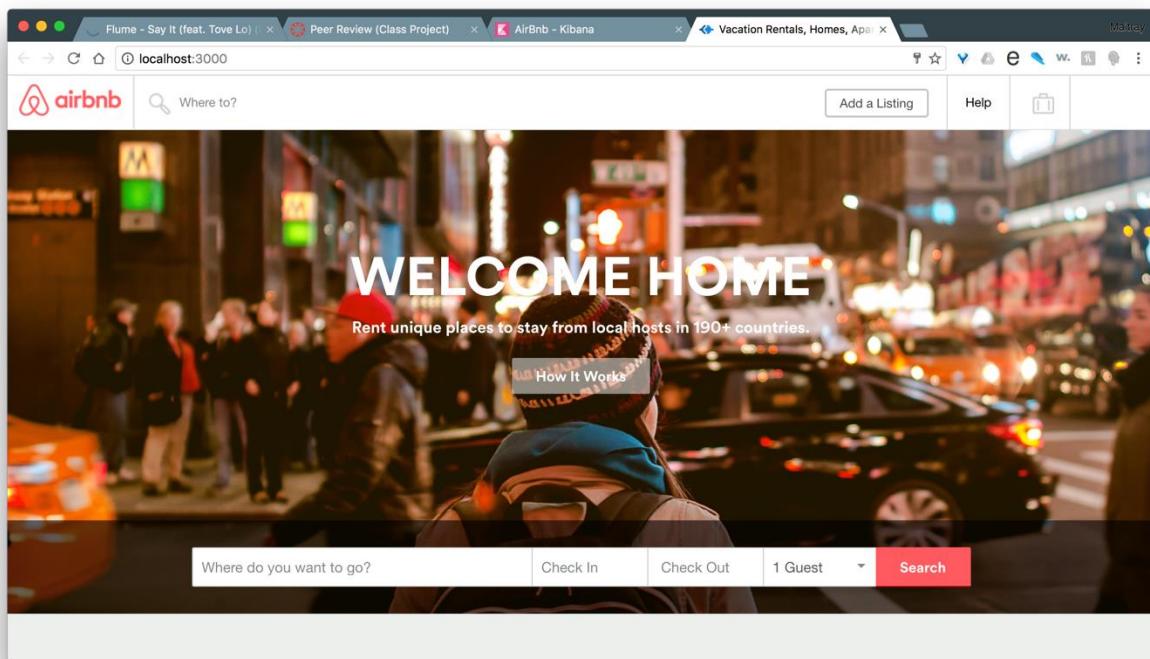
We had decided to insert the data into database optimally. For example; in the scenario like Adding a listing, we are using Mongo session to store the data provided by the Host in multiple screens and we are storing this data into the database only after all the data entry operations of all the screens get completed. Example: Add listing module has 6 screens. If we save the data in to Database in every screen then if the user terminates data entry operation in any of the next screens, then the previous entry operations need to be rolled back.

Therefore, we are storing this kind of data in session in each screen and saving the data only once in the last screen. This will reduce multiple inserts that needs to be done in previous screens and any further updates in the following screens.

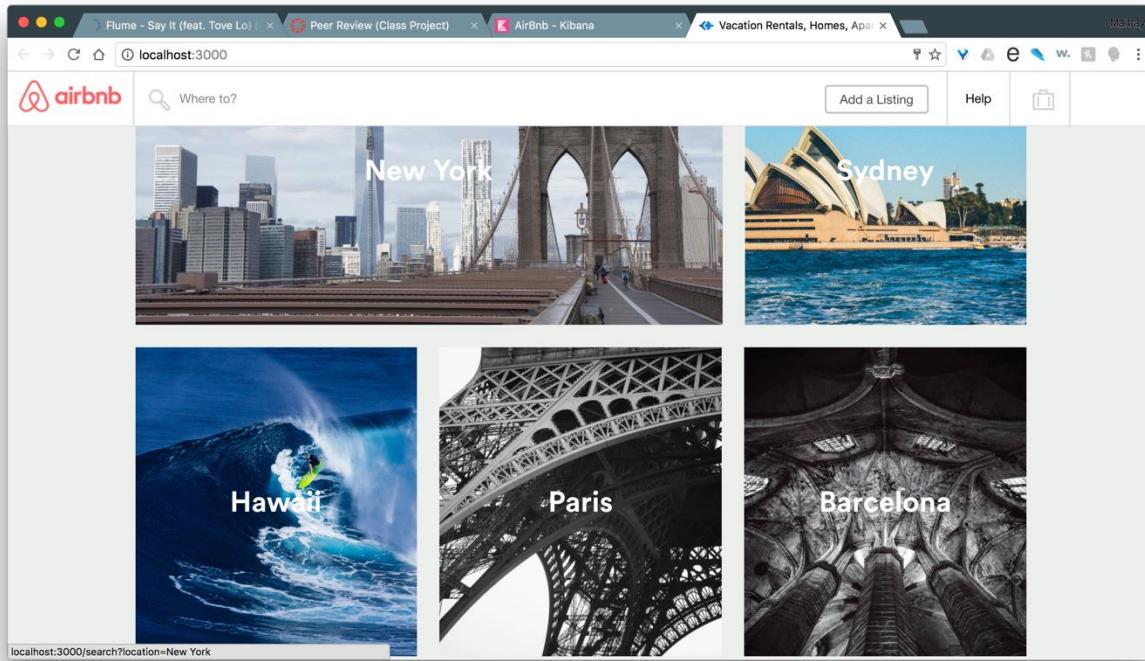
Modules

- **Homepage**

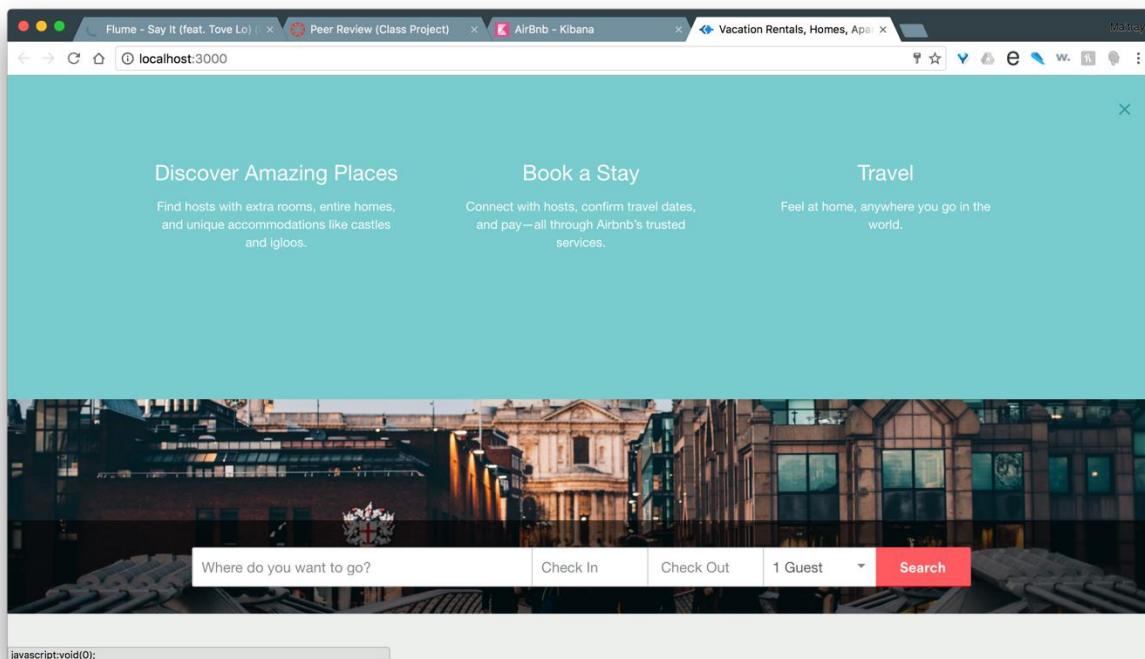
Home Page of the application. Shows all links for navigation and search for cities. Also shows different cities to navigate and search properties.



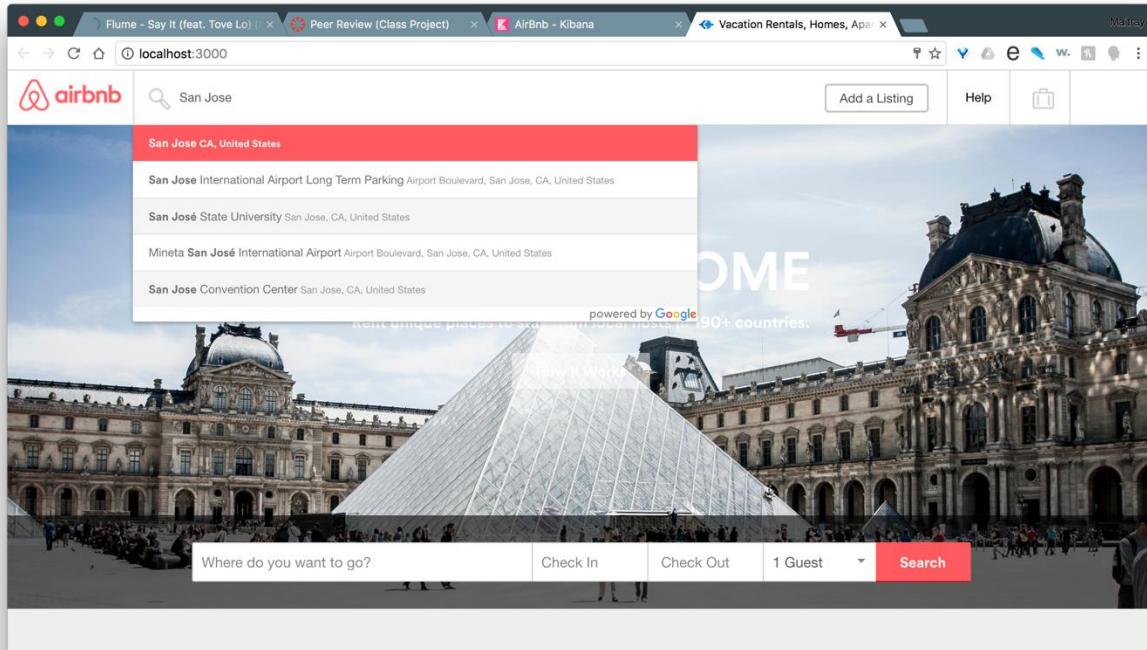
If you scroll down then you will see the Cities in the homepage



On clicking the How In Works button, This window will popup

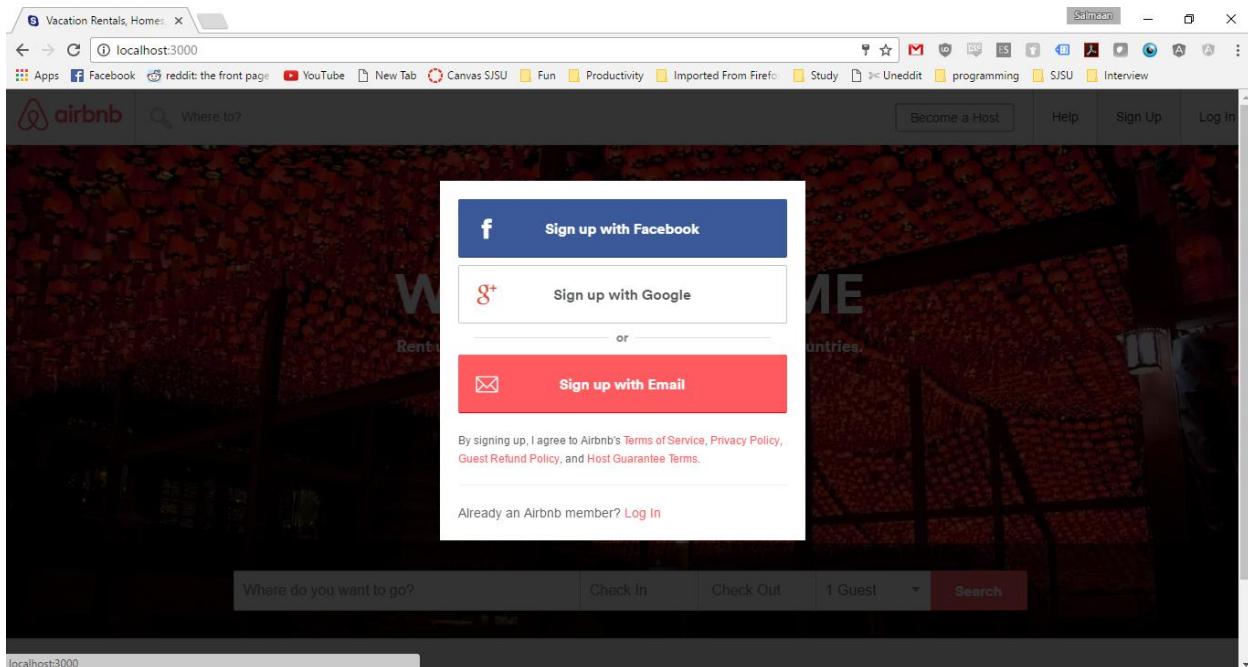


Search Navbar in the homepage



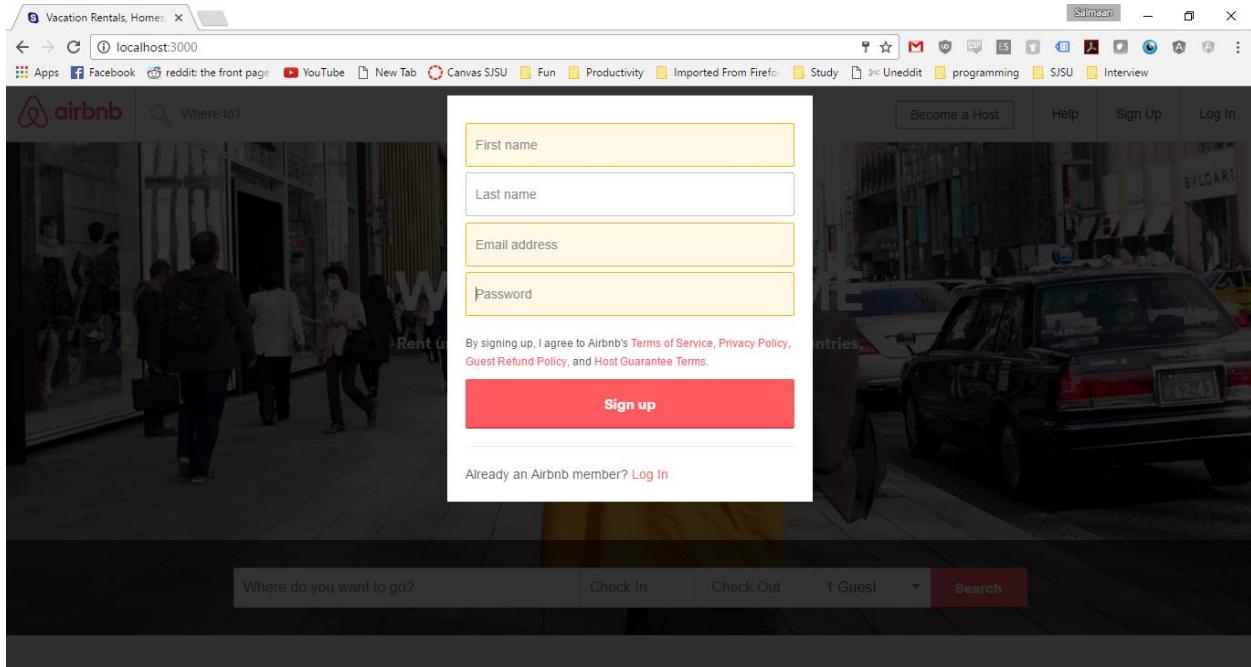
Registration Page

You can SignUp using email using the opened modal.

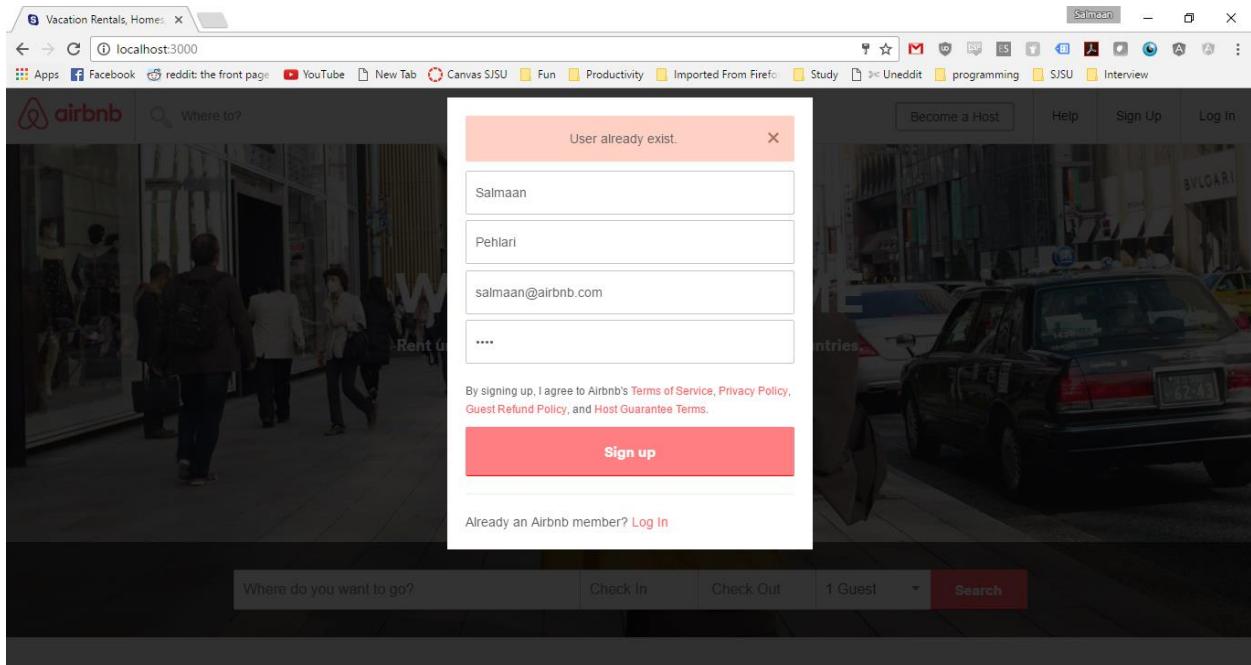


Signup with Email

If you click on sign up with Email then the following screen will popup.



Validation in Sign Up if User Already exists



SERVER CONSOLE AND CLIENT CONSOLE SCREENSHOTS

1. Sign In Call

A screenshot of a terminal window titled "cronBid.js - Airbnb_Client - [~/Desktop/Airbnb17-new/Airbnb_Client]". The terminal shows the following output:

```
in signin
response { _id: '5844d6803f3a625324b6593b',
  isActivated: true,
  isHost: true,
  createdDate: 1480906368829,
  userId: '681-22-2511',
  password: 's2a$109UL.3rqhx30Wbg5bMLVCS1.TH8reV8U7pqrvvCHofXdwEajeQZ07ce',
  email: 'salmaan@airbnb.com',
  lastName: 'Pehlari',
  firstName: 'Salmaan',
  __v: 0,
  isApproved: false,
  address: 'San Jose State University',
  city: 'San Jose',
  state: 'CA',
  zip: '95112',
  profileImage: '5844d6803f3a625324b6593b.png' }
{ _id: '5844d6803f3a625324b6593b',
  isActivated: true,
  isHost: true,
  createdDate: 1480906368829,
  userId: '681-22-2511',
  password: 's2a$109UL.3rqhx30Wbg5bMLVCS1.TH8reV8U7pqrvvCHofXdwEajeQZ07ce',
  email: 'salmaan@airbnb.com',
  lastName: 'Pehlari',
  firstName: 'Salmaan',
  __v: 0,
  isApproved: false,
  address: 'San Jose State University',
  city: 'San Jose',
  state: 'CA',
  zip: '95112',
  profileImage: '5844d6803f3a625324b6593b.png' }
RestifyObject Object
Info: Salmaan loaded in user=Salmaan, url_clicked=/
POST /signin 200 408 396 ms - 38
Info: Salmaan clicked on home user=Salmaan, url_clicked=/
GET / 200 14.977 ms - 72874
GET /css/common.css 304 92.141 ms - -
GET /css/home.css 304 96.371 ms - -
GET /css/themes.css 304 94.938 ms - -
```

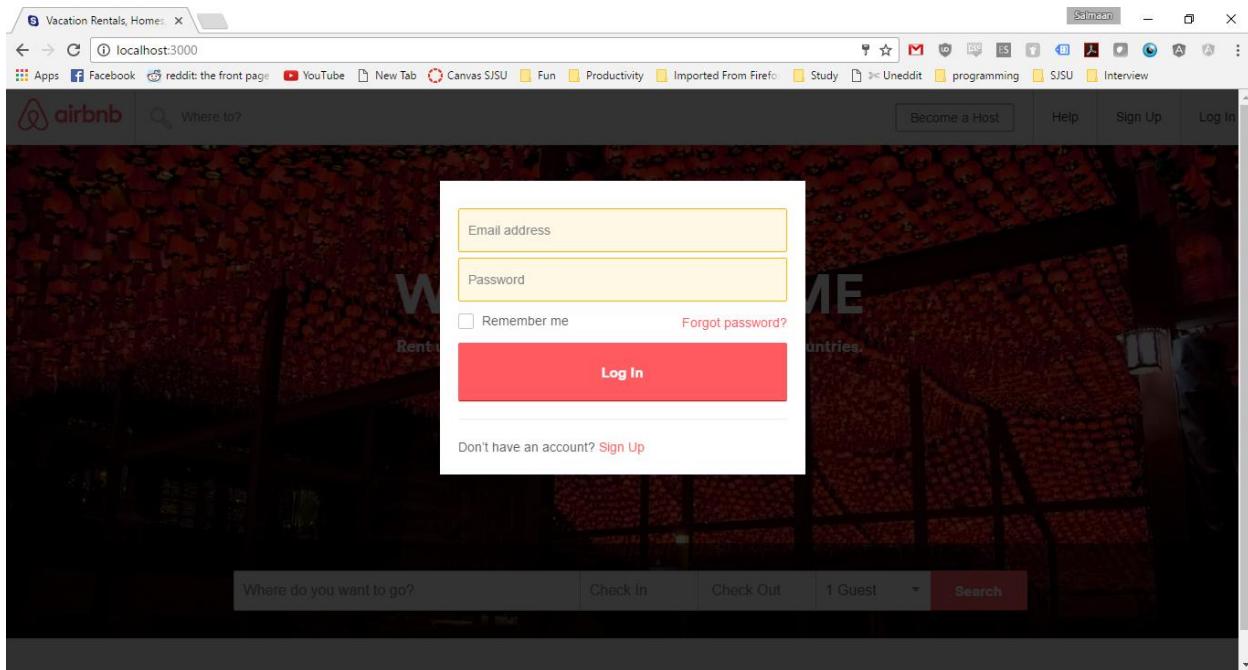
The terminal also shows the command "Process finished with exit code 130 (interrupted by signal 2: SIGINT)".

A screenshot of a terminal window titled "server.js - Airbnb_server - [~/Desktop/Airbnb17-new/Airbnb_server]". The terminal shows the following output:

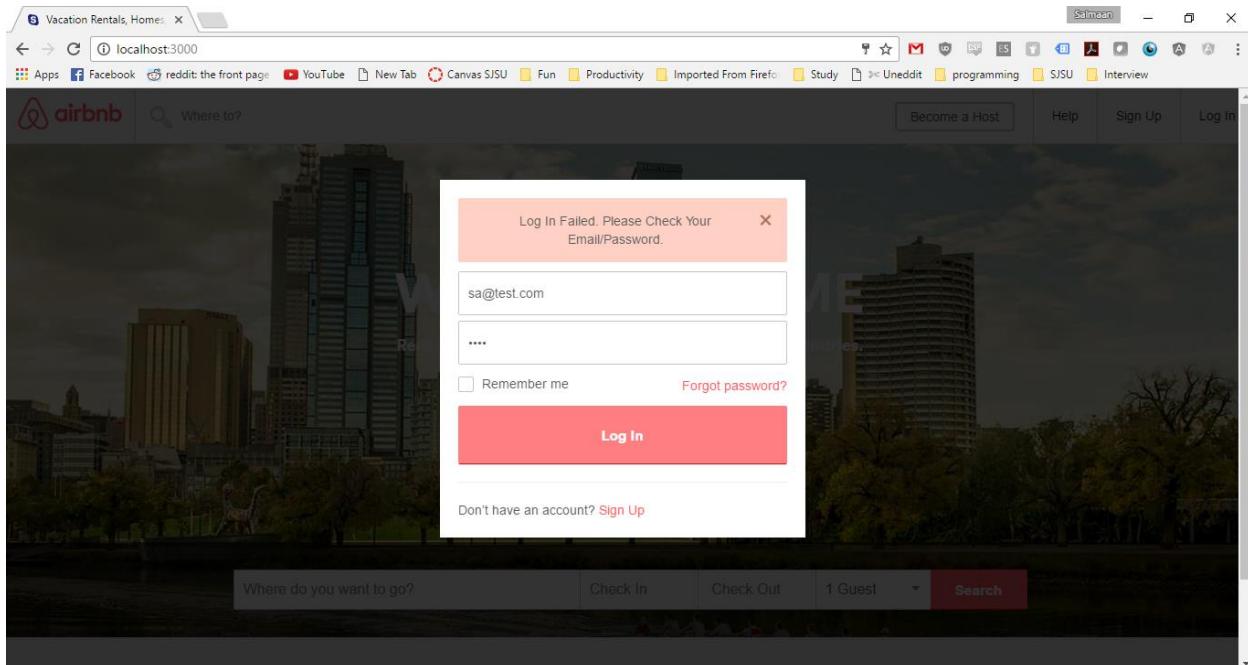
```
4 Dec 21:22:09 - login_queue { username: 'salmaan@airbnb.com', password: 'test' }
4 Dec 21:22:09 - Message: { "username": "salmaan@airbnb.com", "password": "test" }
4 Dec 21:22:09 - DeliveryInfo: { "contentType": "application/json", "contentEncoding": "utf-8", "correlationId": "9110b1275339fed453937f038bff0395", "replyTo": "amq.gen-Ltc_dW2m
USERNAME: salmaan@airbnb.com PASSWORD: test
{ _id: '5844d6803f3a625324b6593b',
  isActivated: true,
  isHost: true,
  createdDate: 1480906368829,
  userId: '681-22-2511',
  password: 's2a$109UL.3rqhx30Wbg5bMLVCS1.TH8reV8U7pqrvvCHofXdwEajeQZ07ce',
  email: 'salmaan@airbnb.com',
  lastName: 'Pehlari',
  firstName: 'Salmaan',
  __v: 0,
  isApproved: false,
  address: 'San Jose State University',
  city: 'San Jose',
  state: 'CA',
  zip: '95112',
  profileImage: '5844d6803f3a625324b6593b.png' }
```

The terminal also shows the message "Unregistered VCS root detected: The directory /Users/maltrayshah/Desktop/Airbnb17-new is under Git, but is not registered in the Settings. // Add root Configure Ignore (22 minutes ago)".

Login Page Screenshots



Validation in Login Page



On clicking the Add a Listing button, the following page will appear

The screenshot shows the 'Become a Host' page on the Airbnb website. At the top, there's a search bar with 'Where to?' and a 'Add a Listing' button. Below the header, the title 'Become a Host' is displayed, followed by the subtext 'Airbnb lets you make money renting out your place.' The main form consists of three sections: 'Home Type' (with options for Apartment, House, Bed & Breakfast, and Other), 'Room Type' (with options for Entire home/apt, Private room, Shared room, and Other), and 'Accommodates' (set to 1). The background features a light gray gradient.

This screenshot shows the continuation of the 'Become a Host' form. It adds 'City' and 'Continue' buttons below the previous sections. At the bottom, there are three footer features: 'Secure Payments' (lock icon), 'Trust & Safety' (handshake icon), and 'Host Guarantee' (host icon with a '8' inside). Each feature has a brief description: Secure Payments says 'Our fast, flexible payment system puts', Trust & Safety says 'Trust & safety tools help you accept a', and Host Guarantee says 'Your peace of mind is priceless. So we don't'.

Example of entering the details

The screenshot shows a web browser window with the URL localhost:3000/becomeHost. The page displays several input fields with accompanying descriptions:

- Home Type:** A dropdown menu showing "Apartment" selected, with a note: "Airbnb guests love the variety of home types available."
- Room Type:** A dropdown menu showing "Entire home/apt" selected, with a note: "Room type is one of the most important criteria for Airbnb guests."
- Accommodates:** A dropdown menu showing "11" selected, with a note: "Whether you're hosting a lone traveler or a large group, it's important for your guests to feel comfortable."
- City:** A dropdown menu showing "San Jose, CA, United States" selected, with a note: "What a great place to call home"

A red "Continue" button is centered at the bottom of the form.

Basic Tab in Adding a Listing

The screenshot shows the Airbnb "Edit Listing" page with the URL localhost:3000/addListing. The "Basic" tab is selected on the left sidebar. The main content area displays the following information:

Help Travelers Find the Right Fit
People searching on Airbnb can filter by listing basics to find a space that matches their needs.

Rooms and Beds

Bedrooms	Beds	Bathrooms
<input type="text" value="2"/>	<input type="text" value="2"/>	<input type="text" value="2"/>

A red "Next" button is located at the bottom right of the form.

The footer includes links to Terms of Service, Privacy Policy, Host Guarantee, Guest Refund, Copyright Policy, Consent Disagree, and About Us. It also features a language selection dropdown set to English.

SERVER AND CLIENT CONSOLE Description tab in Adding a Listing

Project: Airbnb_Client | routes | cronBid.js

```

in signin
  response { _id: '5844d6803f3a625324b6593b',
  isActivated: true,
  isHost: true,
  createdDate: 1480906368829,
  userId: '681-22-2511',
  password: '52a$105UL3rqhx30Wbg5bMLVCS1.TH8reV8U7pqrvvCHofXdwEajeQZ07ce',
  email: 'salmaan@airbnb.com',
  lastName: 'Pehlari',
  firstName: 'Salmaan',
  __v: 0,
  isApproved: false,
  address: 'San Jose State University',
  city: 'San Jose',
  state: 'CA',
  zip: '95112',
  profileImage: '5844d6803f3a625324b6593b.png' }
{ _id: '5844d6803f3a625324b6593b',
  isActivated: true,
  isHost: true,
  createdDate: 1480906368829,
  userId: '681-22-2511',
  password: '52a$105UL3rqhx30Wbg5bMLVCS1.TH8reV8U7pqrvvCHofXdwEajeQZ07ce',
  email: 'salmaan@airbnb.com',
  lastName: 'Pehlari',
  firstName: 'Salmaan',
  __v: 0,
  isApproved: false,
  address: 'San Jose State University',
  city: 'San Jose',
  state: 'CA',
  zip: '95112',
  profileImage: '5844d6803f3a625324b6593b.png' }
ReactiveObject Object
info: Salmaan clicked in user=Salmaan, url_clicked=/sign
POST /sign 200 408 395 ms - 38
info: Salmaan clicked on home user=Salmaan, url_clicked=
GET / 200 14.977 ms - 72874
GET /css/common.css 304 92.141 ms - -
GET /css/home.css 304 96.371 ms - -
GET /css/themes.css 304 94.938 ms - -
4: Run TODO Terminal Event Log
Process finished with exit code 130 (interrupted by signal 2: SIGINT)
84:1 LF: UTF-8: 8

```

Project: Airbnb_Client | routes | dynamicPricingCron.js

```

info: Salmaan clicked on becomeHost user=Salmaan, url_clicked=/becomeHost
GET /becomeHost 200 12.330 ms - 75613
GET /css/common.css 304 90.853 ms - -
GET /css/home.css 200 1.804 ms - 5833
GET /css/themes.css 304 86.430 ms - -
GET /css/pcss.css 304 87.168 ms - -
GET /js/common.js 304 87.591 ms - -
GET /controller/common_controller.js 304 87.360 ms - -
GET /images/logos/logo.png 304 86.885 ms - -
GET /images/search-icon.png 304 90.276 ms - -
GET /fonts/airlymphs-5ebc51824ab0c88d579d01078cff346a.woff 304 90.090 ms - -
GET /fonts/circular_Air-Bold-ba3e38967877fa1729525589ca6f5.woff 304 86.964 ms - -
GET /images/trip-icon.png 304 86.925 ms - -
GET /images/user/5844d6803f3a625324b6593b.png 404 11.520 ms - 109270
GET /getSession 200 0.998 ms - 299
GET /renderFooter 404 13.220 ms - 109270
POST /becomeHost 200 2085.822 ms - 109270
info: Salmaan clicked on addlisting user=Salmaan, url_clicked=/addListing
GET /addlisting 200 17.201 ms - 111828
GET /css/common.css 304 92.089 ms - -
GET /css/home.css 304 98.503 ms - -
GET /css/themes.css 304 87.645 ms - -
GET /css/pcss.css 304 91.323 ms - -
GET /css/manage_listing.css 304 87.298 ms - -
info: Salmaan clicked on addlisting user=Salmaan, url_clicked=/addListing
GET /images/lightbulbx.png 404 24.710 ms - 109270
GET /fonts/circular_Air-Bold-ba3e38967877fa1729525589ca6f5.woff 304 86.964 ms - -
GET /addlisting 304 16.040 ms - -
GET /common.js 304 83.886 ms - -
GET /controller/common_controller.js 304 86.722 ms - -
GET /images/logos/logo.png 304 89.223 ms - -
GET /images/search-icon.png 304 85.784 ms - -
GET /fonts/airlymphs-5ebc51824ab0c88d579d01078cff346a.woff 304 97.398 ms - -
GET /fonts/circular_Air-Bold-ba3e38967877fa1729525589ca6f5.woff 304 85.467 ms - -
GET /images/trip-icon.png 304 87.064 ms - -
GET /images/user/5844d6803f3a625324b6593b.png 404 17.294 ms - 109270
GET /css/images/background-7c730785fe74bc03505fc1f77672fd.png 404 7.054 ms - 109270
GET /images/footer_cover_image.png 304 88.552 ms - -
Error: Error: Redis connection to 127.0.0.1:6379 failed - connect ECONNREFUSED 127.0.0.1:6379
Mon, 05 Dec 2016 05:36:52 GMT express deprecated req.param(name): Use req.params, req.body, or req.query instead at ../routes/listings.js:131:25
4: Run TODO Terminal Event Log
Unregistered VCS root detected: The directory /Users/mairtayshah/Desktop/Airbnb17-new is under Git, but is not registered in the Settings. // Add root Configure Ignore (19 minutes ago)
340:1 LF: UTF-8: 8

```

Location Tab in Adding a Listing

The screenshot shows the Airbnb listing creation interface. On the left, a sidebar lists categories: Listing, Basics, Description, Location, Photos, Hosting, and Pricing. The 'Location' category is selected and expanded, showing fields for Country (United States), Address Line 1 (San Jose), Address Line 2 (Apt., suite, building access code), City / Town / District (San Jose), State / Province / County / Region (CA), and ZIP / Postal Code (95112). At the bottom of the sidebar, there are links to Terms of Service, Privacy Policy, Host Guarantee, Guest Refund, Copyright Policy, Consent Disagree, and About Us. A language selector shows English.

Photos Tab in Adding a Listing

Host can add multiple images and video in this screen

The screenshot shows the Airbnb listing creation interface with the 'Photos' category selected in the sidebar. It features sections for uploading photos and videos. The 'Photos' section includes a 'Choose Files' button with '5 files' selected and a 'Click to Upload Photos' button. The 'Video' section includes a 'Select' button, a 'Click to Upload Video' button, and a video preview area showing a black frame with controls for play, volume, and seek. To the right, a box titled 'Guests Love Photos' contains the text: 'Include a few well-lit photos. Cell phone photos are just fine.' At the bottom, there are 'Back' and 'Next' buttons.

Pricing Tab in Adding a Listing

The screenshot shows the Airbnb 'Edit Listing' interface. On the left, a sidebar lists categories: Listing, Basics, Description, Location, Photos, Hosting, and Pricing. The Pricing section is currently selected. The main content area is titled 'Set a Nightly Price for Your Space' and includes a sub-section 'Base price' with a input field set to '\$ 500'. Below it is a 'Listing for bid' section with a checkbox. A note states: 'You can offer discounts for longer stays by setting [weekly](#) and [monthly](#) prices.' To the right, a callout box titled 'Nightly Price' provides information: 'You may want attract your first few guests by offering a great deal. You can always increase your price after you've received some great reviews.' At the bottom are 'Back' and 'Finish Remaining Steps' buttons.

After searching a property in Search bar and applying filters the following screen will appear Pinpoint pricing is available in the Maps.

The screenshot shows the Airbnb search results for 'Fresno, CA, United States'. It displays two listing cards: one for an 'Entire home/apt' priced at \$461 and another for an 'Entire home/apt' priced at \$318. To the right is a map of the Fresno area with red pins indicating specific listing locations and their prices: \$358, \$323, \$388, \$461, \$317, \$436, \$436, \$318, and \$436. The map also shows various towns like Prather, Tollhouse, Humphreys Station, Ovis, Piedra, Avocado, Squaw Valley, Sanger, Del Rey, Reedley, Selma, Kingsburg, Dinuba, Orosi, Culler, Yettem, Elderwood, and Pinehurst. A legend indicates 'Map' and 'Satellite' view options. At the bottom, there are links for 'Become a Host', 'Help', 'Trips', and 'Sal...'. The Google logo is visible at the bottom right.

SERVER AND CLIENT CONSOLE FOR SEARCH QUERY

dynamicPricingCron.js - Airbnb_Client - [~/Desktop/Airbnb17-new/Airbnb_Client]

```

GET /search?location=San+Jose,%20CA,%20United%20States&checkin=&checkout=&guests=1&room_type= 200 160.911 ms - 102638
GET /css/home.css 304 88.407 ms -
GET /css/common.css 304 106.751 ms -
GET /css/themes.css 304 93.597 ms -
GET /css/dynamic.css 304 98.106 ms -
GET /css/map-icons-master/dist/css/map-icons.css 200 3.718 ms - 9886
GET /css/map-search.css 200 1.292 ms - 31630
GET /images/user/%7B%7B%20rooms.images[0].%7D%7D 404 30.899 ms - 189270
GET /images/user/%7B%7B%20rooms.users.profile_picture.src%20%7D%7D 404 12.215 ms - 109270
GET /common.js 304 91.129 ms -
GET /controller/common_controller.js 304 94.410 ms -
GET /js/infbubble.js 200 3.188 ms - 6046
GET /css/map-icons-master/dist/css/map-icons.js 200 1.625 ms - 3962
GET /fonts/glyphicons-halflings-regular.woff 51824a8c080579d01078cff346a.woff 304 87.015 ms -
GET /fonts/circular-Air-Bold-ba3e38967877fa1729525589caf5.woff 304 87.960 ms -
GET /images/lopp/logo.png 204 97.828
GET /images/trip-icon.png 304 87.892 ms -
GET /images/search-icon.png 304 87.055 ms -
GET /images/index.gif 200 1.552 ms - 3947
response { total: 13,
  per_page: 13,
  current_page: 1,
  last_page: 1,
  next_page_url: null,
  prev_page_url: null,
  from: 1,
  to: 13,
  data:
  [ { id: '5844d9b1a7f87265a0a0ab34',
    owner_id: '5844d6803f3a625324b6593b',
    name: '101 San Fernando',
    summary: 'Sample Property, Spacious and open',
    accommodates: 1,
    bedrooms: 2,
    bathrooms: 1.5,
    ...
  } ]
}

```

Event Log

Unregistered VCS root detected: The directory /Users/maitrayshah/Desktop/Airbnb17-new is under Git, but is not registered in the Settings. // Add root Configure Ignore (5 minutes ago)

Airbnb_server - [~/Desktop/Airbnb17-new/Airbnb_server]

```

server.js - Run
4 Dec 21:23:15 - search_queue { location: 'San Jose, CA, United States',
  property_type: '',
  checkin: '',
  checkout: '',
  guests: '1' }
4 Dec 21:23:15 - Message: {"location":"San Jose, CA, United States","property_type":"","checkin":"","checkout":"","guests":"1"}
4 Dec 21:23:15 - DeliveryInfo: {"contentType":"application/json","contentEncoding":"utf-8","correlationId":"b731bf128854066684f7fc90dd33111","replyTo":"amq.gen-Ltc_dWmv
[ { _id: 5844d9b1a7f87265a0a0ab34,
  mediaId:
  { _id: 5844d9b1a7f87265a0a0ab35,
    videoUrl: '14809071564535844d6803f3a625324b6593b.mp4',
    __v: 0,
    imageUrls: [Object],
    multiplier: 1,
    revenue: 200,
    endDate: 'NaN',
    startDate: 'NaN',
    isAvailable: true,
    isApproved: true,
    createdDate: 1480907185188,
    longitude: '-121.886323600000001',
    latitude: '37.3362082',
    price: 200,
    description: 'Sample Property, Spacious and open',
    name: '101 San Fernando',
    bathrooms: 1.5,
    bedrooms: 2,
    zip: '95112',
    address: 'San Jose',
    state: 'CA',
    city: 'San Jose',
    category: 'Entire home/apt',
    maxGuest: 1,
    hostId:
    { _id: 5844d6803f3a625324b6593b,
      isActivated: true,
      isHost: true,
      createdDate: 1480906368829,
      userId: '681-22-2511',
      password: '$2a$10$UL3rqHx30Wbg5bWLVC51.THBreVBU7pqrvvChofXdwEajeQZ07ce',
      email: 'salmaan@airbnb.com' },
    ...
  }
}

```

Event Log

Unregistered VCS root detected: The directory /Users/maitrayshah/Desktop/Airbnb17-new is under Git, but is not registered in the Settings. // Add root Configure Ignore (24 minutes ago)

Adding Check In and Check Out Dates in filters

The screenshot shows the Airbnb search interface for Fresno, CA, United States. The search bar at the top has "Fresno, CA, United States". Below it, the "Dates" section shows "04-12-2016" and "07-12-2016" with a dropdown for "1 Guest". A calendar for December 2016 is displayed, with the 4th highlighted in red. To the right of the calendar is a "Apply Filters" button. On the left, there's a thumbnail for listing 461 and a snippet of text: "Magnam incident rem error accusa... Entire home/apt". On the right, there's a map of the Squaw Valley area with several price pins: \$358, \$323, \$388, \$317, \$461, \$436, \$318, and \$436. The map includes labels for Prather, Tolhouse, Humphreys Station, Trimmer, Balch Camp, Crabtree, Squaw Valley, Pinehurst, and various roads like 168, 180, 201, and 245.

On clicking the property Image The Property Details page will appear

The screenshot shows the Airbnb property details page for listing 101 San Fernando, located in San Jose, CA. The main image is a close-up of a dining table set with a candle and a bottle of wine. A price overlay says "\$ 200X 1=\$200 Per Night". Below the image, the listing information includes the user profile image (Salmaan), the address "101 San Fernando", "San Jose, CA", a "User Profile Image" placeholder, and icons for "Apartment", "1 guests", and "3 beds". To the right, there are fields for "Check In" (dd-mm-yyyy), "Check Out" (dd-mm-yyyy), "Guests" (dropdown), and a "Request to Book" button. A note at the bottom says "You'll be able to review before paying".

S Salmaan

localhost:3000/property?propertyId=5844d9b1a7f87265a0a0ab34

Search Property Details

User Profile Image Salmaan Apartment 1 guests 3 beds \$ 200X 1=\$200 Per Night

Check In dd-mm-yyyy Check Out dd-mm-yyyy Guests

Request to Book You'll be able to review before paying.

About this listing

Sample Property. Spacious and open

The Space Bed type: **Pull-out Sofa** Property type: **Bed & Breakfast** Accommodates: 1 Bedrooms: 2 Bathrooms: 1.5 Beds: 3

Amenities Essentials TV Air Conditioning + More

Cable TV Heating

Prices Extra people: **No Charge** Monthly Price: **\$ 6000 /month**

Salmaan Apartment Guests Beds

S Salmaan

localhost:3000/property?propertyId=5844d9b1a7f87265a0a0ab34

Search Property Details

User Profile Image Salmaan Apartment 1 guests 3 beds \$ 200X 1=\$200 Per Night

Check In dd-mm-yyyy Check Out dd-mm-yyyy Guests

Request to Book You'll be able to review before paying.

About this listing

Sample Property. Spacious and open

The Space Bed type: **Pull-out Sofa** Property type: **Bed & Breakfast** Accommodates: 1 Bedrooms: 2 Bathrooms: 1.5 Beds: 3

Amenities Essentials TV Air Conditioning + More

Cable TV Heating

Prices Extra people: **No Charge** Monthly Price: **\$ 6000 /month**
Weekly Price: **\$ 1400 /week**

Safety Features Smoke-Detector Carbon Monoxide Detector
 First Aid Kit Safety Card
 Fire-Extinguisher

Availability **1 night minimum stay** View Calendar

Salmaan Apartment Guests Beds

Photos and Video in Property Detail page

Screenshot of a web browser showing a property detail page. The URL is localhost:3000/property?propertyId=5844d9b1a7f87265a0a0ab34. The page displays a video player showing a listing intro video of a two-story house with a spiral staircase. Below the video is a section titled "Listing Pictures" featuring four thumbnail images: a close-up of a lit candle, a kitchen area with white cabinets, a dining room with a large window, and a modern sofa set.

Screenshot of a web browser showing a property detail page. The URL is localhost:3000/property?propertyId=5844d9b1a7f87265a0a0ab34. The page displays a video player showing a listing intro video of a two-story house with a spiral staircase. Below the video is a section titled "Listing Pictures" featuring four thumbnail images: a close-up of a lit candle, a kitchen area with white cabinets, a dining room with a large window, and a modern sofa set. At the bottom of the page, there is a "About the Host, Salmaan" section with a profile picture, the name "Salmaan", and the text "Member since December 2016".

SERVER AND CLIENT CONSOLE

dynamicPricingCron.js - Airbnb_Client - [~/Desktop/Airbnb17-new/Airbnb_Client]

```

GET /property?propertyId=5844d070eb037a6060fe1ee8 304 16.912 ms --
GET /css/common.css 304 100.284 ms --
GET /css/home.css 304 87.077 ms --
GET /css/themes.css 304 89.795 ms --
GET /css/dynamic.css 304 94.253 ms --
GET /css/room_detail.css 304 84.288 ms --
GET /css/default.css 304 87.865 ms --
GET /css/p3.css 304 89.308 ms --
GET /images/user/%7B%7Broom_result.users.profile_picture.src%7D%7D 404 11.884 ms - 109270
GET /js/room.js 304 87.427 ms --
GET /js/common.js 304 98.034 ms --
GET /controller/common_controller.js 304 93.500 ms --
GET /images//user/%7B%7Broom_result.images[0].src%7D%7D 404 14.630 ms - 109270
GET /images/user/%7B%7Buser.profileImageSrc%7D%7D 404 8.611 ms - 109270
GET /fonts/fontawesome-webfont.woff 304 89.576 ms --
GET /fonts/Cleaner-Air-Bold-b3c3806777f6f17295255589cb6f5.woff 304 87.015 ms --
GET /images/trip-icon.png 304 89.999 ms --
GET /images/user/%7B%7Broom_result.images[0].src%7D%7D 404 4.386 ms - 109270
GET /images/logo.png 304 86.202 ms --
GET /images/search-icon.png 304 95.172 ms --
GET /images/user/5844d68a3f3a625324b6593b.png 404 4.939 ms - 109270
GET /images/user/%7B%7Breview.imageUrl.split(',')[0].src%7D%7D 404 5.389 ms - 109270
GET /images/user/%7B%7Bimage%7D%7D 404 5.591 ms - 109270
GET /images/footer_cover_image.png 304 88.804 ms --
GET /images/user/ 404 3.455 ms - 109270
response: { reviews: [] },
rating: 'No Reviews',
id: '5844d070eb037a6060fe1ee8',
propertyId: '689-82-7736',
userId: '58449a9fe037a50f086f062',
name: 'Incidunt laudantium aspernatur fuga alias neque.',
summary: 'Rem aspernatur necessitatibus ipsam voluptas magni.',
accommodates: 1,
bedrooms: 5,
bathrooms: 9,
host_name: 'Miranda',
created_at: '2016-12-05T00:10:24.068Z',
updated_at: '2016-12-05T00:10:24.068Z',
photo_name: 'property_26.jpg',
images: [
  'property_26.jpg'
],
  
```

Event Log

Unregistered VCS root detected: The directory /Users/maitrayshah/Desktop/Airbnb17-new is under Git, but is not registered in the Settings. // Add root Configure Ignore (6 minutes ago)

151:1 LF: UTF-8: 8 Event Log

Airbnb_server - [~/Desktop/Airbnb17-new/Airbnb_server]

```

server.js - Airbnb_server - [~/Desktop/Airbnb17-new/Airbnb_server]
Run server.js
4 Dec 21:24:12 - property_detail_queue { id: '5844b070eb037a6060fe1ee8' }
4 Dec 21:24:12 - Message: {"id":"5844b070eb037a6060fe1ee8"}
4 Dec 21:24:12 - DeliveryInfo: {"contentType":"application/json","contentEncoding":"utf-8","correlationId":"9d3aacc9db0216d40dcc1006bf79ed18","replyTo":"amq.gen-Ltc_dW2mv5844b070eb037a6060fe1ee8"
response: { reviews: [] },
rating: 'No Reviews',
id: '5844b070eb037a6060fe1ee8',
propertyId: '689-82-7736',
userId: '58449a9fe037a50f086f062',
name: 'Incidunt laudantium aspernatur fuga alias neque.',
summary: 'Rem aspernatur necessitatibus ipsam voluptas magni.',
accommodates: 1,
bedrooms: 5,
bathrooms: 9,
host_name: 'Miranda',
created_at: '2016-12-05T00:10:24.068Z',
updated_at: '2016-12-05T00:10:24.068Z',
photo_name: 'property_26.jpg',
images: ['property_26.jpg','property_27.jpg','property_28.jpg','property_29.jpg','property_30.jpg'],
video_url: undefined,
isBidding: false,
sub_name: '',
property_type: 'Apartment',
room_type: 0,
beds: 0,
bed_type: 'Pull-out Sofa',
amenities: '1,2,4,5,6,7,10,11,15,17,18,19,22,23,24,25,28,29,30',
calendar_type: 'Always',
booking_type: null,
cancel_policy: 'Flexible',
popular: 'Yes',
started: 'Yes',
status: 'Listed',
deleted_at: null,
steps_count: 0,
property_type_name: 'Bed & Break Fast',
room_type_name: 'Shared room',
bed_type_name: 'Futon',
reviews_count: 0,
overall_star_rating: '',
startDate: undefined,
  
```

Event Log

Unregistered VCS root detected: The directory /Users/maitrayshah/Desktop/Airbnb17-new is under Git, but is not registered in the Settings. // Add root Configure Ignore (24 minutes ago)

117:1 LF: UTF-8: 8 Event Log

Profile page

Profile Page will display User is Listed Properties

A screenshot of a web browser showing an Airbnb profile page. The URL in the address bar is `localhost:3000/profile/681-22-2511`. The page header includes tabs for "Search", "Your Trips", and "Profile". The main content area features a profile picture of a man and the text "Hello, I'm Salmaan!" followed by "Member since Dec 2016". A sidebar on the left lists "Verifications" with two items: "Email Address Verified" and "Properties Verified". Below this is a section titled "Salmaan's Listed Properties (1)" with a thumbnail image of a bridge over water labeled "San Jose" and a "View Listing" button. Further down are sections for "Reviews (0)", "Reviews From Users", and "Reviews From Hosts". The browser toolbar at the top shows various open tabs and icons.

A screenshot of a web browser showing an Airbnb profile page for Salmaan, similar to the one above but with a different listing. The URL is again `localhost:3000/profile/681-22-2511`. The main content area shows a thumbnail image of a bridge over water labeled "San Jose" and a "View Listing" button. To the left is a sidebar with "Verifications" for "Email Address Verified" and "Properties Verified". Below this are sections for "Reviews (0)", "Reviews From Users", and "Reviews From Hosts". At the bottom of the page, there is a footer with language and currency selection dropdowns ("English", "USD"), links to "Company" (Terms of Service, Privacy Policy, etc.) and "Discover" (Travel Credit, Trust & Safety), and a "Hosting" section with "Why Host Hospitality". The browser toolbar at the top shows various open tabs and icons.

CLIENT AND SERVER CONSOLE CALL

dynamicPricingCron.js - Airbnb_Client - [~/Desktop/Airbnb17-new/Airbnb_Client]

```

info: Salman clicked on profile user=Salmaan, url_clicked=/userProfile
GET /profile/681-22-2511 304 7.666 ms --
GET /css/common.css 304 95.748 ms --
GET /css/themes.css 304 94.235 ms --
GET /css/home.css 304 91.441 ms --
GET /css/dynamic.css 304 87.002 ms --
GET /css/profile.css 304 88.502 ms --
info: Salman clicked on profile user=Salmaan, url_clicked=/userProfile
GET /images/user/%7B%7Buser.profileImage%7D%7D 404 15.702 ms - 109270
GET /images/user/%7B%7Buser.profileImage%7D%7D 404 14.044 ms - 109270
GET /profile/Images/user/%7B%7Breview.imageUrl.split(',')[0].%7D%7D 304 11.232 ms --
GET /profile/Images/user/%7B%7Breview.imageUrl.split(',')[0].%7D%7D 404 8.801 ms - 109270
GET /common.js 304 88.974 ms --
GET /controller/common_controller.js 304 90.004 ms --
681-22-2511
GET /images/logos/logo.png 304 84.002 ms --
response {_id: '5844d6803f3a625324b6593b', isActivated: true, isHost: true, createdDate: 1480906368829, userId: '681-22-2511', password: '52a$10$UL3rqhx30Wbg5bMLVCS1.TH8reV8U7pqrvvChofXdwEajeQZ07ce', email: 'salmaan@airbnb.com', lastName: 'Pehlari', firstName: 'Salmaan', __v: 0, isApproved: false, address: 'San Jose State University', city: 'San Jose', state: 'CA', zip: '95112', profileImage: '5844d6803f3a625324b6593b.png', cardNumber: 4111111111111111, cvv: 111, expDate: '2/2022' }
{ _id: '5844d6803f3a625324b6593b', isActivated: true, isHost: true, createdDate: 1480906368829, userId: '681-22-2511' }

```

Event Log

Unregistered VCS root detected: The directory /Users/maitrayshah/Desktop/Airbnb17-new is under Git, but is not registered in the Settings. // Add root Configure Ignore (16 minutes ago)

Airbnb_server - [~/Desktop/Airbnb17-new/Airbnb_server]

```

server.js
Run server.js

{
  _id: '5844d6803f3a625324b6593b',
  isActivated: true,
  isHost: true,
  createdDate: 1480906368829,
  userId: '681-22-2511',
  password: '52a$10$UL3rqhx30Wbg5bMLVCS1.TH8reV8U7pqrvvChofXdwEajeQZ07ce',
  email: 'salmaan@airbnb.com',
  lastName: 'Pehlari',
  firstName: 'Salmaan',
  __v: 0,
  isApproved: false,
  address: 'San Jose State University',
  city: 'San Jose',
  state: 'CA',
  zip: '95112',
  profileImage: '5844d6803f3a625324b6593b.png',
  cardNumber: 4111111111111111,
  cvv: 111,
  expDate: '2/2022' }
{ userId: '5844d6803f3a625324b6593b' }

[ { _id: '5844de4a7f87265a0a0ab39',
  mediaId: { _id: '5844de4a7f87265a0a0ab3a', __v: 0, imageUrl: [Object] },
  multiplier: 1,
  revenue: 0,
  endDate: 'NaN',
  startDate: 'NaN',
  isAvailable: true,
  isApproved: false,
  createdDate: '1480908452742',
  longitude: '-121.8863286000001',
  latitude: '37.3382082',
  price: 312,
  description: 'House For Rent',
  name: 'Spacious House',
  bathrooms: 2,
  bedrooms: 2,
  zip: '95112',
  address: 'San Jose',
  state: 'CA',
  __v: 0 } ]

```

Event Log

Unregistered VCS root detected: The directory /Users/maitrayshah/Desktop/Airbnb17-new is under Git, but is not registered in the Settings. // Add root Configure Ignore (34 minutes ago)

You can Edit the profile in the following screen

A screenshot of a web browser displaying the Airbnb profile editing interface. The URL in the address bar is `localhost:3000/editProfilePage`. The page has a dark header with the Airbnb logo and navigation links for Dashboard, Your Listings, Your Trips, Profile (which is active), and Account. On the left, there's a sidebar with 'Edit Profile' sections for Photo, Reviews, and a 'View Profile' button. The main content area is titled 'Required' and contains fields for First Name (Salmaan), Last Name (Pehlari), Email Address (salmaan@airbnb.com), and Address, State, City, and Zip Code fields. A note below the email field states: 'This is only shared once you have a confirmed booking with another Airbnb user.' and 'We won't share your private email address with other Airbnb users.'

User can upload photos in edit profile page

A screenshot of a web browser displaying the Airbnb profile photo upload interface. The URL in the address bar is `localhost:3000/getUserPhotoPage`. The layout is similar to the previous screenshot, with the Airbnb logo and navigation links at the top. The sidebar on the left shows 'Edit Profile', 'Photos' (which is active), and 'Reviews'. The main content area is titled 'Profile Photo' and features a circular placeholder image of a man's face. Below the image is a note: 'Clear frontal face photos are an important way for hosts and guests to learn about each other. It's not much fun to host a landscape! Please upload a photo that clearly shows your face.' There is a 'Upload a file from your computer' input field and a red 'Upload' button. At the bottom of the page, there are language selection dropdowns for English and USD, and links for Company (Terms of Service, Privacy Policy, Host Guarantee), Discover (Travel Credit, Trust & Safety), and Hosting (Why Host, Hospitality).

Your Trips Page

A screenshot of a web browser displaying the Airbnb 'Your Trips' page. The URL in the address bar is `localhost:3000/yourTrips`. The page header includes the Airbnb logo, a search bar with placeholder text 'Where to?', and navigation links for 'Become a Host', 'Help', 'Trips', and 'test'. Below the header, there are links for 'Your Listings', 'Your Trips' (which is currently selected), 'Profile', and 'Account'. The main content area is titled 'Your Trips' and contains a table with one row. The table columns are 'Status', 'Location', 'Host', 'Dates', and 'Options'. The data in the row is: Status - Pending, Location - San Jose, San Jose, Host - Salmaan Pehlari, Dates - Dec 4, 2016 -- Dec 5, 2016, Options - View Itinerary.

Reviews and ratings can be added after the trip is accepted

A screenshot of a web browser displaying the Airbnb 'Your Trips' page, identical to the previous one but with a different trip status. The URL is again `localhost:3000/yourTrips`. The trip details are: Status - Accepted, Location - San Jose, San Jose, Host - Salmaan Pehlari, Dates - Dec 4, 2016 – Dec 5, 2016. The 'Options' column now includes a red button labeled 'Write Review' and a text input field containing the text 'Awesome Trip!'. Below the input field is a rating slider set to the value '2'. At the bottom of the 'Options' section are two red buttons: 'Submit Review' and 'View Itinerary'.

Add Host Review

Screenshot of the Airbnb 'Your Trips' section showing a host review for an accepted trip.

The screenshot shows a table with columns: Status, Location, Host, Dates, and Options. One row is highlighted with a green 'Accepted' status, San Jose location, Salmaan Pehlari as the host, Dec 4, 2016 - Dec 5, 2016 dates, and an 'Options' section.

The 'Options' section contains a 'Write Review' button, a text area with placeholder text 'Awesome Place. Must visit!', a rating input field set to 5, a file upload input labeled 'Choose Files' with the path '58427bccc4...882581.png', and a thumbnail image of a room interior.

Below the table are links: 'Submit Review', 'View Itinerary', and 'View receipt'.

Add user review

Screenshot of the Airbnb 'Your Listings' section showing a guest review for an accepted listing.

The screenshot shows a table with columns: Status, Dates and Location, Guest, and Details. One row is highlighted with an 'Accepted' status, dates from 12/4/16 to 12/5/16, location '101 San Fernando San Jose, CA, 95112', and a guest named 'test test'.

The 'Details' section contains a 'Write Review' button, a text area with placeholder text 'Great Person! Kept the place clean!', a rating input field set to 5, a file upload input labeled 'Choose Files' with the path '583d4d1778...45280.png', and a thumbnail image of a hallway.

Below the table is a 'Submit Review' button.

localhost:3000 says:

Review added

OK

Print this page

Status	Dates and Location	Guest	Details
Hide Past Reservations			
Status	Dates and Location	Guest	Details
Accepted	12/4/16 to 12/5/16 101 San Fernando San Jose San Jose , CA , 95112	 test test Contact by Email	\$200 Write Review Great Person! Kept the place clean! S <input type="button" value="Choose Files"/> 583d4d1778...45280.png 
Submit Review			

Your Listings Page

localhost:3000/yourListings

Search **Where to?**

Add a Listing **Help** **Trips** **Salmaan**

Dashboard **Your Listings** **Your Trips** **Profile** **Account**

Active Listings

Pending Listings

Your Reservations

Unapproved Reservations

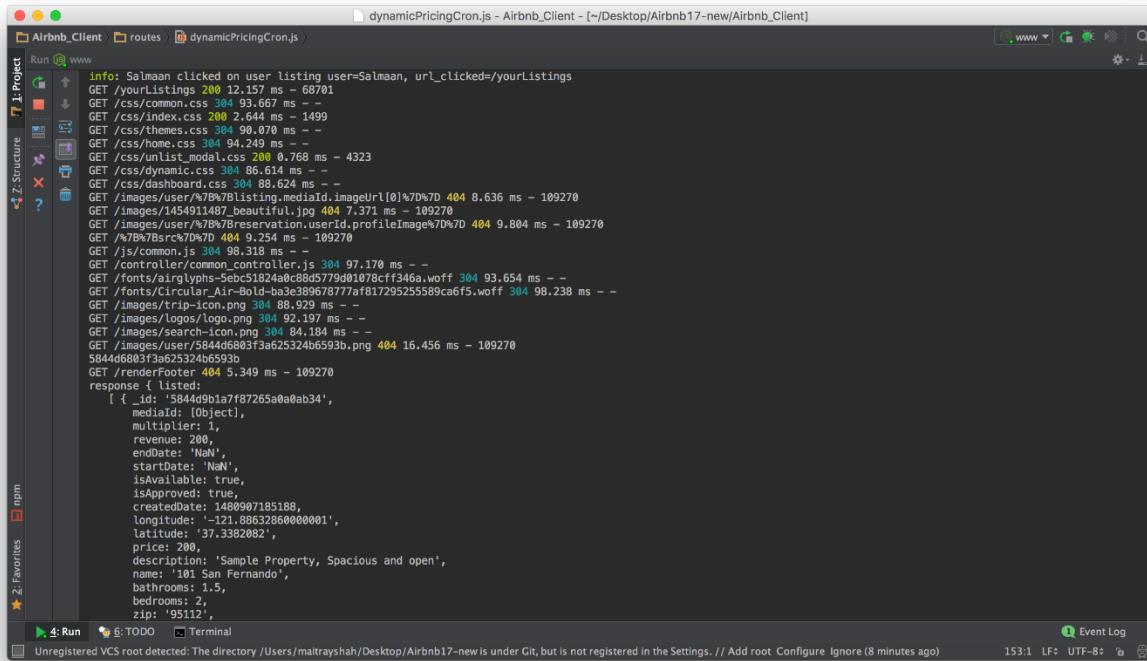
All Reservations

Print this page

Action	Dates and Location	Guest	Details
Approve	12/4/16 to 12/5/16 101 San Fernando San Jose San Jose , CA , 95112	 test test Contact by Email	\$ 200 per night Multiplier : 1 Days : 1 Total : \$ 200

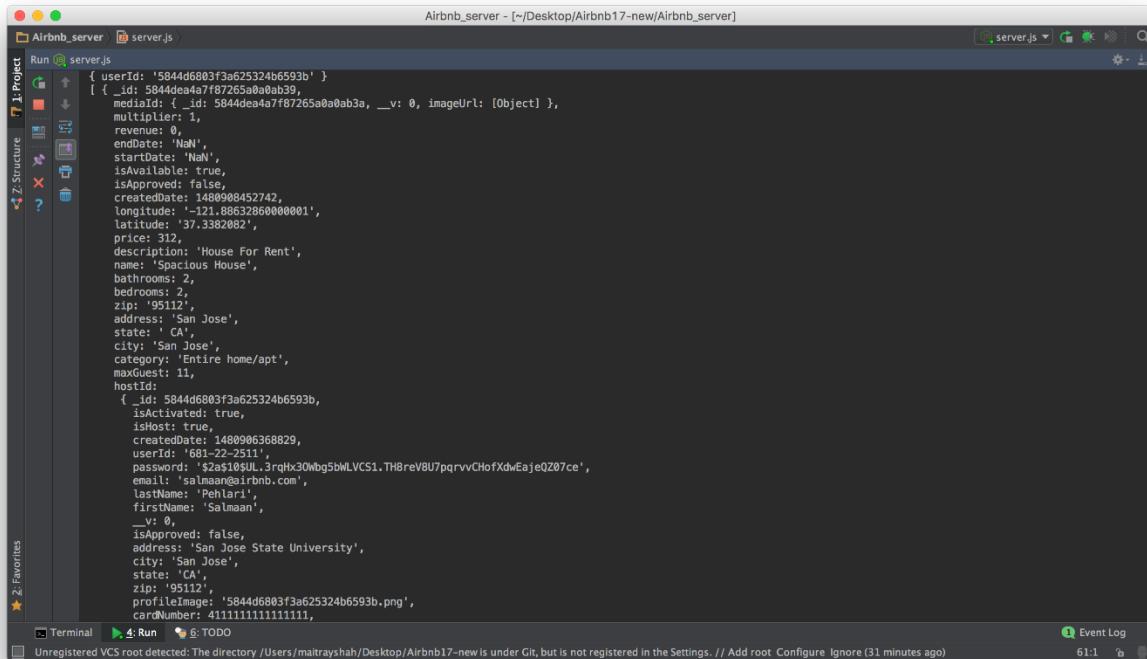
View upcoming reservations

SERVER AND CLIENT CONSOLE CALL FOR USER LISTING



A screenshot of the Airbnb Client application in a code editor. The title bar says "dynamicPricingCron.js - Airbnb_Client - [~/Desktop/Airbnb17-new/Airbnb_Client]". The code in the editor shows a series of GET requests and a response object. The response object contains a list of properties, each with fields like _id, mediaId, multiplier, revenue, enddate, startdate, isAvailable, isApproved, createdDate, longitude, latitude, price, description, name, bathrooms, bedrooms, zip, and address. The address field for the first property is '101 San Fernando'.

```
info: Salmaan clicked on user listing user=Salmaan, url_clicked=/yourListings
GET /yourListings 200 12.157 ms - 68701
GET /css/common.css 304 93.667 ms --
GET /css/index.css 200 2.644 ms - 1499
GET /css/themes.css 304 90.070 ms --
GET /css/home.css 304 94.249 ms --
GET /css/unlist_modal.css 200 0.768 ms - 4323
GET /css/dynamic.css 304 86.614 ms --
GET /css/dashboard.css 304 88.624 ms --
GET /images/user/%7B%7Blisting.mediaId.imageUrl%7D%7D 404 8.636 ms - 109270
GET /images/1454911487_beautiful.jpg 404 7.371 ms - 109270
GET /images/user/%7B%7Breservation.userId.profileImage%7D%7D 404 9.804 ms - 109270
GET /fb7b78rsrcs 304 98.318 ms --
GET /js/common.js 304 98.318 ms --
GET /controller/common_controller.js 304 97.170 ms --
GET /fonts/circular_Air-Bold-ba3e389678777af17295255589ca6f5.woff 304 93.654 ms --
GET /images/trip-icon.png 304 88.929 ms --
GET /images/logos/logo.png 304 92.197 ms --
GET /images/search-icon.png 304 84.184 ms --
GET /images/user/5844d6803f3a625324b6593b.png 404 16.456 ms - 109270
5844d6803f3a625324b6593b
GET /renderFooter 404 5.349 ms - 109270
response { listed:
  [ { _id: '5844d69b1a7f87265a0a0ab34',
    mediaId: [Object],
    multiplier: 1,
    revenue: 200,
    endDate: 'NaN',
    startDate: 'NaN',
    isAvailable: true,
    isApproved: true,
    createdDate: 1480907185188,
    longitude: '-121.886328600000001',
    latitude: '37.3382082',
    price: 200,
    description: 'Sample Property, Spacious and open',
    name: '101 San Fernando',
    bathrooms: 1.5,
    bedrooms: 2,
    zip: '95112',
    address: '101 San Fernando' } ] }
```



A screenshot of the Airbnb Server application in a code editor. The title bar says "server.js - Airbnb_server - [~/Desktop/Airbnb17-new/Airbnb_server]". The code in the editor shows a list of user objects. Each user object has fields like _id, mediaId, multiplier, revenue, enddate, startdate, isAvailable, isApproved, createdDate, longitude, latitude, price, description, name, bathrooms, bedrooms, zip, address, state, city, category, maxGuest, hostId, and lastName. The address field for the first user is 'San Jose'.

```
{ userId: '5844d6803f3a625324b6593b' }
[ { _id: '5844dea4a7f87265a0a0ab39',
  mediaId: { _id: '5844dea4a7f87265a0a0ab3a', __v: 0, imageUrl: [Object] },
  multiplier: 1,
  revenue: 0,
  endDate: 'NaN',
  startDate: 'NaN',
  isAvailable: true,
  isApproved: false,
  createdDate: 1480908452742,
  longitude: '-121.886328600000001',
  latitude: '37.3382082',
  price: 312,
  description: 'House For Rent',
  name: 'Spacious House',
  bathrooms: 2,
  bedrooms: 2,
  zip: '95112',
  address: 'San Jose',
  state: 'CA',
  city: 'San Jose',
  category: 'Entire home/apt',
  maxGuest: 11,
  hostId:
    { _id: '5844d6803f3a625324b6593b',
      isActivated: true,
      isHost: true,
      createdDate: 1480906368829,
      userId: '681-22-2511',
      password: '$2as10sUL3qjbx30Wbg5bwLVC51.TH8reVBU7pqrvvChofXdwEaje0Z07ce',
      email: 'salmaan@airbnb.com',
      lastName: 'Penlari',
      firstName: 'Salmaan',
      __v: 0,
      isApproved: false,
      address: 'San Jose State University',
      city: 'San Jose',
      state: 'CA',
      zip: '95112',
      profileImage: '5844d6803f3a625324b6593b.png',
      cardNumber: 4111111111111111 } ] }
```

Your past reservations page

The screenshot shows the Airbnb account interface. The top navigation bar includes links for 'Search', 'Your Trips', 'Active Listings', and the user's name 'Salmaan'. Below the bar, there are tabs for 'Dashboard', 'Your Listings', 'Your Trips', 'Profile', and 'Account'. On the left sidebar, under 'Your Reservations', there are links for 'Active Listings', 'Pending Listings', and 'Unapproved Reservations'. The main content area is titled 'All Reservations' and contains a table with columns for 'Status', 'Dates and Location', 'Guest', and 'Details'. One listing is visible:

Status	Dates and Location	Guest	Details
Accepted	12/4/16 to 12/5/16 101 San Fernando San Jose San Jose , CA , 95112	test test Send Message Contact by Email	\$200 Write Review Print Confirmation

Below the table, there is a link 'Hide Past Reservations'.

Delete Host Account Page

You can delete your host account

The screenshot shows the Airbnb account interface. The top navigation bar includes links for 'Search', 'Your Trips', 'Account', and the user's name 'Salmaan'. Below the bar, there are tabs for 'Dashboard', 'Your Listings', 'Your Trips', 'Profile', and 'Account'. On the left sidebar, under 'Deactivate Host Account', there are links for 'Payment Method', 'Transaction History', 'Security', and 'Deactivate Host Account'. The main content area is titled 'Deactivate Host Account' and contains the following text:

Once your host account is deactivated you will no longer be able add listings on Airbnb and all your listings will be removed

Are you sure you want to deactivate your host account? Click cancel to return

At the bottom right are two buttons: 'Delete' (in red) and 'Cancel'.

The footer of the page includes language and currency selection dropdowns ('English', 'USD'), and links for 'Company' (Terms of Service, Privacy Policy, Host Guarantee, Guest Refund, Copyright Policy, General Disclaimers), 'Discover' (Travel Credit, Trust & Safety), and 'Hosting' (Why Host, Hospitality).

You can change your password in the security panel of the account tab as shown In the following screen

The screenshot shows the Airbnb account security page. The URL in the address bar is `localhost:3000/Account_Security`. The page has a dark header with the Airbnb logo and a search bar. Below the header, there are navigation links: Dashboard, Your Listings, Your Trips, Profile, and Account. The Account link is highlighted. On the left, there's a sidebar with options: Payment Method, Transaction History, Security (which is selected), and Deactivate Host Account. The main content area is titled "Change Your Password" and contains three input fields: "Old Password", "New Password", and "Confirm Password". A red "Update Password" button is at the bottom right. At the bottom of the page, there are language and currency dropdowns set to English and USD, respectively. There are also links for Company, Discover, and Hosting.

Validations in change password screen

If newly entered password and confirm passwords do not match then the following screen will be appeared

This screenshot shows the same Airbnb account security page as the previous one, but with validation errors. The "Confirm Password" field is highlighted in orange, indicating it is invalid. A red box at the bottom contains the message "Passwords do not match!". The rest of the page structure is identical to the first screenshot.

Add payment Method page

User can enter Credit card details in the following screen.

The screenshot shows the Airbnb account interface. In the top navigation bar, there are tabs for 'Search', 'Your Trips', and 'Account'. Below the navigation bar, the 'Account' tab is selected. On the left, a sidebar menu includes 'Payment Method', 'Transaction History', 'Security', and 'Deactivate Host Account'. The main content area is titled 'Payment Methods' and contains a form for adding a credit card. The form fields are: 'Name on Card' (Salmaan), 'Card Number' (5264952000142325), 'Expiration Date' (Feb (02) 2019), and 'Card CVV' (123). A red 'Add Payout Method' button is at the bottom. At the bottom of the page, there are language and currency dropdowns set to 'English' and 'USD', and links for 'Company', 'Discover', and 'Hosting'.

Payment Validation page

The screenshot shows the Airbnb booking validation page. The top navigation bar includes 'Search', 'Booking', and 'Account' tabs, with 'Account' selected. The main content area is titled 'Payment' and contains a form for entering payment details. The fields are: 'Country' (United States), 'Payment type' (Credit Card), 'Card type' (Visa), 'Card number' (empty field with error message 'Please Enter Valid Card Details'), 'Expires on' (mm and yyyy dropdowns with error message 'Please Enter Valid Expire Dates'), and 'Security code' (empty field with error message 'Please Enter Valid Security Code'). To the right of the form, there is a large image of a bedroom labeled '101 San Fernando' and 'San Jose, CA'. Below the image, it says 'Entire home/apt for 2 Guests' and '4-12-2016 to 5-12-2016'. At the bottom, there is a section for 'Billing Information' with fields for 'First name' and 'Last name'.

localhost:3000/getPaymentPage?propertyId=5844d9b1a7fb7265a0a0ab34&checkin=4-12-2016&checkout=5-12-2016&guests=2.8

Please Enter Valid Expire Dates

Please Enter Valid Security Code

Billing Information

First name: test Last name: test

Postal code: United States

Cancellation Policy	Flexible
House Rules	Read policy
Nights	1
\$200 x 1 per night	\$200
\$200 x 1 night	\$200
Airbnb Service Fees	\$35
Total	\$235

House Rules

By booking this space you're agreeing to follow House Rules..

By clicking on "Continue", you agree to pay the total amount shown, which includes Service Fees, on the right and to the [Terms of Service](#), [House Rules](#), [Cancellation Policy](#) and [Guest Refund Policy](#).

Confirm

CLIENT AND SERVER CONSOLE – PAYMENT PAGE



A screenshot of a Mac OS X desktop environment. The top menu bar shows the application name "dynamicPricingCron.js - Airbnb_Client - [~/Desktop/Airbnb17-new/Airbnb_Client]" and system status icons. The main window is a file browser titled "Airbnb.Client" showing a file named "dynamicPricingCron.js". The code editor below displays the contents of this file, which is a JSON object representing a payment page response. The JSON includes fields like address, bathrooms, revenue, and a host profile with details such as name, city, state, and payment information.

```
POST /loadPaymentPage 200 136.396 ms - 446
response { _id: '5844b070eb037a6060feee8',
  bathrooms: 9,
  revenue: 3292,
  address: '81702 Ray Manors Apt. 809',
  isApproved: true,
  multiplier: 1.2,
  propertyId: '690-82-7736',
  category: 'Shared room',
  city: 'San Jose',
  zip: '56948',
  state: 'CA',
  maxGuest: 1,
  latitude: '36.89866748',
  hostId:
  { _id: '58449a9feb837a50f086f062',
    city: 'South Lindamouth',
    profileImage: '29.jpg',
    zip: '36809',
    firstName: 'Miranda',
    isApproved: true,
    cvv: 636,
    lastName: 'White',
    createDate: '1480891039273',
    userId: '750-85-1900',
    updateDate: '04/22',
    state: 'Maryland',
    cardNumber: 333767685002982,
    phoneNumber: '184-973-0042',
    address: '692 David Radial Suite 292',
    isHost: true,
    password: '$2a$10$98gGUAZn0.AjRZMrKyMew0af7u0yGV/Me.NmZhkxq0aK1Iz59jFRG',
    isActivated: true,
    email: 'xcruz@gmail.com' },
  mediaId:
  { _id: '584392c81fb52c08143c2578',
    __v: 0,
    imageUrl:
      [ 'property_26.jpg',
        'property_27.jpg',
        'property_28.jpg' ] }
```

View Itinerary page

S Salmaan

localhost:3000/itinerary?tripId=321-09-7635

34

Where to?

Become a Host Help Trips test

You're going to San Jose

Reservation code: [View receipt](#).

Check In	Sunday, December 4, 2016	Check Out	Monday, December 5, 2016
	Flexible check in time		Flexible check out time

Address San Jose
San Jose
95112 [Get directions](#)

Host Salmaan [Message host](#) 



localhost:3000/itinerary?tripId=321-09-7635

Flexible check in time Flexible check out time

Address San Jose
San Jose
95112 [Get directions](#)

Host Salmaan [Message host](#)

Billing 3 nights \$200 [Detailed receipt](#)



English ▾
USD ▾

Company
[Terms of Service](#)
[Privacy Policy](#)
[Host Guarantee](#)
[Guest Refund](#)

Discover
[Travel Credit](#)
[Trust & Safety](#)

Hosting
[Why Host](#)
[Hospitality](#)

Transaction History Page

The screenshot shows the Airbnb Transaction History page. The URL in the address bar is `localhost:3000/Account_Transactions`. The main content area displays a table titled "PayOut Transactions". The table has columns: Date, Type, Details, Amount, and User. There is one entry: Dec 4, 2016, Credit Card, 101 San Fernando, 200, test.

Date	Type	Details	Amount	User
Dec 4, 2016	Credit Card	101 San Fernando	200	test

Billing Receipt

The screenshot shows the Airbnb Customer Receipt page. The URL in the address bar is `localhost:3000/receipt/5844db3da7f87265a0a0ab37`. The receipt is for a stay on Sun Dec 04 2016, Confirmation Code: 4TOQ5I, Receipt # 023-76-0688. It lists the guest as "test test" staying at "San Jose CA" with "Salmaan Pehlari" as the host. The stay was from 12/4/2016 to 12/5/2016, 12:00 PM (noon) to 12:00 PM (noon). The total cost was \$200 (\$200 Per Night). The receipt includes the Airbnb contact information: +1 (415) 800-5959 and WWW.AIRBNB.COM/CONTACT.

Name	Travel Destination	Nights	Accommodation Type
test test	San Jose CA	1	Entire home/apt

Accommodation Address	Accommodation Host	Check In	Check Out
San Jose San Jose CA	Salmaan Pehlari	12/4/2016 12:00 PM (noon)	12/5/2016 12:00 PM (noon)

Payment Details

1 Nights	\$200 (\$200 Per Night)
----------	-------------------------

Salmaan

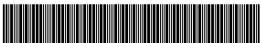
localhost:3000/receipt/5844db3da7f87265a0a0ab37

Receipt, Confirmation Co.

Payment Details

1 Nights	\$200 (\$200 Per Night)
Dynamic Price (1X)	\$200
Airbnb Service Fee	\$35
Airbnb Concierge <small>BETA</small>	Free
Total	\$235

Payment Received: Sun Dec 04 2016 19:19:40 GMT-0800 (Pacific Standard Time)	\$235
--	-------



THANKS FOR TRAVELLING WITH AIRBNB

Airbnb is authorized to accept Accommodation Fees on behalf of the Host as its limited agent. This means that your payment obligation to the Host is satisfied by your payment to Airbnb. Any disagreements by the Host regarding that payment must be settled between the Host and Airbnb.

You can save the receipt as pdf format

Print

Total: 1 page

Save Cancel

Destination: Save as PDF Change...

Pages: All e.g. 1-5, 8, 11-13

Layout: Portrait

Paper size: Letter

Margins: Default

Options: Headers and footers Background graphics

Customer Receipt
CONFIRMATION CODE: 4TQG8I
Rec'd Dec 04 2016
RECEIPT # 033-16-0388

Name	Travel Destination	Nights	Accommodation Type
Salmaan	San Jose CA	1	Entire Apartment
	Accommodation Address	Check In	Check Out
	1 bedroom	10:00AM (1000 HRs)	10:00AM (1000 HRs)

Payment Details

1 Nights	\$200 (\$200 Per Night)
Dynamic Price (1X)	\$200
Airbnb Service Fee	\$35
Airbnb Concierge <small>BETA</small>	Free
Total	\$235

Payment Received: Sun Dec 04 2016 19:19:40 GMT-0800 (Pacific Standard Time)	\$235
--	-------

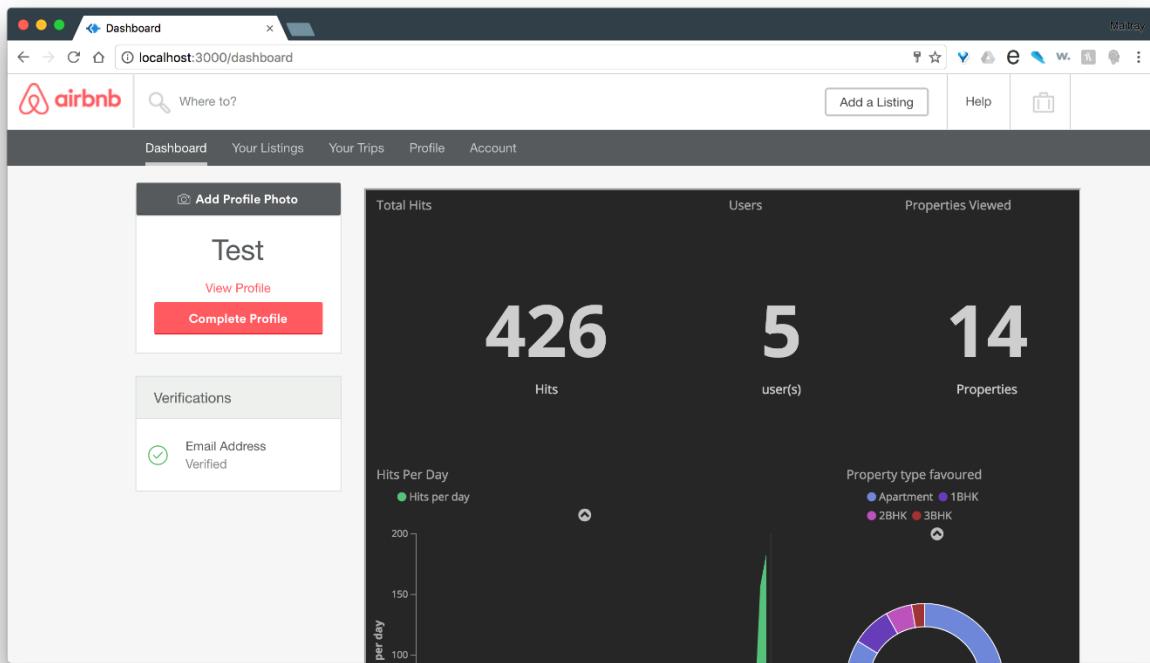


THANKS FOR TRAVELLING WITH AIRBNB

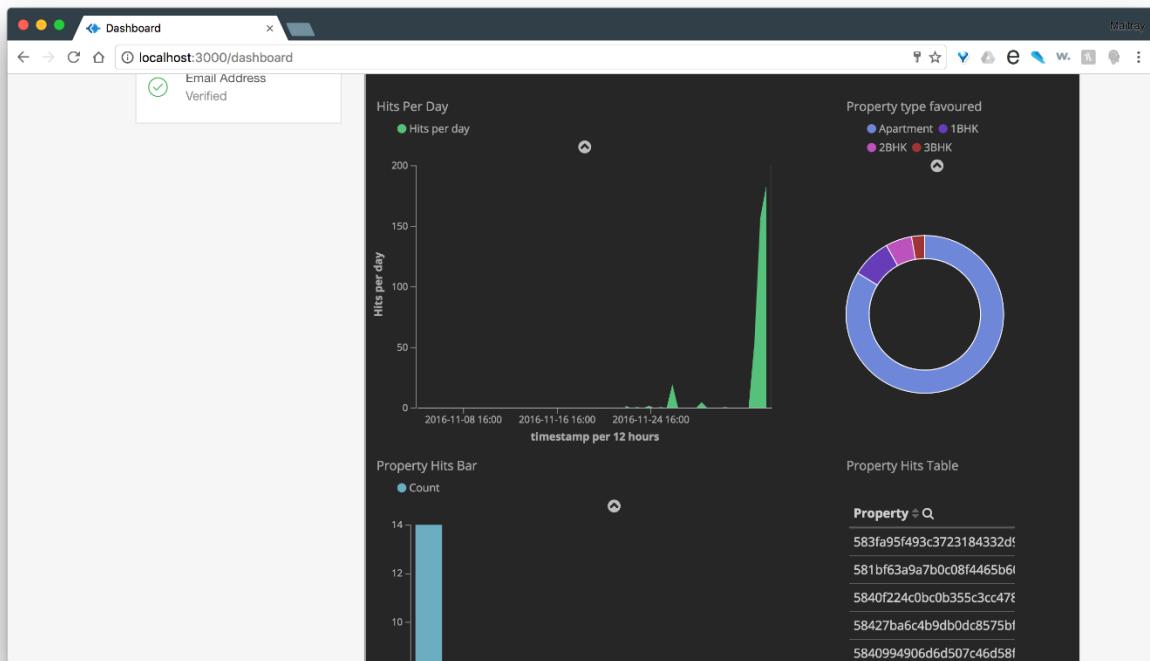
Airbnb is authorized to accept Accommodation Fees on behalf of the Host as its limited agent. This means that your payment obligation to the Host is satisfied by your payment to Airbnb. Any disagreements by the Host regarding that payment must be settled between the Host and Airbnb.

Analytics Module

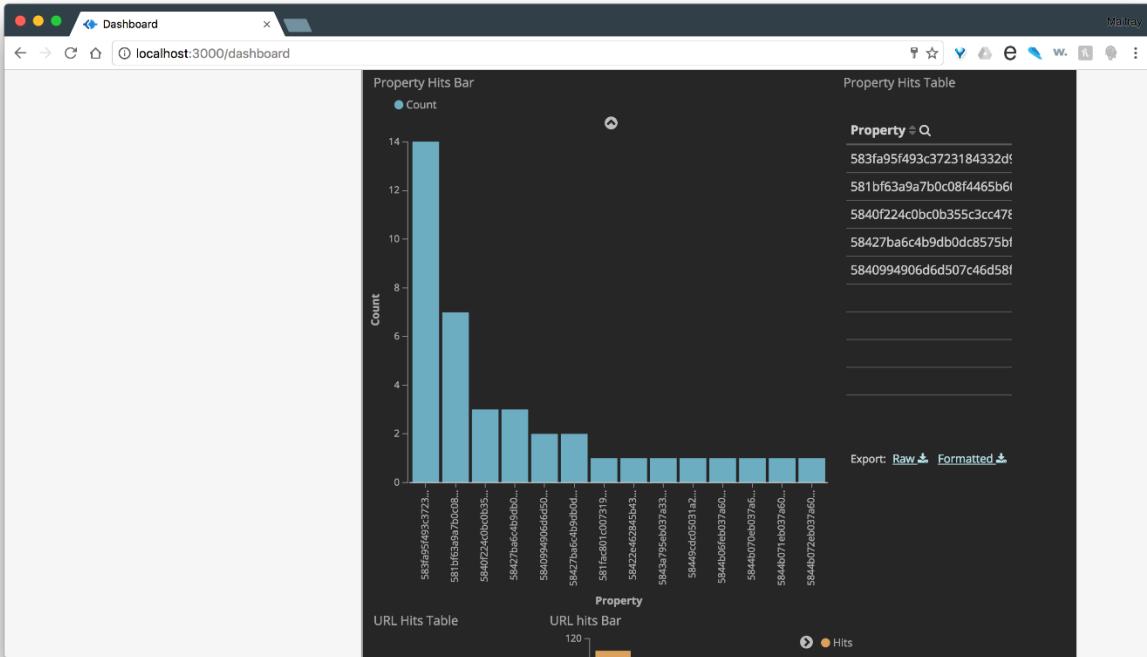
Dashboard tab



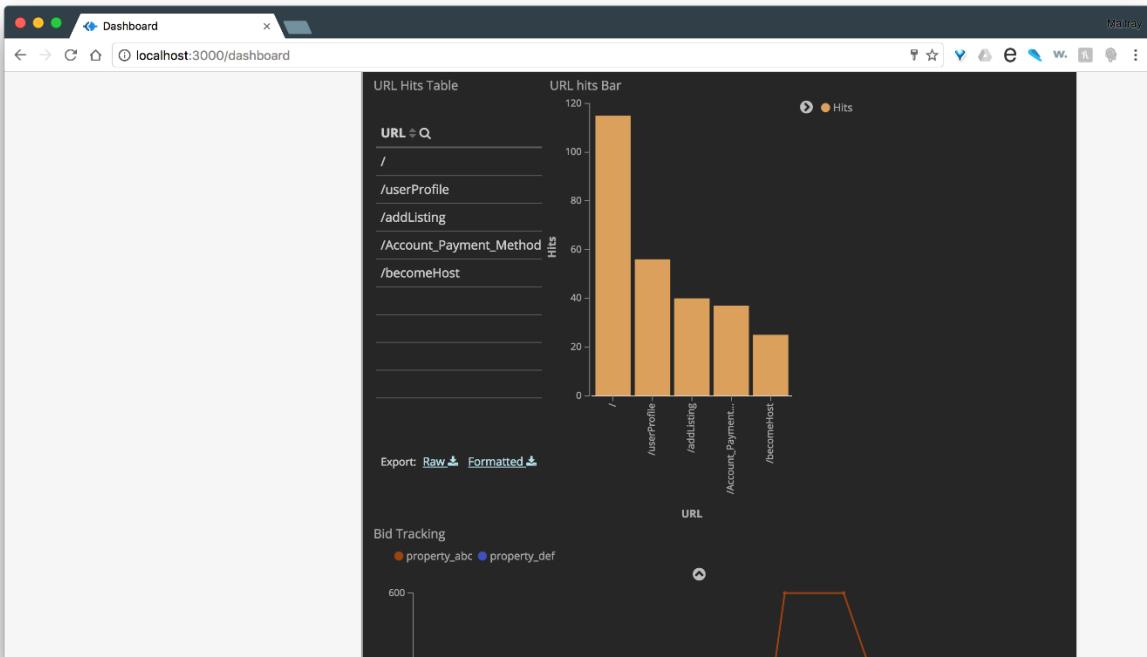
Property Hits and Hits per page will be shown in the following page



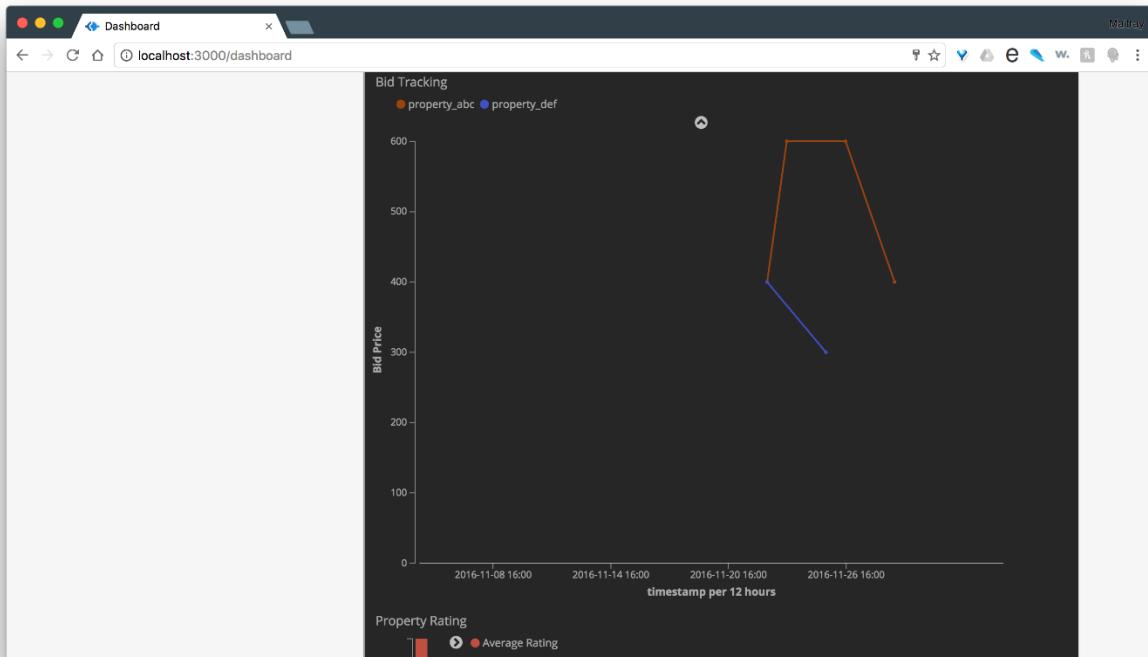
Property Hits bar will be shown in the following screen



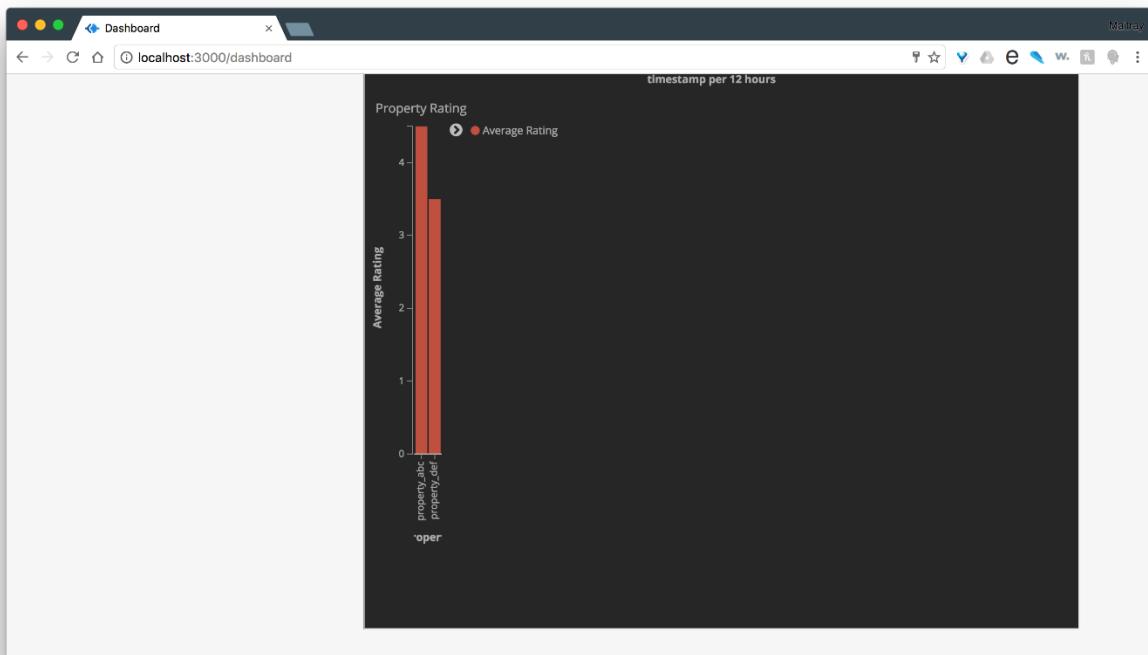
URL hits bar will appear in the following screen



Bid Tracking will appear in the following page



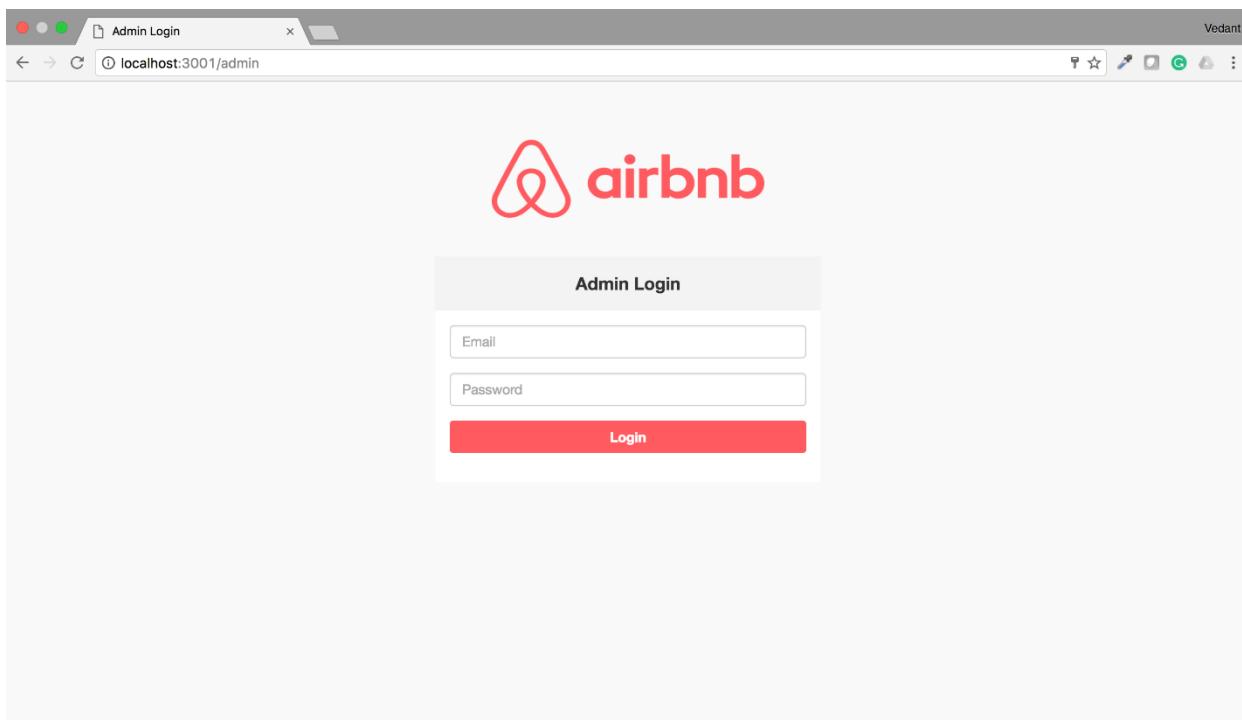
Property Rating will appear in the following page



Admin portal features

- View total users, hosts, properties, revenue and number of bills
- View top 10 properties with their revenue
- View 10 cities with their revenue
- View top 10 hosts with their revenue
- View active Users, Hosts and Properties
- View pending Users, Hosts and Properties
- Approved pending Host and Property requests
- View all the bills
- Add new admin to for the system

Admin login page



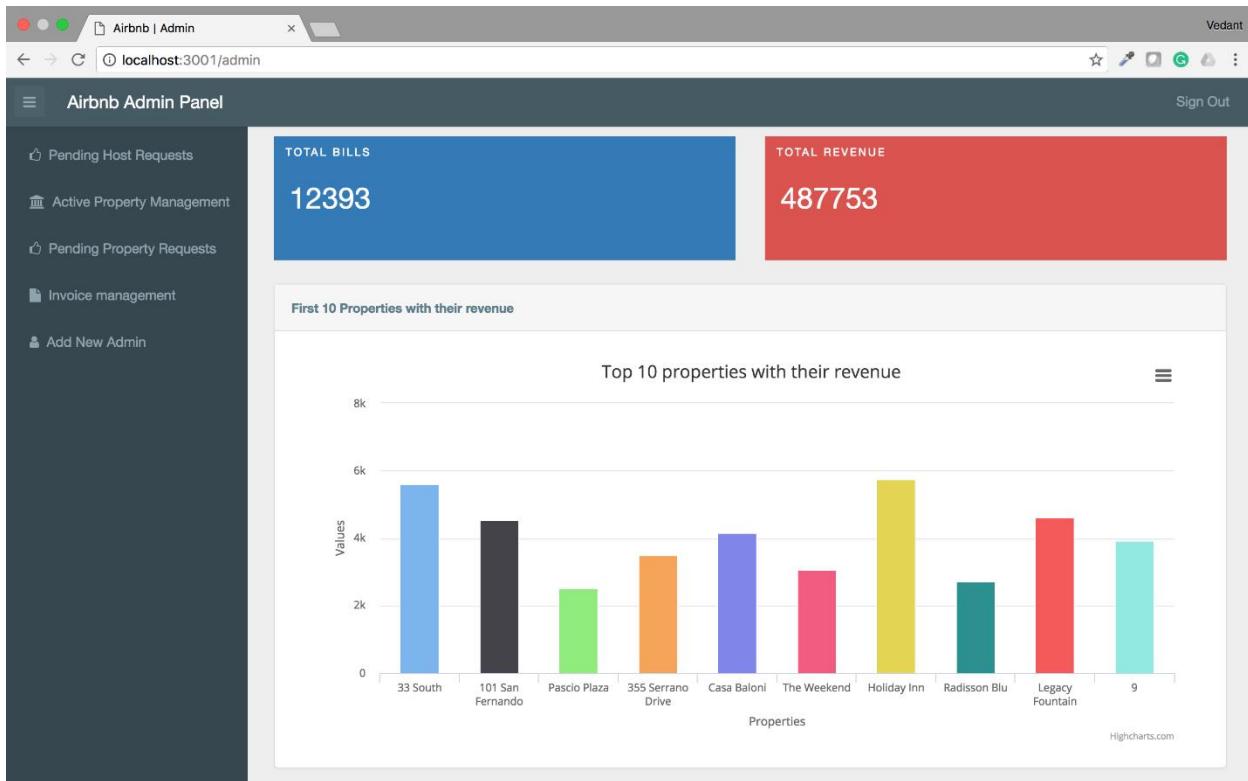
Admin dashboard

The screenshot shows the Airbnb Admin Panel dashboard. On the left is a sidebar with a user profile icon and the text "Welcome, Admin". The sidebar includes links for "Dashboard", "User Management", "Active Host Management", "Pending Host Requests", "Active Property Management", "Pending Property Requests", "Invoice management", and "Add New Admin". The main content area has a title "Dashboard". It features four large colored boxes: a teal box for "TOTAL HOSTS" (11033), a green box for "TOTAL USERS" (11092), an orange box for "TOTAL PROPERTIES" (10232), and a blue box for "TOTAL BILLS" (12393). Below these is a red box for "TOTAL REVENUE" (487753). A section titled "First 10 Properties with their revenue" contains a bar chart with the following approximate data:

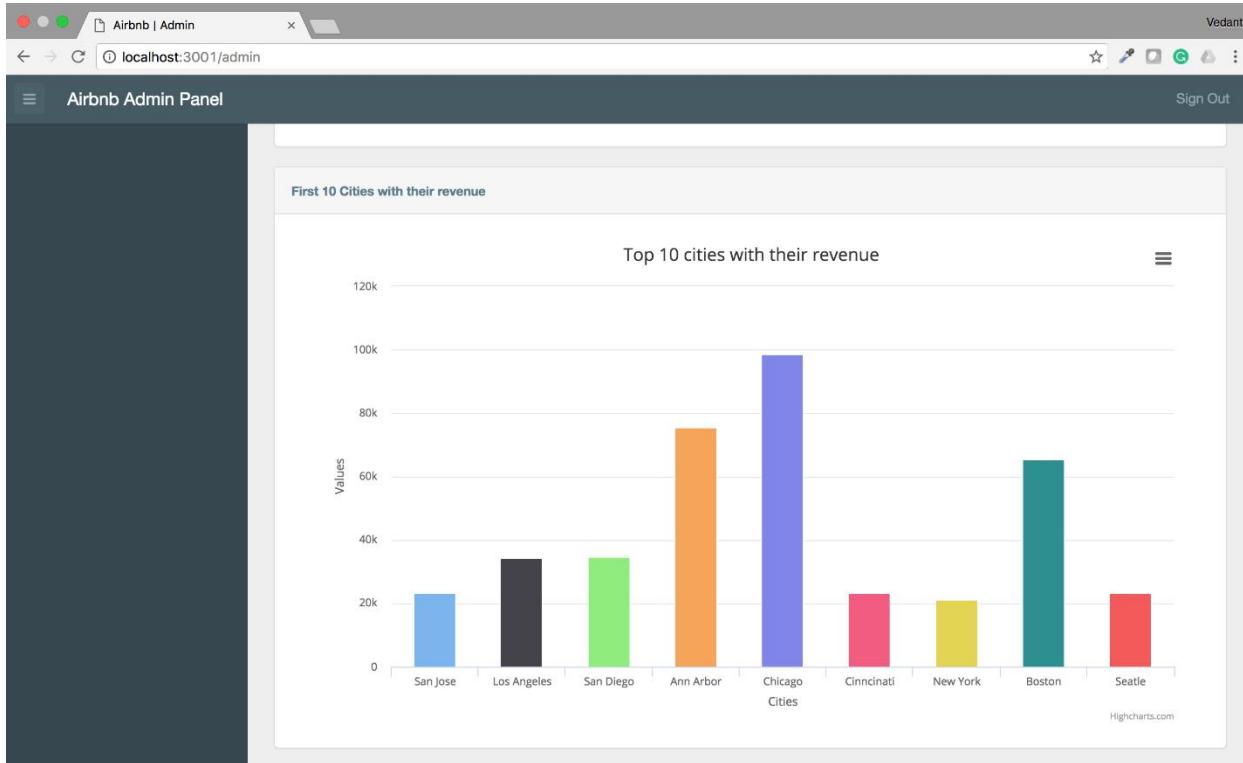
Property	Revenue (k)
1	4.5
2	3.8
3	4.8
4	5.2
5	5.5
6	3.2
7	4.2
8	4.0
9	3.5
10	3.8

The chart has a y-axis labeled "Values" with ticks at 4k, 6k, and 8k.

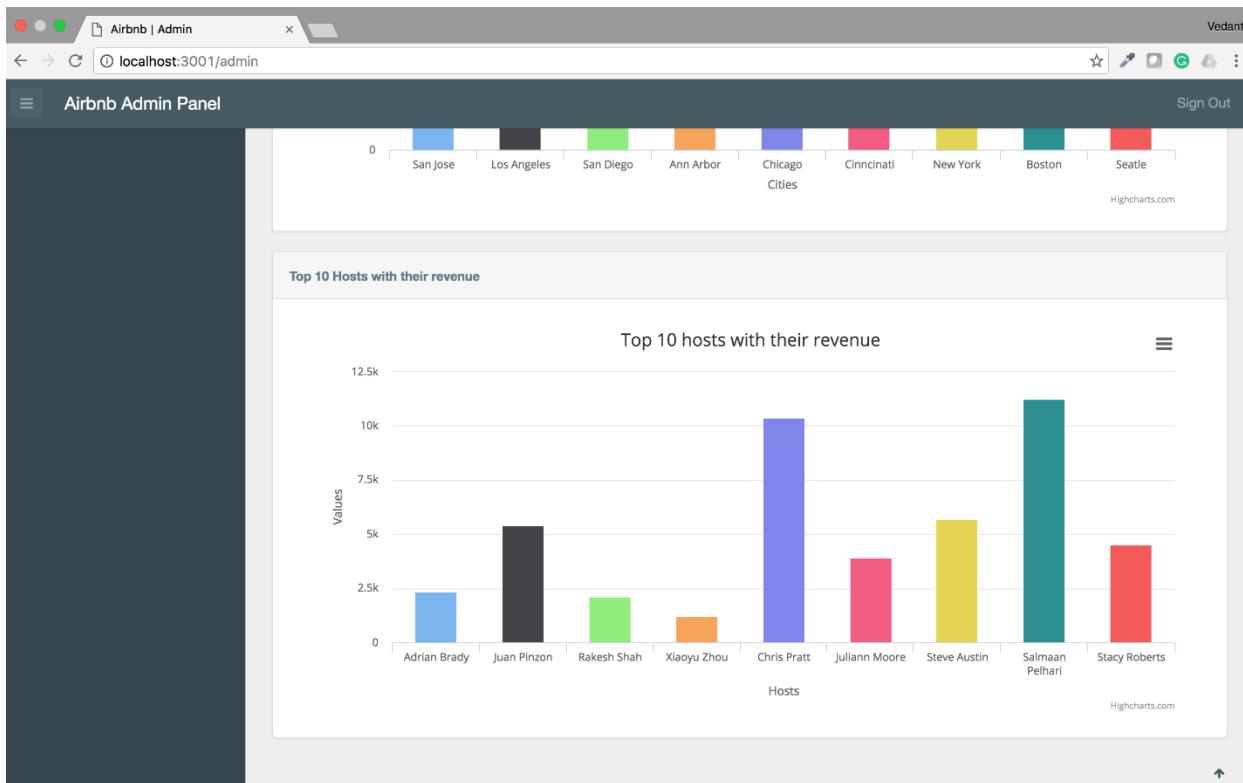
Top 10 properties with their revenue



First 10 cities with their revenue



Top 10 hosts with their revenue



List of all the active users

The screenshot shows the Airbnb Admin Panel User Management interface. The left sidebar has a dark theme with white icons and text. It includes links for Dashboard, User Management (which is currently selected), Active Host Management, Pending Host Requests, Active Property Management, Pending Property Requests, Invoice management, and Add New Admin. The main content area has a light gray background. At the top, it says "User Management: Active Users". Below that is a search bar with a magnifying glass icon and the placeholder "Search user...". A table titled "Manage Users" lists eight active users with columns for Name, City, State, and Action (with a "View" button). The data is as follows:

Name	City	State	Action
Alexandra Martin	North Cheryl	Montana	<button>View</button>
Andrea Mason	New Clifford	Pennsylvania	<button>View</button>
Andrew Rowland	Port Michael	Mississippi	<button>View</button>
Angela Burns	Port Lori	West Virginia	<button>View</button>
Ashley Roberts	South Monicamouth	Massachusetts	<button>View</button>
Austin Marshall	East Blaketown	Michigan	<button>View</button>
Beverly Harrison	Smithland	Louisiana	<button>View</button>
Brian Flovad	Martinshire	West Virainia	<button>View</button>

Users profile can be viewed by the admin

The screenshot shows the Airbnb Admin Panel interface. On the left is a sidebar with navigation links: Dashboard, User Management (which is active and highlighted in blue), Active Host Management, Pending Host Requests, Active Property Management, Pending Property Requests, Invoice management, and Add New Admin. The main content area has a header "User Management" and a sub-header "Active User Management". Below this is a table listing users with columns: Name, City, State, and Action (with a "View" button). A modal window titled "User Info" is open over the table, displaying detailed information for a user named Alexandra Martin. The modal content includes:

Name: Alexandra Martin
Email: rickymyers@yahoo.com
User Id: 5844e60eeb037a68e8bc9e0e
Address: 705 Anderson Mill Apt. 681
City: North Cheryl
State: Montana
Zip: 06625
Phone number: 01148559699
Created date: 12/04/2016 @ 7:59PM

At the bottom right of the modal is a red "Close" button.

Name	City	State	Action
Andrew Rowland	Port Michael	Mississippi	<button>View</button>
Angela Burns	Port Lori	West Virginia	<button>View</button>
Ashley Roberts	South Monicamouth	Massachusetts	<button>View</button>
Austin Marshall	East Blaketown	Michigan	<button>View</button>
Beverly Harrison	Smithland	Louisiana	<button>View</button>
Brian Floyd	Martinshire	West Virginia	<button>View</button>

User can be searched based on his Name, city or state(Area)

The screenshot shows the Airbnb Admin Panel's User Management section. The left sidebar has a dark theme with white text and icons. It includes links for Dashboard, User Management (which is currently selected and highlighted in blue), Active Host Management, Pending Host Requests, Active Property Management, Pending Property Requests, Invoice management, and Add New Admin. The main content area has a light gray background. At the top, it says "User Management: Active Users". Below that is a search bar with a magnifying glass icon and the placeholder text "Port L". Underneath the search bar is a table header with columns: "Name", "City", "State", and "Action". A single row of data is shown: "Angela Burns" in the Name column, "Port Lori" in the City column, "West Virginia" in the State column, and a green "View" button in the Action column. At the bottom of the main content area, it says "© 2016 AIRBNB".

List of all active hosts

The screenshot shows a web browser window titled "Host Management" with the URL "localhost:3001/admin_activeHostManagement". The page is part of the "Airbnb Admin Panel". On the left, there is a dark sidebar with the following navigation options:

- Welcome, Admin
- Dashboard
- User Management
- Active Host Management** (this option is highlighted)
- Pending Host Requests
- Active Property Management
- Pending Property Requests
- Invoice management
- Add New Admin

The main content area is titled "Host Management: Active Hosts" and contains a sub-section titled "Manage Users". It features a search bar with the placeholder "Search user..." and a table with the following data:

Name	City	State	Action
Aaron Jacobs	North Dustin	Washington	<button>View</button>
Aaron Barnes	East Nathan	Kansas	<button>View</button>
Aaron Castro	Shawnmouth	Delaware	<button>View</button>
Aaron Ortega	South Justin	Hawaii	<button>View</button>
Aaron Baker	East Betty	Kentucky	<button>View</button>
Aaron Washington	Jenniferfurt	Wisconsin	<button>View</button>
Aaron Williams	New Johnville	Louisiana	<button>View</button>

Host profile can be viewed by the admin

The screenshot shows the Airbnb Admin Panel interface. A modal window titled "Host Info" is open, displaying detailed information about a host named Aaron Jacobs. The modal contains the following data:

Name	Email	Host Id	Address	City	State	Zip	Phone number	Created date
Aaron Jacobs	svance@yahoo.com	584497d7eb037a2f308bf2b4	21253 Carolyn Center	North Dustin	Washington	79152	(939)803-5203x295	12/04/2016 @ 2:25PM

Below the modal, a table lists other hosts with their names, cities, states, and "View" buttons. The table data is as follows:

Name	City	State	Action
Aaron Castro	Shawnmouth	Delaware	<button>View</button>
Aaron Ortega	South Justin	Hawaii	<button>View</button>
Aaron Baker	East Betty	Kentucky	<button>View</button>
Aaron Washington	Jenniferfurt	Wisconsin	<button>View</button>
Aaron Williams	New Johnville	Louisiana	<button>View</button>

A red "Close" button is located at the bottom right of the modal.

Host can be searched by the admin based on his Name, city, state(area)

The screenshot shows the Airbnb Admin Panel interface. A search bar at the top of the main content area is populated with the text "South". Below the search bar, a table titled "Manage Users" displays a list of hosts matching the search term. The table has columns for Name, City, State, and Action. The data in the table is as follows:

Name	City	State	Action
Aaron Ortega	South Justin	Hawaii	<button>View</button>
Aaron Fitzgerald	Lake Jennifer	South Carolina	<button>View</button>
Aaron Alexander	South Sharon	Rhode Island	<button>View</button>
Abigail McNeil	Moranshire	South Carolina	<button>View</button>
Abigail Cruz	South Gregory	Montana	<button>View</button>
Adam Walker	South Samantha	Wisconsin	<button>View</button>
Adam Reynolds	South Lisa	New Mexico	<button>View</button>

List of all pending hosts can be seen by the users

The screenshot shows a web browser window titled "Host Management" with the URL "localhost:3001/admin_pendingHost". The page is part of the "Airbnb Admin Panel". On the left, there is a sidebar with a user profile picture and the text "Welcome, Admin". The sidebar includes links for "Dashboard", "User Management", "Active Host Management", "Pending Host Requests" (which is currently selected), "Active Property Management", "Pending Property Requests", "Invoice management", and "Add New Admin". The main content area is titled "Host Management: Pending Host Requests" and contains a table titled "Manage Users". The table has columns for "Id", "Name", "City", "State", and "Action". Each row represents a pending host request with the following data:

ID	Name	City	State	Action
5844e63ceb037a679c657561	Aaron Rubio	New Melindahaven	California	<button>View</button> <button>Approve</button>
5844e63aeb037a679c657557	Adam Martinez	Lanehaven	Kentucky	<button>View</button> <button>Approve</button>
5844e63aeb037a679c657556	Albert Hines	New Ashley	California	<button>View</button> <button>Approve</button>
5844e63feb037a679c657573	April Morton	West Wendy	Massachusetts	<button>View</button> <button>Approve</button>
5844e639eb037a679c657552	Ashley Weaver	South Jennifer	Vermont	<button>View</button> <button>Approve</button>
5844e638eb037a679c65754d	Barbara Wright	Justinport	Rhode Island	<button>View</button> <button>Approve</button>
5844e63aeb037a679c657559	Bianca Sullivan	Baileyfort	Oregon	<button>View</button> <button>Approve</button>

Profile of pending hosts can be viewed by the admin

The screenshot shows the Airbnb Admin Panel interface. On the left is a sidebar with navigation links: Dashboard, User Management, Active Host Management, Pending Host Requests (which is currently selected), Active Property Management, Pending Property Requests, Invoice management, and Add New Admin. The main content area is titled "Host Management" and shows a table of pending host requests. A modal window titled "Pending Host Info" is open over the table, displaying the details for a host named Adam Martinez. The modal contains the following information:

Name	Email	Host Id	Address	City	State	Zip	Phone number	Created date
Adam Martinez	leuisharold@yahoo.com	5844e63aeb037a679c657557	5822 Alexa Avenue Apt. 409	Lanehaven	Kentucky	35047	760.964.0912	12/04/2016 @ 7:59PM

At the bottom right of the modal is a red "Close" button.

Host can be approved by the admin

The screenshot shows the same Airbnb Admin Panel interface as the previous one, but the table in the main content area now shows the host request for Adam Martinez with an "Approved" status. The "Action" column for his row now contains blue "View" and green "Approved" buttons, indicating the host has been approved. The other host entries remain in the "Pending" state with their original "View" and "Approve" buttons.

ID	Name	City	State	Action
5844e63aeb037a679c657557	Adam Martinez	Lanehaven	Kentucky	View Approved
5844e63aeb037a679c657556	Albert Hines	New Ashley	California	View Approve
5844e63feb037a679c657573	April Morton	West Wendy	Massachusetts	View Approve
5844e639eb037a679c657552	Ashley Weaver	South Jennifer	Vermont	View Approve
5844e638eb037a679c65754d	Barbara Wright	Justinport	Rhode Island	View Approve
5844e63aeb037a679c657559	Bianca Sullivan	Baileyfort	Oregon	View Approve
5844e63ceb037a679c657563	Charles Watson	New Jasonville	New Mexico	View Approve

List of all active properties can be seen by the admin

The screenshot shows the Airbnb Admin Panel interface. On the left, there is a sidebar with various administrative options: Dashboard, User Management, Active Host Management, Pending Host Requests, Active Property Management (which is currently selected), Pending Property Requests, Invoice management, and Add New Admin. The main content area is titled "Property Management: Active Property". It features a search bar at the top. Below the search bar is a table titled "Manage Properties" with columns for Name, City, State, and Action. The table lists several properties, each with a "View" button. The properties listed are: "Quidem illo nostrum aliquam esse.", "Vero dicta veritatis vero natus in.", "Incidunt laudantium aspernatur fuga alias neque.", "Corrupti ipsam doloremque quidem.", "Sapiente exercitationem libero sed ex.", "Sequi eius quas praesentium autem velit vel.", and "Aliquam repudiandae quis qui quo impedit.". All properties are located in San Jose, CA.

Name	City	State	Action
Quidem illo nostrum aliquam esse.	San Jose	CA	<button>View</button>
Vero dicta veritatis vero natus in.	San Jose	CA	<button>View</button>
Incidunt laudantium aspernatur fuga alias neque.	San Jose	CA	<button>View</button>
Corrupti ipsam doloremque quidem.	San Jose	CA	<button>View</button>
Sapiente exercitationem libero sed ex.	San Jose	CA	<button>View</button>
Sequi eius quas praesentium autem velit vel.	San Jose	CA	<button>View</button>
Aliquam repudiandae quis qui quo impedit.	San Jose	CA	<button>View</button>

Property details can be viewed by the admin

The screenshot shows the Airbnb Admin Panel interface. The sidebar is identical to the previous screenshot, showing the Active Property Management section is selected. A modal window titled "Property Info" is open in the center. Inside the modal, detailed information about a specific property is displayed, including its name, ID, category, address, city, state, country, zip code, maximum guests, number of bedrooms, number of bathrooms, description, price, and creation date. The property is described as a "Shared room" located at 01702 Ray Manors Apt. 809 in San Jose, CA, United States, with a zip code of 56948, a max guest of 1, 5 bedrooms, and 9 bathrooms. The description states: "Rem aspernatur necessitatibus ipsam voluptas magni." The price is listed as 358, and it was created on 04/12/2016. At the bottom right of the modal, there is a red "Close" button.

Property can be searched by the admin based on the properties Name, city, state(area)

The screenshot shows the Airbnb Admin Panel interface. The left sidebar has a dark theme with white text and icons. It includes links for Dashboard, User Management, Active Host Management, Pending Host Requests, Active Property Management (which is highlighted in blue), Pending Property Requests, Invoice management, and Add New Admin. The main content area has a light gray background. At the top, it says "Property Management: Active Property". Below that is a search bar with the placeholder "Corru". A table titled "Manage Properties" lists two entries:

Name	City	State	Action
Quidem illo nostrum aliquam esse.	San Jose	CA	<button>View</button>
Corrupti ipsam doloremque quidem.	San Jose	CA	<button>View</button>

At the bottom left of the main area, it says "© 2016 AIRBNB". The top of the browser window shows "Property Management" in the tab, the address "localhost:3001/admin_activePropertyManagement", and the user "Vedant". There are also standard browser control buttons like back, forward, and refresh.

List of all pending properties can be viewed by the admin

Pending properties can be approved by the admin

The screenshot shows a web-based administration interface titled "Airbnb Admin Panel". The left sidebar contains a user profile icon and the text "Welcome, Admin". Below the profile are several menu items: Dashboard, User Management, Active Host Management, Pending Host Requests, Active Property Management, **Pending Property Requests** (which is currently selected), Invoice management, and Add New Admin. The main content area is titled "Property Management: Pending Property Requests" and features a sub-header "Manage Properties". It includes a search bar with the placeholder "Search user...". A table lists seven pending property requests, each with columns for Name, City, State, and Action. The "Action" column contains two buttons: "View" (blue) and "Approve" (green). The table rows are as follows:

Name	City	State	Action
Distinctio sit.	San Jose	CA	View Approve
Accusamus quo a.	San Jose	CA	View Approve
Ea harum unde.	San Jose	CA	View Approve
Sunt eos libero.	San Jose	CA	View Approve
Minima deleniti.	San Jose	CA	View Approve
Placeat quo.	San Jose	CA	View Approve
Recusandae accusamus.	San Jose	CA	View Approve

Property details can be viewed by the admin

The screenshot shows a web browser window titled "Property Management" with the URL "localhost:3001/admin_pendingProperty". The main interface is the "Airbnb Admin Panel" with a dark sidebar menu. A modal window titled "Pending Property Info" is open, displaying detailed information about a property. The property details include:

- Name: Ea harum unde.
- Property Id: 5844e5a8eb037a3cccb69239
- Host Id: 58449c4ceb037a50f086fb09
- Category: Private room
- Address: 9748 Navarro Radial Apt. 186
- City: San Jose
- State: CA
- Country: United States
- Zip: 66135
- Max Guest: 2
- Bedrooms: 2
- Bathrooms: 9
- Description: Beatae esse amet.
- Price: 324
- Created date: 12/04/2016 @ 7:57PM

Below the modal, there is a table with two columns: "Action" and "View" and "Approve" buttons for each row. A red "Close" button is located at the bottom right of the modal. The footer of the page shows "Recusandae accusamus.", "San Jose", "CA", and another set of "View" and "Approve" buttons.

List of all the bills can be viewed by the admin

The screenshot shows a web browser window titled "Active Property Management" with the URL "localhost:3001/admin_invoiceManagement". The main interface is the "Airbnb Admin Panel" with a dark sidebar menu. The "Invoice management" section is active. The "Manage Invoices" page displays a table of invoices with the following data:

ID	Name	Date created	Action
5844d92feb037a2c88f1715f	Gail Le	Sun Dec 04 2016	<button>View</button> <button>Delete</button>
5844d92feb037a2c88f17161	Larry Lowery	Sun Dec 04 2016	<button>View</button> <button>Delete</button>
5844d930eb037a2c88f17163	Matthew Holmes	Sun Dec 04 2016	<button>View</button> <button>Delete</button>
5844d931eb037a2c88f17165	Miranda White	Sun Dec 04 2016	<button>View</button> <button>Delete</button>
5844d931eb037a2c88f17167	Gail Le	Sun Dec 04 2016	<button>View</button> <button>Delete</button>
5844d932eb037a2c88f17169	Jesus Mccoy	Sun Dec 04 2016	<button>View</button> <button>Delete</button>
5844d932eb037a2c88f1716b	Bailey Jones	Sun Dec 04 2016	<button>View</button> <button>Delete</button>

Details of all the bills can be viewed by the admin

The screenshot shows the Airbnb Admin Panel interface. On the left, there's a sidebar with various management options like Dashboard, User Management, Active Host Management, Pending Host Requests, Active Property Management, Pending Property Requests, and Invoice management. The Invoice management section is currently selected. In the main area, there's a modal window titled "Invoice info" displaying details for a specific invoice. The modal contains the following information:

- Billing Id: 804-26-0228
- Date created: 12/04/2016 @ 7:04PM
- Property Name: Corrupti ipsum doloremque quidem.
- Address: 93507 Jennifer Ridge Suite 199, San Jose, CA-18662
- Host Name: Larry Lowery
- Customer Name: Ronald Travis
- Total Charge: 317

At the bottom right of the modal is a "Close" button. Below the modal, there's a table listing multiple invoices with columns for Id, Name, Date created, and Action (View, Delete). The table data is as follows:

Id	Name	Date created	Action
5844d92feb037a2c88f17161	Larry Lowery	Sun Dec 04 2016	<button>View</button> <button>Delete</button>
5844d930eb037a2c88f17163	Matthew Holmes	Sun Dec 04 2016	<button>View</button> <button>Delete</button>
5844d931eb037a2c88f17165	Miranda White	Sun Dec 04 2016	<button>View</button> <button>Delete</button>
5844d931eb037a2c88f17167	Gail Le	Sun Dec 04 2016	<button>View</button> <button>Delete</button>
5844d932eb037a2c88f17169	Jesus Mccoy	Sun Dec 04 2016	<button>View</button> <button>Delete</button>
5844d932eb037a2c88f1716b	Bailey Jones	Sun Dec 04 2016	<button>View</button> <button>Delete</button>

Bills can be deleted by the admin

The screenshot shows the Airbnb Admin Panel interface, specifically the "Manage Invoices" page under the Invoice management section. The sidebar on the left is identical to the previous screenshot. The main area features a search bar at the top labeled "Search user...". Below it is a table listing invoices with columns for Id, Name, Date created, and Action (View, Delete). The table data is as follows:

Id	Name	Date created	Action
5844d92feb037a2c88f17161	Larry Lowery	Sun Dec 04 2016	<button>View</button> <button>Delete</button>
5844d930eb037a2c88f17163	Matthew Holmes	Sun Dec 04 2016	<button>View</button> <button>Delete</button>
5844d931eb037a2c88f17165	Miranda White	Sun Dec 04 2016	<button>View</button> <button>Delete</button>
5844d931eb037a2c88f17167	Gail Le	Sun Dec 04 2016	<button>View</button> <button>Delete</button>
5844d932eb037a2c88f17169	Jesus Mccoy	Sun Dec 04 2016	<button>View</button> <button>Delete</button>
5844d932eb037a2c88f1716b	Bailey Jones	Sun Dec 04 2016	<button>View</button> <button>Delete</button>
5844d933eb037a2c88f1716d	Shannon Collins	Sun Dec 04 2016	<button>View</button> <button>Delete</button>

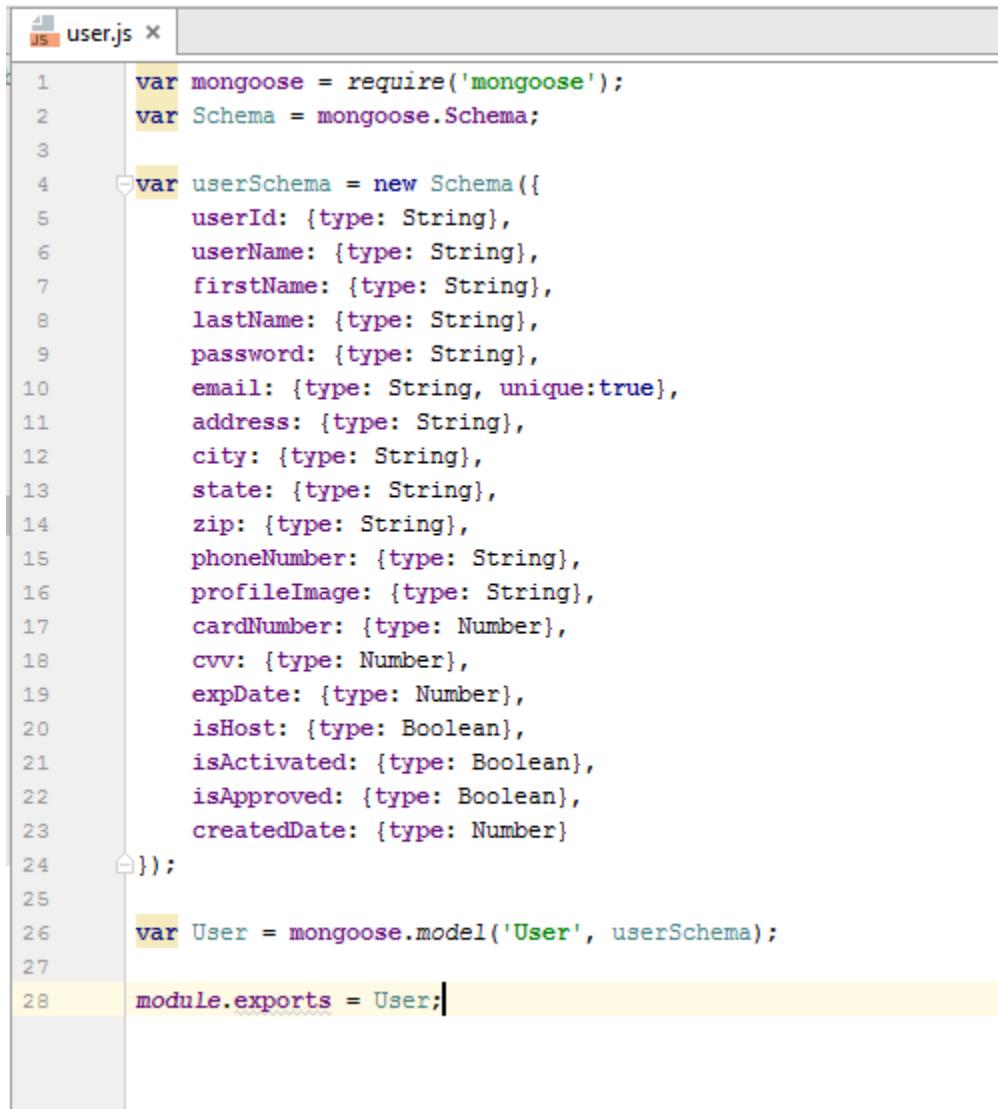
New admin can be added by the admin

The screenshot shows a web browser window with the Airbnb logo at the top. The title bar says "Add Admin". The address bar shows "localhost:3001/viewAddAdminPage". The main content area has a light gray background with a white input form. The form is titled "Add New Admin" in bold black text. It contains nine input fields, each with a placeholder label: "Email", "Password", "First Name", "Last Name", "Address", "City", "State", "Zip Code", and "Phone number". Below the input fields are two buttons: a red "Add Admin" button on the left and a blue "Back" button on the right. The overall design is clean and modern, mimicking the look of the Airbnb website.

Database Design

MongoDB

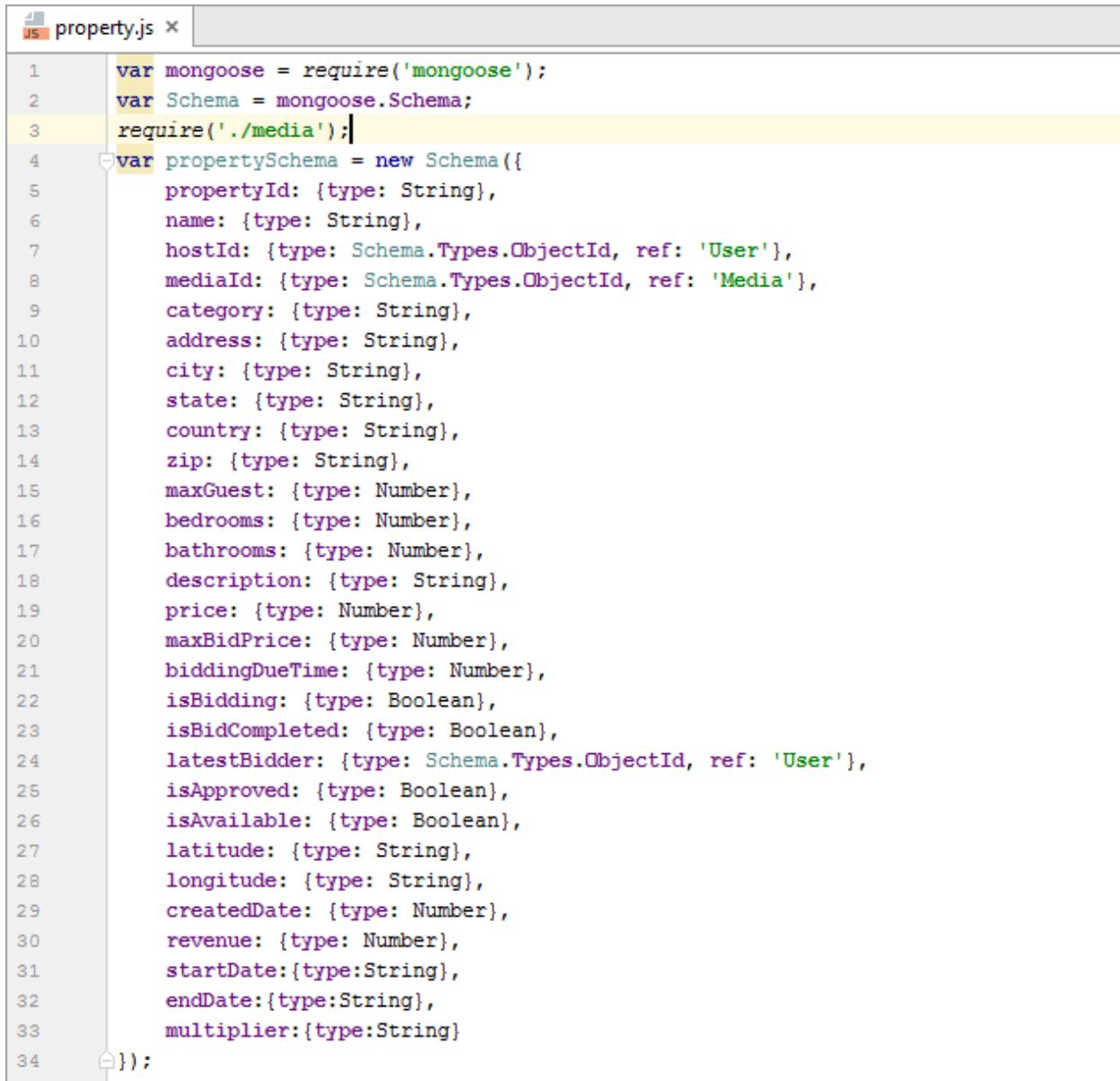
1) User Collection



```
1  var mongoose = require('mongoose');
2  var Schema = mongoose.Schema;
3
4  var userSchema = new Schema({
5      userId: {type: String},
6      userName: {type: String},
7      firstName: {type: String},
8      lastName: {type: String},
9      password: {type: String},
10     email: {type: String, unique:true},
11     address: {type: String},
12     city: {type: String},
13     state: {type: String},
14     zip: {type: String},
15     phoneNumber: {type: String},
16     profileImage: {type: String},
17     cardNumber: {type: Number},
18     cvv: {type: Number},
19     expDate: {type: Number},
20     isHost: {type: Boolean},
21     isActivated: {type: Boolean},
22     isApproved: {type: Boolean},
23     createdDate: {type: Number}
24 });
25
26 var User = mongoose.model('User', userSchema);
27
28 module.exports = User;
```

Key	Value	Type
> (1) ObjectId("583d4d177828aa2f286c396d")	{ 16 fields }	Object
> (2) ObjectId("583fa6ee493c3723184332d8")	{ 11 fields }	Object
> (3) ObjectId("583faac0493c3723184332db")	{ 12 fields }	Object
> (4) ObjectId("583f6dae3141c14bf8941ec2")	{ 16 fields }	Object
> (5) ObjectId("584278d65e4a293614fa8c11")	{ 16 fields }	Object
_id	ObjectId("584278d65e4a293614fa8c11")	ObjectId
isActivated	true	Boolean
isHost	true	Boolean
isApproved	false	Boolean
createdDate	1480751318781.0	Double
userId	574-24-2721	String
password	\$2a\$10\$vuPcHgKxx68EdSAImNdWQu3oSMSYbCjsAXEOJQcoglvt...	String
email	shrey@gmail.com	String
lastName	Bhatt	String
firstName	Shrey	String
__v	0	Int32
address	101 E San Fernando St.	String
city	San Jose	String
state	CA	String
zip	95112	String
profileImage	584278d65e4a293614fa8c11.png	String
> (6) ObjectId("58427bcc4b9db0dc8575bfe")	{ 11 fields }	Object
> (7) ObjectId("5842835a77ea9c36acaf0cfe")	{ 16 fields }	Object
> (8) ObjectId("5842847d77ea9c36acaf0cf")	{ 16 fields }	Object
> (9) ObjectId("5842853377ea9c36acaf0d03")	{ 16 fields }	Object
> (10) ObjectId("584286677ea9c36acaf0d04")	{ 16 fields }	Object
> (11) ObjectId("584286aa77ea9c36acaf0d05")	{ 11 fields }	Object

2)Property Collection

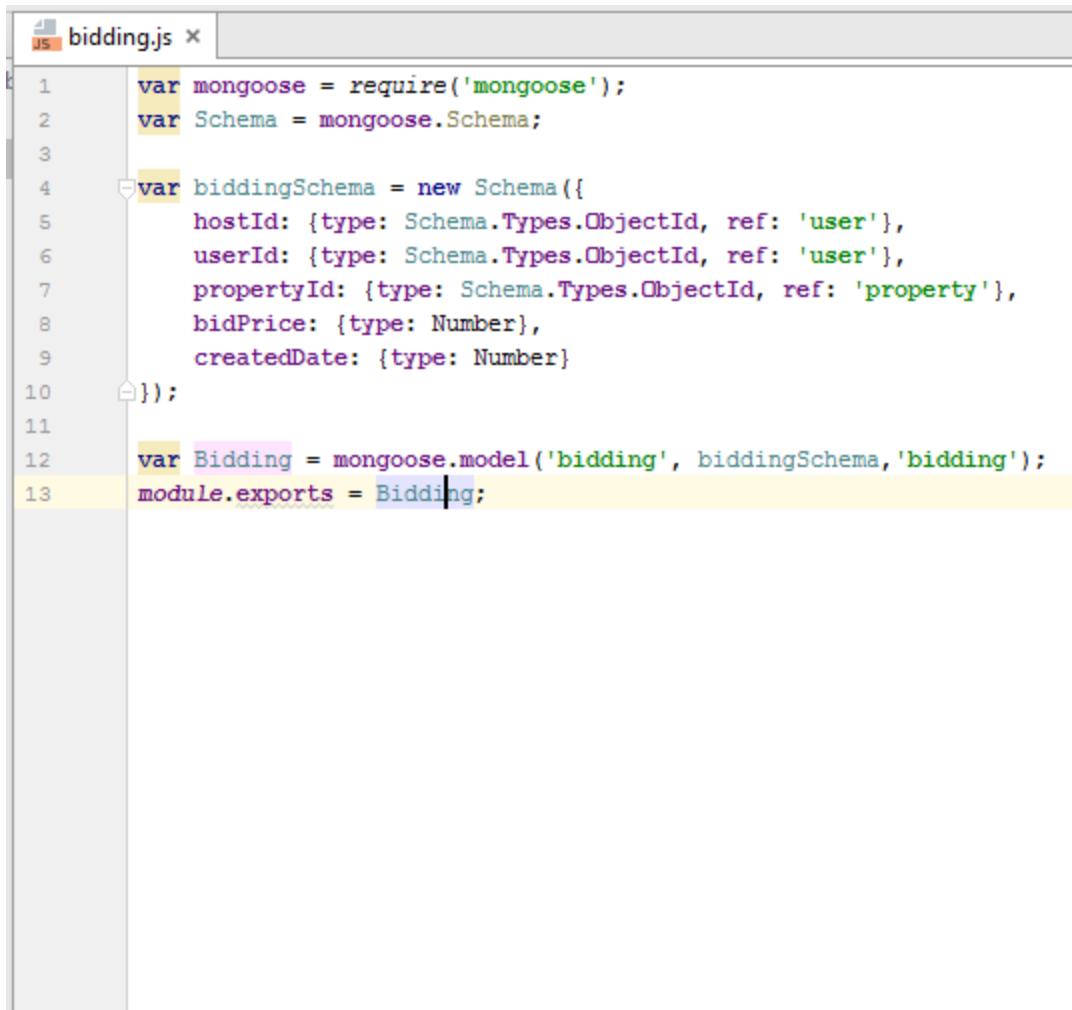


```
1 var mongoose = require('mongoose');
2 var Schema = mongoose.Schema;
3 require('./media');
4 var propertySchema = new Schema({
5   propertyId: {type: String},
6   name: {type: String},
7   hostId: {type: Schema.Types.ObjectId, ref: 'User'},
8   mediaId: {type: Schema.Types.ObjectId, ref: 'Media'},
9   category: {type: String},
10  address: {type: String},
11  city: {type: String},
12  state: {type: String},
13  country: {type: String},
14  zip: {type: String},
15  maxGuest: {type: Number},
16  bedrooms: {type: Number},
17  bathrooms: {type: Number},
18  description: {type: String},
19  price: {type: Number},
20  maxBidPrice: {type: Number},
21  biddingDueTime: {type: Number},
22  isBidding: {type: Boolean},
23  isBidCompleted: {type: Boolean},
24  latestBidder: {type: Schema.Types.ObjectId, ref: 'User'},
25  isApproved: {type: Boolean},
26  isAvailable: {type: Boolean},
27  latitude: {type: String},
28  longitude: {type: String},
29  createdDate: {type: Number},
30  revenue: {type: Number},
31  startDate:{type:String},
32  endDate:{type:String},
33  multiplier:{type:String}
34 });

```

db.getCollection('properties').find()		
Key	Value	Type
✓ (1) ObjectId("583e8f508683e11820c158b4")	{ 23 fields }	Object
_id	ObjectId("583e8f508683e11820c158b4")	ObjectId
mediadl	ObjectId("583e8f518683e11820c158b5")	ObjectId
isAvailable	false	Boolean
isApproved	true	Boolean
createdDate	1480494928.49	Double
longitude	-121.93304510000002	String
latitude	37.3568721	String
price	100	Int32
description	Listing Video	String
name	Listing Video Test1	String
bathrooms	1.5	Double
bedrooms	2	Int32
zip	95112	String
address	1230 Coleman Avenue	String
state	CA	String
city	Santa Clara	String
country	United States	String
category	Private room	String
maxGuest	3	Int32
hostId	ObjectId("583d4d177828aa2f286c396d")	ObjectId
propertyId	620-22-9290	String
revenue	800	Int32
_v	0	Int32
> (2) ObjectId("583fa95f493c3723184332d9")	{ 23 fields }	Object
> (3) ObjectId("5840994906dd507c46d58f9")	{ 27 fields }	Object
> (4) ObjectId("5840a80856568023c4b76813")	{ 24 fields }	Object

3)Bidding Collection

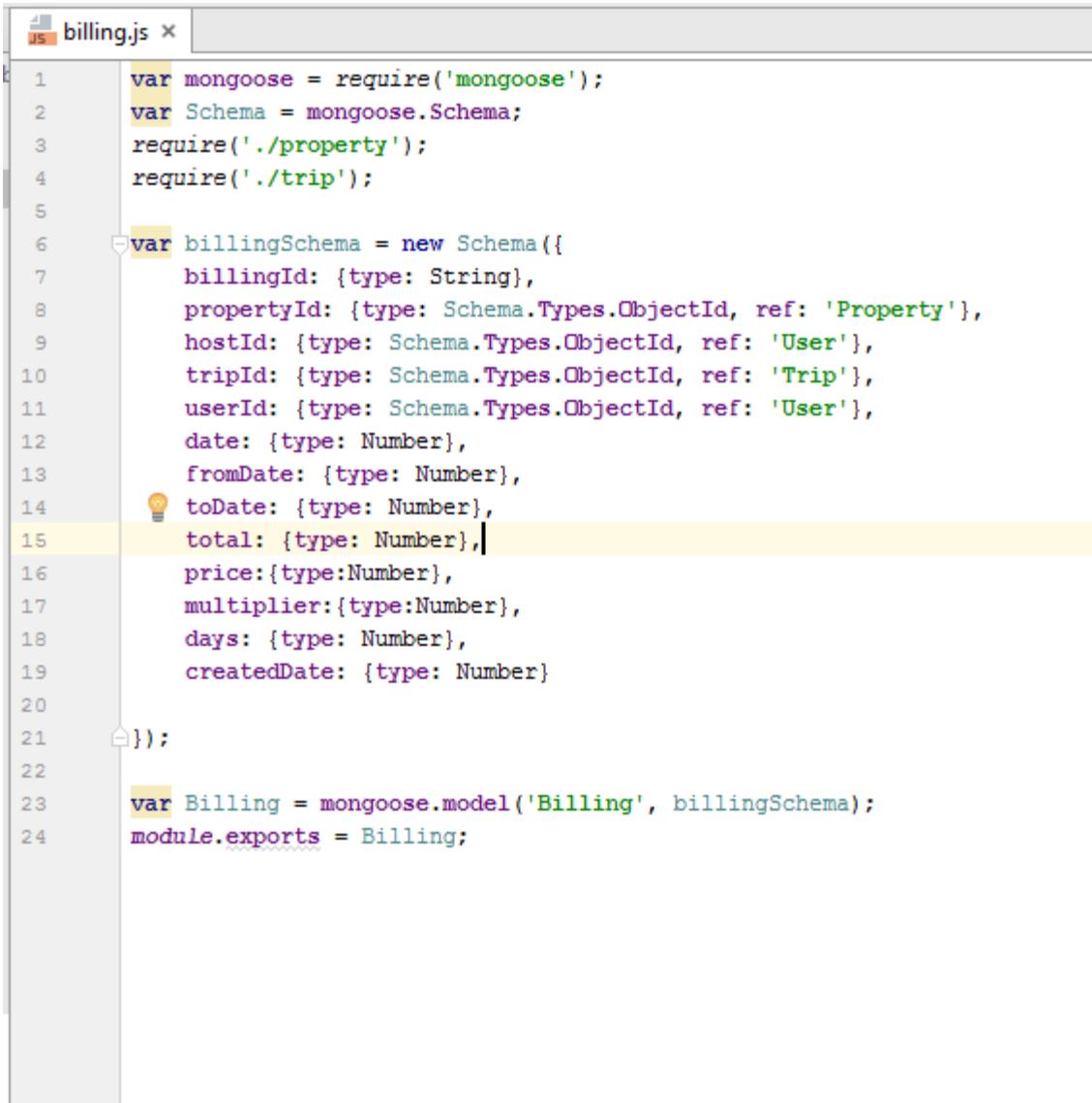


The screenshot shows a code editor window with a tab labeled "bidding.js". The file contains the following JavaScript code:

```
1 var mongoose = require('mongoose');
2 var Schema = mongoose.Schema;
3
4 var biddingSchema = new Schema({
5     hostId: {type: Schema.Types.ObjectId, ref: 'user'},
6     userId: {type: Schema.Types.ObjectId, ref: 'user'},
7     propertyId: {type: Schema.Types.ObjectId, ref: 'property'},
8     bidPrice: {type: Number},
9     createdDate: {type: Number}
10});
11
12 var Bidding = mongoose.model('bidding', biddingSchema, 'bidding');
13 module.exports = Bidding;
```

Key	Value	Type
✓ (1) ObjectId("583ff50912b81b26305e56af")	{ 6 fields }	Object
└ _id	ObjectId("583ff50912b81b26305e56af")	ObjectId
└ hostId	ObjectId("583d4d177828aa2f286c396d")	ObjectId
└ createdDate	1480586505168.0	Double
└ bidPrice	1001	Int32
└ propertyId	ObjectId("583e4186115b4227a87672cd")	ObjectId
└ _v	0	Int32
✓ (2) ObjectId("583ffa5512b81b26305e56b0")	{ 6 fields }	Object
└ _id	ObjectId("583ffa5512b81b26305e56b0")	ObjectId
└ hostId	ObjectId("583d4d177828aa2f286c396d")	ObjectId
└ createdDate	148057861607.0	Double
└ bidPrice	2001	Int32
└ propertyId	ObjectId("583e4186115b4227a87672cd")	ObjectId
└ _v	0	Int32
✓ (3) ObjectId("58409419633c172848320b2a")	{ 6 fields }	Object
└ _id	ObjectId("58409419633c172848320b2a")	ObjectId
└ hostId	ObjectId("583d4d177828aa2f286c396d")	ObjectId
└ createdDate	1480627225486.0	Double
└ bidPrice	300	Int32
└ propertyId	ObjectId("583e4186115b4227a87672cd")	ObjectId
└ _v	0	Int32
> (4) ObjectId("58409420633c172848320b2b")	{ 6 fields }	Object
> (5) ObjectId("58409421633c172848320b2c")	{ 6 fields }	Object
> (6) ObjectId("58409421633c172848320b2d")	{ 6 fields }	Object
> (7) ObjectId("5840943a633c172848320b2e")	{ 6 fields }	Object
> (8) ObjectId("5840f98c0bc0b355c3cc47a")	{ 6 fields }	Object
> (9) ObjectId("5840fae132f24943c87e458a")	{ 6 fields }	Object

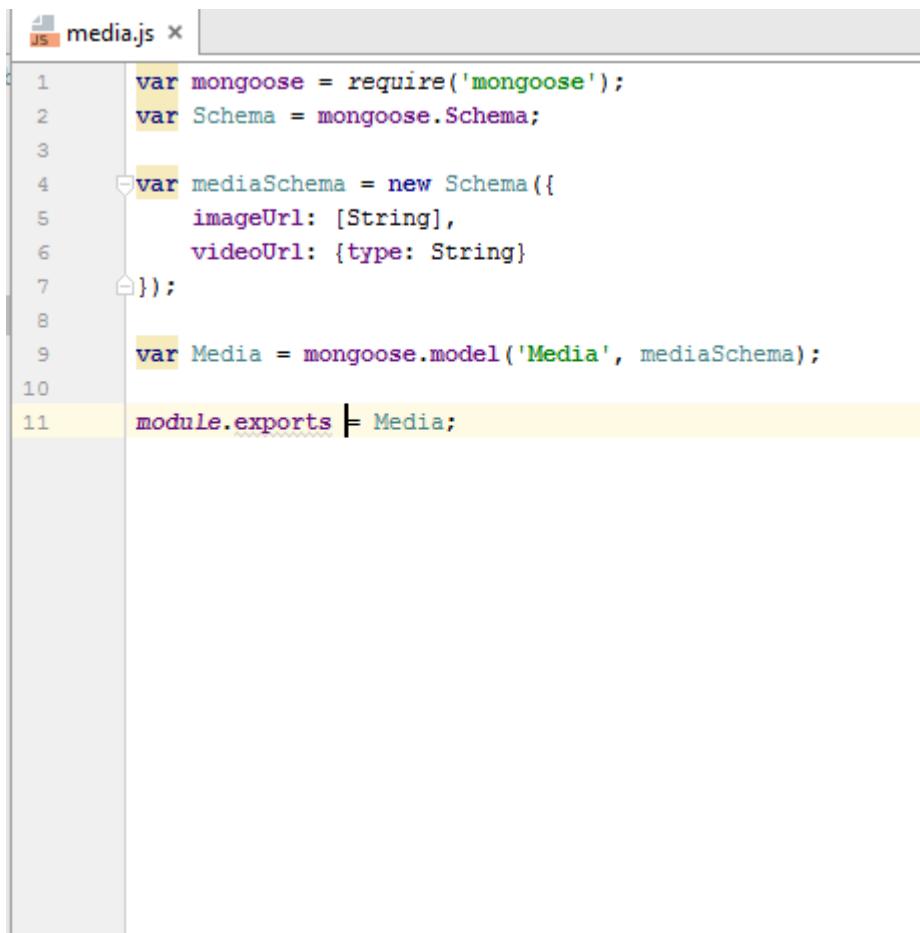
4) Billing Collection



```
1 var mongoose = require('mongoose');
2 var Schema = mongoose.Schema;
3 require('./property');
4 require('./trip');
5
6 var billingSchema = new Schema({
7   billingId: {type: String},
8   propertyId: {type: Schema.Types.ObjectId, ref: 'Property'},
9   hostId: {type: Schema.Types.ObjectId, ref: 'User'},
10  tripId: {type: Schema.Types.ObjectId, ref: 'Trip'},
11  userId: {type: Schema.Types.ObjectId, ref: 'User'},
12  date: {type: Number},
13  fromDate: {type: Number},
14  toDate: {type: Number},
15  total: {type: Number},| // This line is highlighted with a yellow background
16  price:{type:Number},
17  multiplier:{type:Number},
18  days: {type: Number},
19  createdDate: {type: Number}
20 });
21
22 var Billing = mongoose.model('Billing', billingSchema);
23 module.exports = Billing;
```

db.getCollection('billings').find({})		
Key	Value	Type
✓ (1) ObjectId("583f890489d5f560bc6bed6d")	{ 13 fields }	Object
_id	ObjectId("583f890489d5f560bc6bed6d")	ObjectId
billingId	151-28-5121	String
createdDate	1480558852820.0	Double
days	2	Int32
total	200	Int32
toDate	1480752000000.0	Double
fromDate	1480579200000.0	Double
date	1480558852819.0	Double
userId	ObjectId("583f6dae3141c14bf8941ec2")	ObjectId
tripId	ObjectId("583f89489d5f560bc6bed6c")	ObjectId
hostId	ObjectId("583d4d177828aa2f286c396d")	ObjectId
propertyId	ObjectId("583e0f508683e11820c158b4")	ObjectId
_v	0	Int32
✓ (2) ObjectId("583fad74493c3723184332dd")	{ 13 fields }	Object
_id	ObjectId("583fad74493c3723184332dd")	ObjectId
billingId	222-76-4580	String
createdDate	1480568180072.0	Double
days	2	Int32
total	400	Int32
toDate	1480665600000.0	Double
fromDate	1480492800000.0	Double
date	1480568180072.0	Double
userId	ObjectId("583faac0493c3723184332db")	ObjectId
tripId	ObjectId("583fad31493c3723184332dc")	ObjectId
hostId	ObjectId("583faee493c3723184332d8")	ObjectId
propertyId	ObjectId("583fa95f493c3723184332d9")	ObjectId

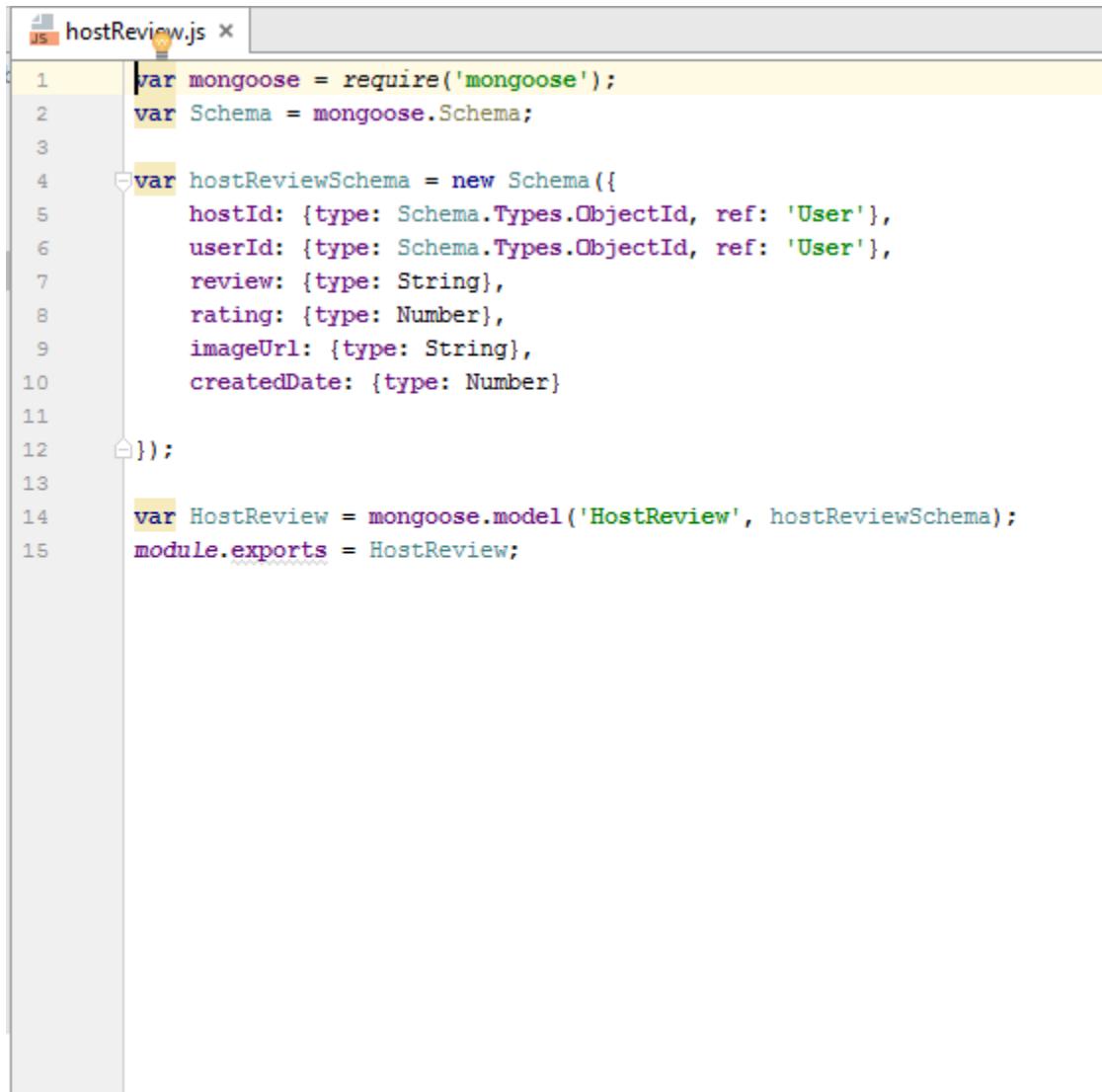
5) Media Collection



```
media.js x
1  var mongoose = require('mongoose');
2  var Schema = mongoose.Schema;
3
4  var mediaSchema = new Schema({
5      imageUrl: [String],
6      videoUrl: {type: String}
7  });
8
9  var Media = mongoose.model('Media', mediaSchema);
10
11 module.exports = Media;
```

db.getCollection('media').find({})		
Key	Value	Type
✓ media	0.087 sec.	
(1) ObjectId("583d515dd3010925608b2c4f")	{ 3 fields }	Object
_id	ObjectId("583d515dd3010925608b2c4f")	ObjectId
imageUrl	[1 element]	Array
0	0	Int32
__v	0	Int32
(2) ObjectId("583e1c01da9d0328cce90cd3")	{ 3 fields }	Object
_id	ObjectId("583e1c01da9d0328cce90cd3")	ObjectId
imageUrl	[1 element]	Array
0	0	Int32
__v	0	Int32
(3) ObjectId("583e27715179251de0479cb9")	{ 3 fields }	Object
(4) ObjectId("583e339859671c0b4cd9a69a")	{ 3 fields }	Object
(5) ObjectId("583e3617e280aa2ab47873de")	{ 3 fields }	Object
(6) ObjectId("583e38c81796481dd43f3b38")	{ 3 fields }	Object
(7) ObjectId("583e3a121796481dd43f3b3a")	{ 3 fields }	Object
(8) ObjectId("583e3bb2e280aa2ab47873e0")	{ 3 fields }	Object
(9) ObjectId("583e3fb9e280aa2ab47873e2")	{ 3 fields }	Object
(10) ObjectId("583e4186115b4227a87672ce")	{ 3 fields }	Object
(11) ObjectId("583e833575296250d44ab2e3")	{ 3 fields }	Object
_id	ObjectId("583e833575296250d44ab2e3")	ObjectId
imageUrl	[1 element]	Array
0	0	Int32
__v	0	Int32
(12) ObjectId("583e8f518683e11820c158b5")	{ 4 fields }	Object
(13) ObjectId("583fa95f493c3723184332da")	{ 4 fields }	Object
(14) ObjectId("583ff0abee4c4f46a4dae6d0")	{ 4 fields }	Object
_id	ObjectId("583ff0abee4c4f46a4dae6d0")	ObjectId
videoUrl	1480585358238583d4d17782aa2f286c396d.mp4	String
imageUrl	[3 elements]	Array
0	0	Int32
__v	0	Int32

6) Host Review Collection



A screenshot of a code editor window titled "hostReview.js". The code defines a schema for a "HostReview" document using the Mongoose library. The schema includes fields for hostId, userId, review, rating, imageUrl, and createdDate, each with specific data types and references to the "User" model.

```
hostReview.js
1 var mongoose = require('mongoose');
2 var Schema = mongoose.Schema;
3
4 var hostReviewSchema = new Schema({
5     hostId: {type: Schema.Types.ObjectId, ref: 'User'},
6     userId: {type: Schema.Types.ObjectId, ref: 'User'},
7     review: {type: String},
8     rating: {type: Number},
9     imageUrl: {type: String},
10    createdDate: {type: Number}
11 });
12
13 var HostReview = mongoose.model('HostReview', hostReviewSchema);
14 module.exports = HostReview;
```

db.getCollection('hostreviews').find({})

Key	Value	Type
1 ObjectId("583fae8e493c3723184332e1")	{ 7 fields }	Object
_id	ObjectId("583fae8e493c3723184332e1")	ObjectId
createdDate	1480568462081.0	Double
rating	3	Int32
review	This is test review from user to host about the trip. Lets see how it ...	String
hostId	ObjectId("583fa6ee493c3723184332d8")	ObjectId
userId	ObjectId("583faac0493c3723184332db")	ObjectId
__v	0	Int32
2 ObjectId("583fb0bc15651d44109c78e9")	{ 7 fields }	Object
_id	ObjectId("583fb0bc15651d44109c78e9")	ObjectId
createdDate	1480569020588.0	Double
rating	2	Int32
review	This is test review from user to host about the trip. Lets see how it ...	String
hostId	ObjectId("583fa6ee493c3723184332d8")	ObjectId
userId	ObjectId("583faac0493c3723184332db")	ObjectId
__v	0	Int32
3 ObjectId("583fb4bb73f9a44f64a314f3")	{ 7 fields }	Object
4 ObjectId("584288c077ea9c36acaf0d0c")	{ 6 fields }	Object
5 ObjectId("584288c377ea9c36acaf0d0d")	{ 6 fields }	Object
6 ObjectId("584288c477ea9c36acaf0d0e")	{ 6 fields }	Object
7 ObjectId("584288c477ea9c36acaf0d0f")	{ 6 fields }	Object
8 ObjectId("584288c877ea9c36acaf0d10")	{ 7 fields }	Object
9 ObjectId("584288c977ea9c36acaf0d11")	{ 7 fields }	Object
10 ObjectId("584288c977ea9c36acaf0d12")	{ 7 fields }	Object
11 ObjectId("584288ca77ea9c36acaf0d13")	{ 7 fields }	Object
12 ObjectId("584288ca77ea9c36acaf0d14")	{ 7 fields }	Object
13 ObjectId("584288ca77ea9c36acaf0d15")	{ 7 fields }	Object

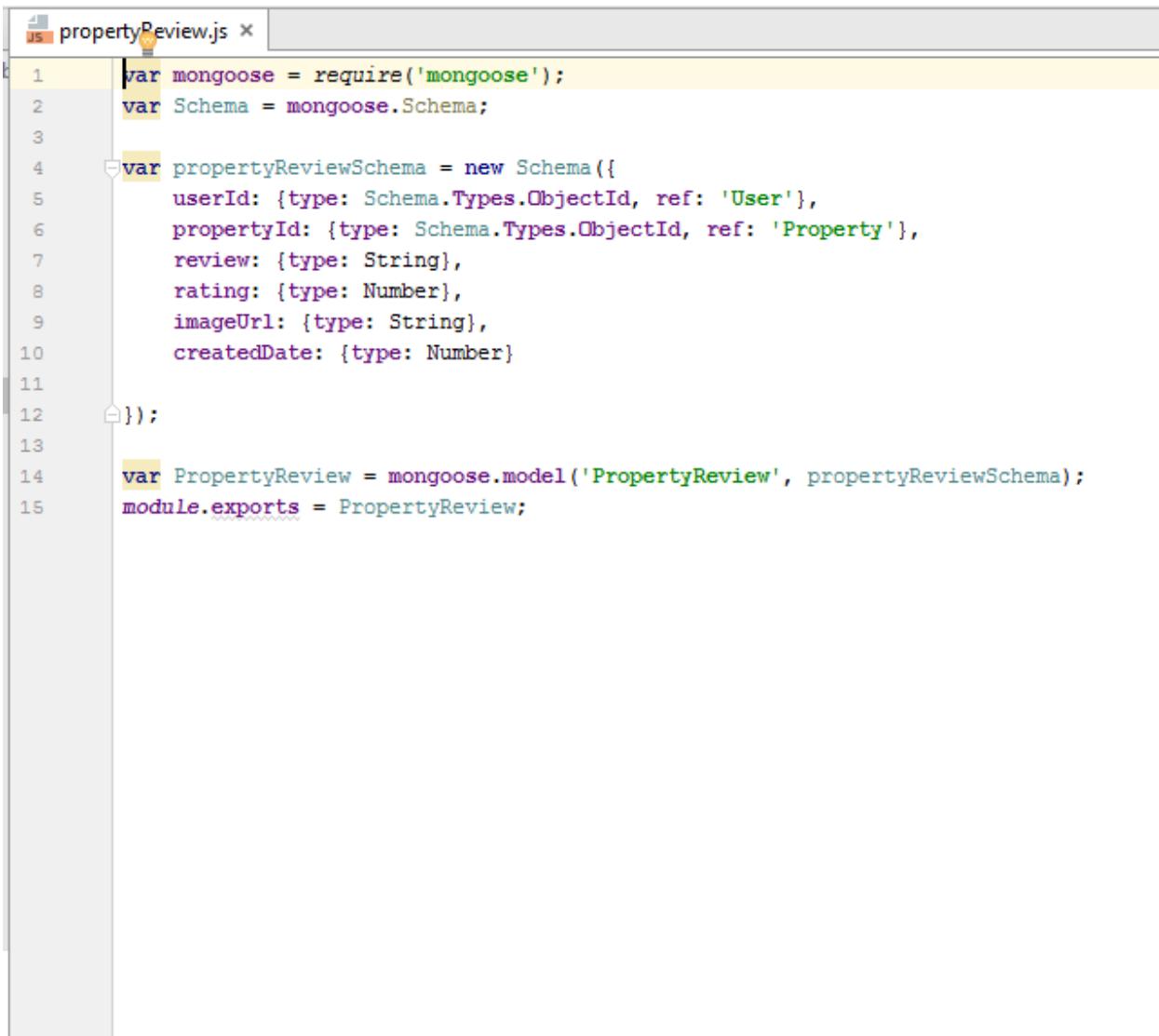
7)User Review Collection

```
1  var mongoose = require('mongoose');
2  var Schema = mongoose.Schema;
3
4  var userSchema = new Schema({
5      userId: {type: String},
6      userName: {type: String},
7      firstName: {type: String},
8      lastName: {type: String},
9      password: {type: String},
10     email: {type: String, unique:true},
11     address: {type: String},
12     city: {type: String},
13     state: {type: String},
14     zip: {type: String},
15     phoneNumber: {type: String},
16     profileImage: {type: String},
17     cardNumber: {type: Number},
18     cvv: {type: Number},
19     expDate: {type: Number},
20     isHost: {type: Boolean},
21     isActivated: {type: Boolean},
22     isApproved: {type: Boolean},
23     createdDate: {type: Number}
24 });
25
26  var User = mongoose.model('User', userSchema);
27
28  module.exports = User;
```

db.getCollection('userreviews').**find({})**

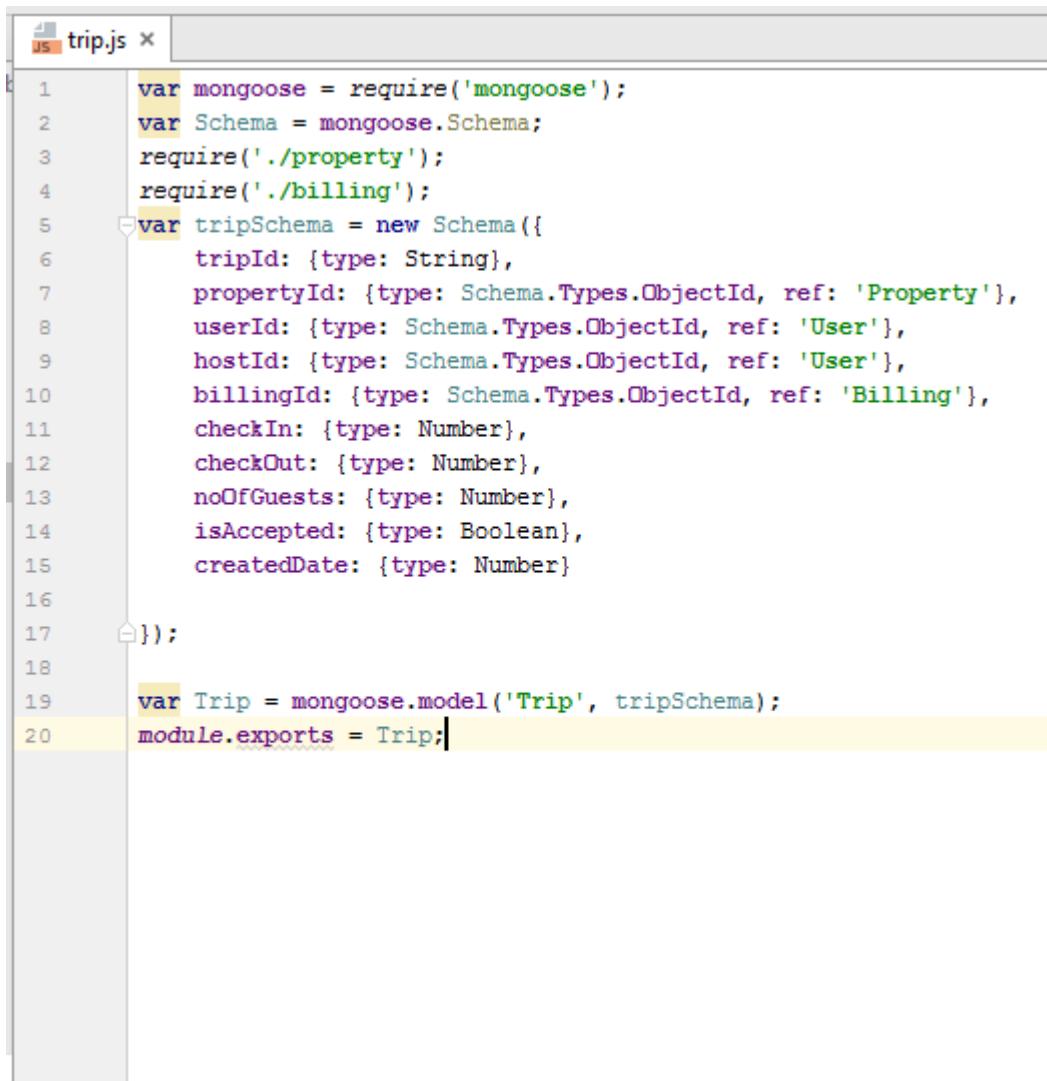
Key	Value	Type
✓ (1) ObjectId("583fb50373f9a44f64a314f4")	{ 7 fields }	Object
_id	ObjectId("583fb50373f9a44f64a314f4")	ObjectId
createdDate	1480570115026.0	Double
rating	1	Int32
review	This is a test review from host to user about the trip. Lets see how ...	String
hostId	ObjectId("583fa6ee493c3723184332d8")	ObjectId
userId	ObjectId("583fb50373f9a44f64a314f5")	ObjectId
__v	0	Int32
✓ (2) ObjectId("583fb57f73f9a44f64a314f6")	{ 7 fields }	Object
_id	ObjectId("583fb57f73f9a44f64a314f6")	ObjectId
createdDate	1480570239141.0	Double
rating	1	Int32
review	This is a test review from host to user about the trip. Lets see how ...	String
hostId	ObjectId("583fa6ee493c3723184332d8")	ObjectId
userId	ObjectId("583fb57f73f9a44f64a314f7")	ObjectId
__v	0	Int32
✓ (3) ObjectId("583fb64873f9a44f64a314f8")	{ 7 fields }	Object
_id	ObjectId("583fb64873f9a44f64a314f8")	ObjectId
createdDate	1480570440299.0	Double
rating	1	Int32
review	This is a test review from host to user about the trip. Lets see how ...	String
hostId	ObjectId("583fa6ee493c3723184332d8")	ObjectId
userId	ObjectId("583fb64873f9a44f64a314f9")	ObjectId
__v	0	Int32
> (4) ObjectId("583fb66273f9a44f64a314fa")	{ 7 fields }	Object
> (5) ObjectId("583fe57caeab1762dc2e9179")	{ 7 fields }	Object
> (6) ObjectId("583fe5d1eaeb1762dc2e917a")	{ 7 fields }	Object

8)Property Review Collection



```
propertyReview.js ×
1  var mongoose = require('mongoose');
2  var Schema = mongoose.Schema;
3
4  var propertyReviewSchema = new Schema({
5      userId: {type: Schema.Types.ObjectId, ref: 'User'},
6      propertyId: {type: Schema.Types.ObjectId, ref: 'Property'},
7      review: {type: String},
8      rating: {type: Number},
9      imageUrl: {type: String},
10     createdDate: {type: Number}
11
12 });
13
14 var PropertyReview = mongoose.model('PropertyReview', propertyReviewSchema);
15 module.exports = PropertyReview;
```

9)Trip Collection



A screenshot of a code editor window titled "trip.js". The code is written in JavaScript and defines a MongoDB schema for a "Trip" collection. The schema includes fields for tripId, propertyId, userId, hostId, billingId, checkIn, checkOut, noOfGuests, isAccepted, and createdDate. It uses mongoose.Types.ObjectId for references and mongoose.model to create the Trip model.

```
1 var mongoose = require('mongoose');
2 var Schema = mongoose.Schema;
3 require('./property');
4 require('./billing');
5 var tripSchema = new Schema({
6     tripId: {type: String},
7     propertyId: {type: Schema.Types.ObjectId, ref: 'Property'},
8     userId: {type: Schema.Types.ObjectId, ref: 'User'},
9     hostId: {type: Schema.Types.ObjectId, ref: 'User'},
10    billingId: {type: Schema.Types.ObjectId, ref: 'Billing'},
11    checkIn: {type: Number},
12    checkOut: {type: Number},
13    noOfGuests: {type: Number},
14    isAccepted: {type: Boolean},
15    createdDate: {type: Number}
16
17 });
18
19 var Trip = mongoose.model('Trip', tripSchema);
20 module.exports = Trip;
```

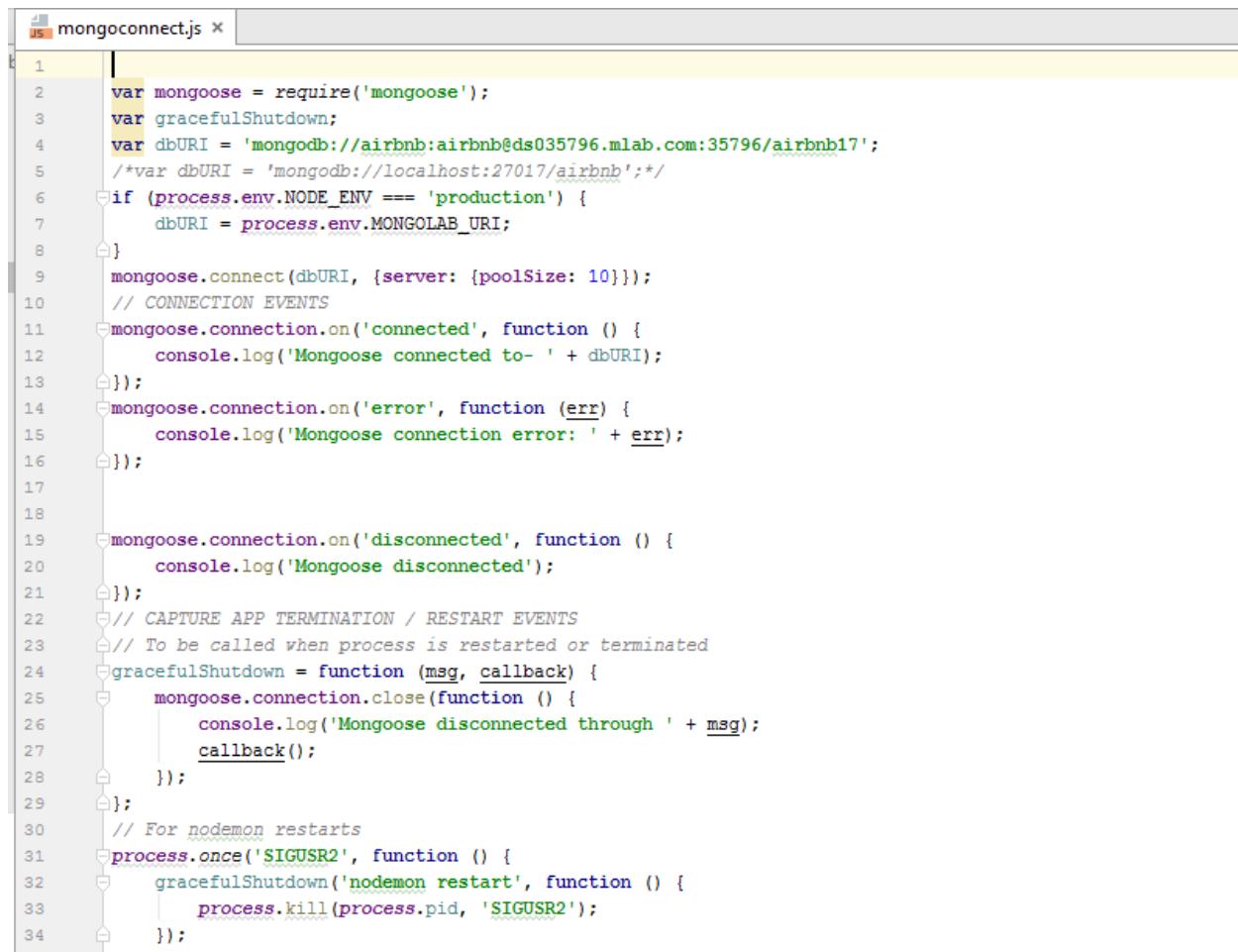
db.getCollection('trips').find({})		
Key	Value	Type
✓ (1) ObjectId("584079dce337c03b882355c7")	{ 12 fields }	Object
_id	ObjectId("584079dce337c03b882355c7")	ObjectId
createdDate	1480620508122.0	Double
isAccepted	true	Boolean
noOfGuests	2	Int32
checkOut	1481011200000.0	Double
checkIn	1480665600000.0	Double
hostId	ObjectId("583d4d177828aa2f286c396d")	ObjectId
userId	ObjectId("583f6dae3141c14bf8941ec2")	ObjectId
propertyId	ObjectId("583e8f508683e11820c158b4")	ObjectId
tripId	471-09-4294	String
__v	0	Int32
billingId	ObjectId("58407a08e337c03b882355c8")	ObjectId
✓ (2) ObjectId("5840a35a633c172848320b2f")	{ 12 fields }	Object
_id	ObjectId("5840a35a633c172848320b2f")	ObjectId
createdDate	1480631130523.0	Double
isAccepted	true	Boolean
noOfGuests	1	Int32
checkOut	1456819200000.0	Double
checkIn	1441090800000.0	Double
hostId	ObjectId("583d4d177828aa2f286c396d")	ObjectId
userId	ObjectId("583d4d177828aa2f286c396d")	ObjectId
propertyId	ObjectId("583d515dd3010925608b2c4e")	ObjectId
tripId	155-18-6060	String
__v	0	Int32
billingId	ObjectId("5841e9ca8f4e62103c23bb48")	ObjectId
✓ (3) ObjectId("5840a35c633c172848320b30")	{ 12 fields }	Object

• Mongo Session Management

```
app.use(session({
  secret: 'keyboard cat',
  resave: false,
  saveUninitialized: true,
  duration: 30 * 60 * 1000,
  activeDuration: 5 * 60 * 1000,
  store: new mongoStore({
    url: "mongodb://airbnb:airbnb@ds035796.mlab.com:35796/airbnb17"
  })
}));
```

db.getCollection('sessions').find()		
Key	Value	Type
(1) ilW3y8hw2T-gHSgTqfu8CwNbVFrdsYm	{ 3 fields } _id session expires	Object String String Date
(2) SzgFyL-eGqVp08ecv9KZTRNY1F0t6FxX	{ 3 fields } _id session expires	Object String String Date
(3) qUo5_BRoDjkXeMPz_BWA7bi-yr4-R7Y	{ 3 fields } _id session expires	Object String String Date
(4) hs_4kPR6WjK_1NYQlYhZQwE1AT_jbHzY	{ 3 fields }	Object
(5) vleK94ptv5M9b0FUzmQEVLsIUvx2s3m	{ 3 fields }	Object
(6) hp-yzVla62tHbk6TmlbAyqXcOmymwcs	{ 3 fields }	Object
(7) knL8-YZrGY50JFSDs78LrmNIXYuQer4	{ 3 fields }	Object
(8) v6A3E9Mqkt8SBVF7H4YnDS_I4BEFAis1	{ 3 fields }	Object
(9) YdqRo0TAN5iWq3wzd6IMO-LiEPBEMpHP	{ 3 fields }	Object
(10) IcM0eatBrJuOcxU35vN2HCEF3q31SHJ	{ 3 fields }	Object
(11) 373LPDVmDzzEl8uCywh94jSrtQxta7V	{ 3 fields }	Object
(12) aZpHg2eUColDgjqAyiizs38R0MYbCx	{ 3 fields }	Object
(13) Q-Bud1eNNEw4ewzxN4RBI_VTQRqH2b0	{ 3 fields }	Object
(14) ypxWu5f_FHHcmrSkv8vAvahi5YE2Fc9r	{ 3 fields }	Object
(15) -Odk7va70qNv174YjuLWNdDGcqhZox3	{ 3 fields }	Object
(16) ug8hvxsX296ldGKYUyP0j9erw58uuV2V	{ 3 fields }	Object
(17) 1VO8zTmte7zc5bPLijjigYxawXGindy	{ 3 fields }	Object
(18) f3beZ n4MsFjsVX FA1K 2YlwYmCMdf	{ 3 fields }	Object

- **Connection Pooling**
- **Mongo Connection Pooling**



```

1  | var mongoose = require('mongoose');
2  | var gracefulShutdown;
3  |
4  | var dbURI = 'mongodb://airbnb:airbnb@ds035796.mlab.com:35796/airbnb17';
5  | /*var dbURI = 'mongodb://localhost:27017/airbnb';*/
6  | if (process.env.NODE_ENV === 'production') {
7  |   dbURI = process.env.MONGOLAB_URI;
8  | }
9  | mongoose.connect(dbURI, {server: {poolSize: 10}});
10 | // CONNECTION EVENTS
11 | mongoose.connection.on('connected', function () {
12 |   console.log('Mongoose connected to- ' + dbURI);
13 | });
14 | mongoose.connection.on('error', function (err) {
15 |   console.log('Mongoose connection error: ' + err);
16 | });
17 |
18 |
19 | mongoose.connection.on('disconnected', function () {
20 |   console.log('Mongoose disconnected');
21 | });
22 | // CAPTURE APP TERMINATION / RESTART EVENTS
23 | // To be called when process is restarted or terminated
24 | gracefulShutdown = function (msg, callback) {
25 |   mongoose.connection.close(function () {
26 |     console.log('Mongoose disconnected through ' + msg);
27 |     callback();
28 |   });
29 | };
30 | // For nodemon restarts
31 | process.once('SIGUSR2', function () {
32 |   gracefulShutdown('nodemon restart', function () {
33 |     process.kill(process.pid, 'SIGUSR2');
34 |   });
}

```

Cache management using Redis

We can use redis to drastically improve performance of the application.

For this application, we implement a caching system to cache the most viewed properties. Everytime a property is requested from the user, it is stored in a hashmap data structure using redis. Next time if the same property is requested, the application doesn't need to query the database and can be returned instantly.

We also set up a expiry time to make sure the cache is not old enough to return old data.

We can see improved performance using this method.

```
6  var redis = require('redis');
7  var client = redis.createClient();
8  var logger=require('../routes/usertracking');
9
10 client.on('ready', function () {
11
12     console.log("Redis Ready");
13
14 });
15
16 client.on('error', function (err) {
17     console.log("Error " + err);
18 });
19
```

```
client.hget("properties", id, function (err, result) {
    if (result) {
        console.log("From redis");
        obj=JSON.parse(result);
        logger.info(req.session.firstName+ " clicked on "+obj.id,{ 'user':req.session.firstName, 'property_clicked':obj.id, 'id':id});
        res.end(result);
    } else {
        getProperty(req, res);
    }
});
```

```
function getProperty (req, res) {
  var id = req.param("propertyId");

  var msg_payload = {
    id: id
  };

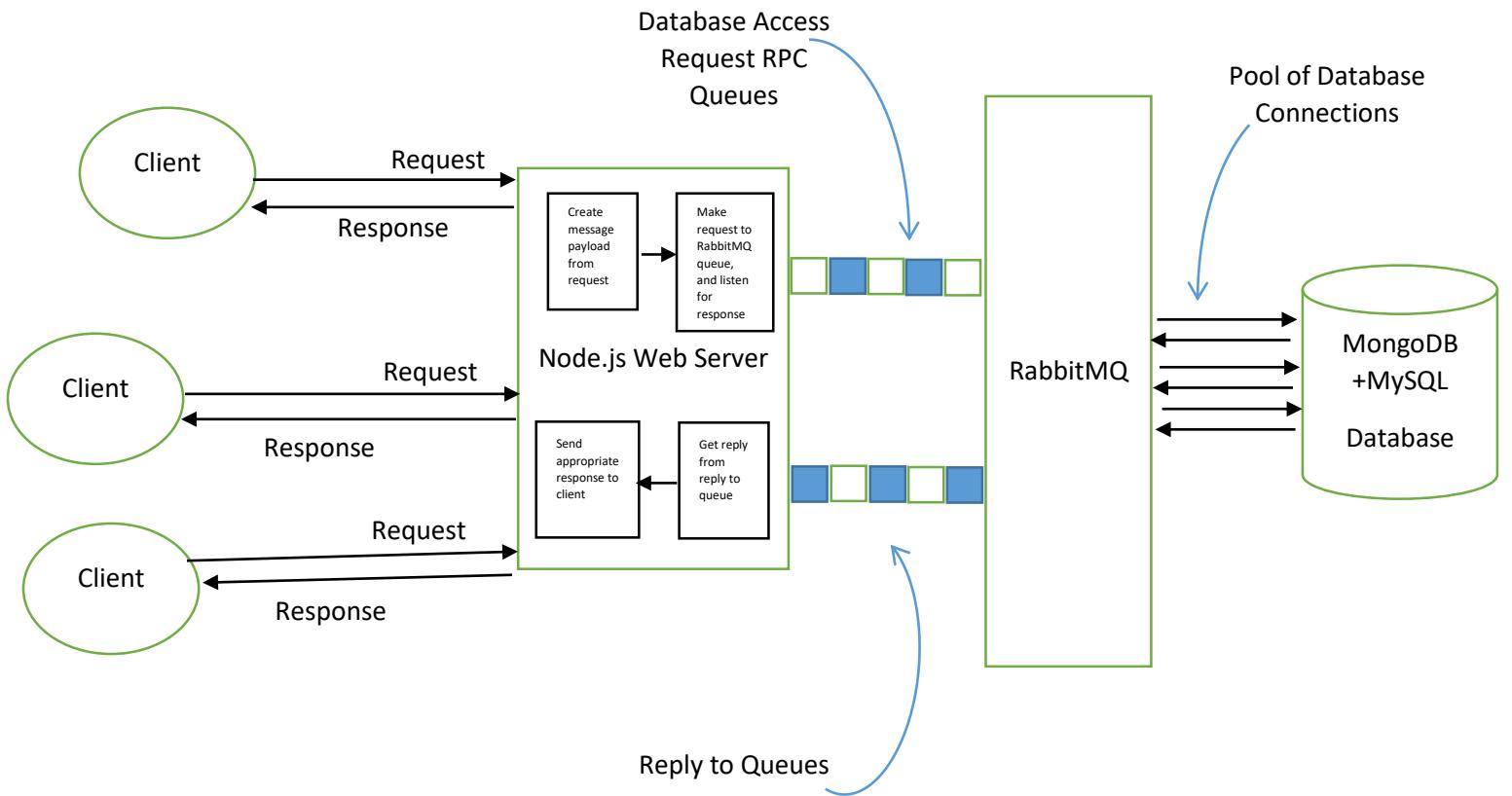
  mq_client.make_request('property_detail_queue', msg_payload, function (err, result) {
    if (err) {
      console.log("From mongoose");
      console.log(err);
      var json_responses = {"statusCode": 401};
      res.send(json_responses);
    } else {
      console.log(result.id);
      console.log("From mongoose");
      client.hmset("properties", result.id, JSON.stringify(result), redis.print);
      // var json_responses = {"statusCode": 200, "data": result};
      logger.info(req.session.firstName+" clicked on "+result.id,{user:req.session.firstName});

      res.send(result);
      res.end();
    }
  });
}
```

- **Scalability and Robustness**

The application has been made scalable with the help of RabbitMQ to handle multiple concurrent users. It is possible because we have used different queues for different requests.

- **RabbitMQ Architecture**



● RabbitMQ Management Console

Totals

Queued messages (chart: last minute) (?)

Ready: 0
Unacked: 0
Total: 0

Message rates (chart: last minute) (?)

Publish: 0.00/s
Confirm: 0.00/s
Publish (In): 0.00/s
Get (noack): 0.00/s
Return: 0.00/s
Publish (Out): 0.00/s
Deliver: 0.00/s
Redelivered: 0.00/s
Acknowledge: 0.00/s
Get: 0.00/s
Deliver (noack): 0.00/s

Global counts (?)

Connections: 2
Channels: 43
Exchanges: 8
Queues: 41
Consumers: 41

Node

Node: rabbit@DESKTOP-I7LUL59 ([More about this node](#))

File descriptors (?)	Socket descriptors (?)	Erlang processes	Memory	Disk space	Rates mode	Info	+/-
0 8192 available	2 7280 available	553 1048576 available	57MB 3.2GB high watermark	362GB 48MB low watermark	basic	Disc 2 Stats	

- RabbitMQ Queues

Overview			Messages			Message rates		
Name	Features	State	Ready	Unacked	Total	incoming	deliver / get	ack
acceptTrip_queue	AD	idle	0	0	0			
addHostReview_queue	AD	idle	0	0	0			
addNewListing_queue	AD	idle	0	0	0			
addUserReview_queue	AD	idle	0	0	0			
amq.gen-PNw_LUjU8pcMer0_gCW76Q	Excl AD	idle	0	0	0	0.00/s	0.00/s	0.00/s
amq.gen-YCbTYDoO3JpmOtKYf5M1Jw	Excl AD	idle	0	0	0	0.00/s	0.60/s	0.00/s
bidCron_queue	AD	running	0	0	0	0.00/s	0.00/s	0.00/s
cardDetail_queue	AD	idle	0	0	0			
checkHost_queue	AD	idle	0	0	0			
confirmBooking_queue	AD	idle	0	0	0			
dynamicPricingCron_queue	AD	running	0	0	0	0.00/s	0.00/s	0.00/s
editProperty_queue	AD	idle	0	0	0			
editUser_queue	AD	idle	0	0	0			
getActiveListings_queue	AD	idle	0	0	0			
getDataInEditProperty_queue	AD	idle	0	0	0			
getHostReview_queue	AD	idle	0	0	0			
getItinerary_queue	AD	idle	0	0	0			
getPropertyDetails_queue	AD	idle	0	0	0			
getReservations_queue	AD	idle	0	0	0			
getUserProfile_queue	AD	idle	0	0	0			
getUserReview_queue	AD	idle	0	0	0			
getUserTrips_queue	AD	idle	0	0	0			
hostReviewsCount_queue	AD	idle	0	0	0			
Last 1000 Rows	AD	idle	0	0	0			

```
cnn.queue('login_queue', function (q) {...});  
  
cnn.queue('register_queue', function (q) {...});  
  
cnn.queue('search_queue', function (q) {...});  
  
cnn.queue('property_detail_queue', function (q) {...});  
  
cnn.queue('editUser_queue', function (q) {...});  
  
cnn.queue('loadEditUser_queue', function (q) {...});  
  
cnn.queue('loadReviewAboutPage_queue', function (q) {...});  
  
cnn.queue('loadReviewByPage_queue', function (q) {...});  
  
cnn.queue('hostReviewsCount_queue', function (q) {...});  
  
cnn.queue('uploadProfileImage_queue', function (q) {...});  
  
cnn.queue('loadProfilePhotoPage_queue', function (q) {...});  
  
cnn.queue('updatePassword_queue', function (q) {...});  
  
cnn.queue('updatePaymentMethod_queue', function (q) {...});  
  
cnn.queue('payinTransaction_queue', function (q) {...});
```

```
+    cnn.queue('updatePaymentMethod_queue', function (q) {...});
+    cnn.queue('payinTransaction_queue', function (q) {...});
+    cnn.queue('payoutTransactions_queue', function (q) {...});
+    cnn.queue('receiptPage_queue', function (q) {...});
+    cnn.queue('loadPaymentPage_queue', function (q) {...});
+    cnn.queue('getPropertyDetails_queue', function (q) {...});
+    cnn.queue('confirmBooking_queue', function (q) {...});
+    cnn.queue('getUserTrips_queue', function (q) {...});
+    cnn.queue('getActiveListings_queue', function (q) {...});
+    cnn.queue('addNewListing_queue', function (q) {...});
+    cnn.queue('getReservations_queue', function (q) {...});
+    cnn.queue('acceptTrip_queue', function (q) {...});
+    cnn.queue('getItinerary_queue', function (q) {...});
+    cnn.queue('getUserProfile_queue', function (q) {...});
+    cnn.queue('getUserReview_queue', function (q) {...});
+    cnn.queue('getHostReview_queue', function (q) {...});
+    cnn.queue('addUserReview_queue', function (q) {...});
+    cnn.queue('addHostReview_queue', function (q) {...});
+    cnn.queue('updateUserToHost_queue', function (q) {...});
+    cnn.queue('editProperty_queue', function (q) {...});
+    cnn.queue('getDataInEditProperty_queue', function (q) {...});
+    cnn.queue('checkHost_queue', function (q) {...});
+    cnn.queue('cardDetail_queue', function (q) {...});
+    cnn.queue('updateBasePrice_queue', function (q) {...});
+    cnn.queue('bidCron_queue', function (q) {...});
+    cnn.queue('dynamicPricingCron_queue', function (q) {...});
+    cnn.queue('removeListing_queue', function (q) {...});
});


```

- Bcrypt Encryption

```
93 exports.registerUser = function (req, res) {
94     var f_name = req.body.first_name;
95     var l_name = req.body.last_name;
96     var email_id = req.body.email_id;
97     var pwd = req.body.password;
98     var sess = req.session;
99     sess.isLoggedIn = false;
100    var salt = bcrypt.genSaltSync(10);
101    var passwordToSave = bcrypt.hashSync(pwd, salt);
102    var msg_payload = {
103        firstName: f_name,
104        lastName: l_name,
105        email_id: email_id,
106        password: passwordToSave
107    };
108
109    mq_client.make_request('register_queue', msg_payload, function (err, results) {
110        if (err) {
111            res.json({
112                success: false,
113                message: 'Error in registration.'
114            });
115            res.end();
116        }
117        if (results) {
118
119            sess.userSSN = results.userId;
120            sess.firstName = results.firstName;
121            sess.lastName = results.lastName;
122            sess.userId = results._id;
123            sess.email = results.email;
124            sess.isLoggedIn = true;
125            res.json({
126                success: true,
```

- Passport.js Authentication

```
exports.authenticateUser = function (req, res, next) {  
  
    // var email_id = req.body.email_id;  
    // var pwd = req.body.password;  
    var sess = req.session;  
    sess.isLoggedIn = false;  
    console.log("in signin");  
    passport.authenticate('login', function (err, user) {  
        console.log("Result" + user);  
        if (err) {  
            return next(err);  
        }  
  
        if (!user) {  
  
            res.json({  
                success: false,  
                message: 'No user found'  
            });  
            res.end();  
            /*return res.redirect('/signup');*/  
        }  
  
        if (user) {  
  
            sess.email = user.email;  
            sess.isLoggedIn = true;  
            sess.userSSN=user.userId;  
            sess.firstName=user.firstName;  
            sess.lastName=user.lastName;  
            sess.userId=user._id;  
            sess.isHost = user.isHost;  
            if(user.profileImage)  
  
});  
function isAuthenticated(req, res, next) {  
    if (req.session.user_id) {  
        console.log(req.session.user_id);  
        return next();  
    }  
    }  
    res.redirect('/signinPg');  
}
```

- **Dynamic Pricing Algorithm**

Explanation

We implemented a dynamic pricing algorithm to generate fair pricing for properties relative to the popularity of the property among the user.

We implement it by using the data of popularity of the property. We can measure popularity by finding number of trips booked in the past two months. We can use this data to generate future demand of the property.

Using the popularity, we generate a multiplier which we can assign to the property. The multiplier ranges from 1 to 2 in steps of 0.2-0.3 depending on the popularity.

The multiplier is clearly displayed while booking the property as well as in the receipts.

File Watcher

```
1  /**
2   * Created by Parth on 03-12-2016.
3  */
4  var Trip= require('../model/trip');
5  var Property=require('../model/property');
6  var Map=require('hashmap');
7
8 exports.dynamicPriceCron = function (msg, callback) {
9
10
11     var makeDate = new Date();
12     makeDate = new Date(makeDate.setMonth(makeDate.getMonth() - 2)).getTime();
13     console.log(new Date(makeDate).toDateString());
14
15     Trip.find({createdDate: {$gt: makeDate}}).populate('propertyId').exec(function (err, result) {
16
17         if (err) {
18             console.log(err);
19             callback(err, null);
20         }
21         else {
22
23             //console.log(result);
24             var map = new Map();
25             var priceMap = new Map();
26
27             for (var i = 0; i < result.length; i++) {
28
29
30                 var propertyId = result[i].propertyId._id.toString();
31
32                 if (map.has(propertyId)) {
33                     map.set(propertyId, map.get(propertyId) + 1);
34
35                 }
36
37             }
38
39             callback(null, map);
40         }
41     });
42 }
```

File Watcher

```
28
29
30         var propertyId = result[i].propertyId._id.toString();
31
32     if (map.has(propertyId)) {
33         map.set(propertyId, map.get(propertyId) + 1);
34     }
35     else {
36         map.set(propertyId, 1);
37         priceMap.set(propertyId, result[i].propertyId.price);
38     }
39 }
40
41 //onsole.log(map);
42 var propertyIDArray = [];
43 var dynamicMultiplierArray = [];
44 //onsole.log("count " + map.count());
45 if (map.count() > 0) {
46
47     map.forEach(function (value, key) {
48         if (value >= 5 && value < 20) {
49
50             propertyIDArray.push(key);
51             var newMultiplier = 1.2;
52             dynamicMultiplierArray.push(newMultiplier);
53         }
54         else if (value >= 20 && value < 40) {
55
56             propertyIDArray.push(key);
57             var newMultiplier = Number(priceMap.get(key)) * 1.5;
58             dynamicMultiplierArray.push(newMultiplier);
59         }
60         else if (value >= 40 && value < 50) {
61             propertyIDArray.push(key);
62         }
63     });
64 }
```

File Watcher

```
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
```

File Watcher

```
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
```

Dynamic pricing cron

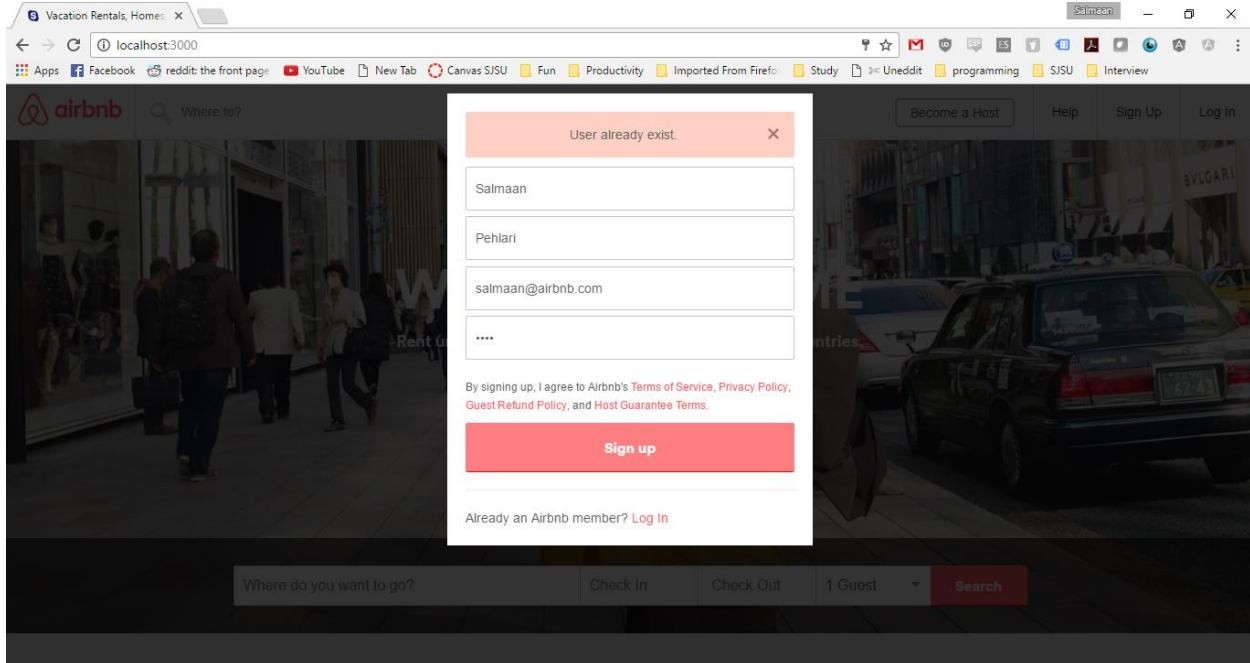
We implemented a cron job to generate and update the dynamic pricing of the properties. The cron checks the trip details for the properties and passes it to the dynamic pricing algorithm.

```
* Created by Parth on 03-12-2016.
*/
var mq_client = require('../rpc/client');
var cron = require('node-cron');

cron.schedule('*/*600 * * * *', function (req, res, next) {
  var msg_payload = {};
  mq_client.make_request('dynamicPricingCron_queue', msg_payload, function (err, results) {
    if (err) {
      console.log(err);
    } else {
      console.log("dynamic Cron Success");
    }
  });
});
```

- **Exception Handling and Validation**

Validation in Registration - If user already exists then the following screen will appear



Project Structure:

```

Project
  |- routes
    |- home.js
    |- index.js
    |- users.js
  |- rpc
    |- amqprpc.js
    |- client.js
  |- views
    |- advertise.ejs
    |- cart.ejs
    |- error.ejs
    |- extra_user_information.ejs
    |- index.ejs
    |- mainpage.ejs
    |- productpage.ejs
    |- signup.ejs
  |- app.js
  |- package.json
  |- node_modules
  |- Server
    |- config
      |- models
        |- bids.js
        |- cart.js
        |- db.js
        |- product.js
        |- transaction.js
        |- user.js
      |- services
        |- server.js
      |- server1.js
  |- External Libraries

```

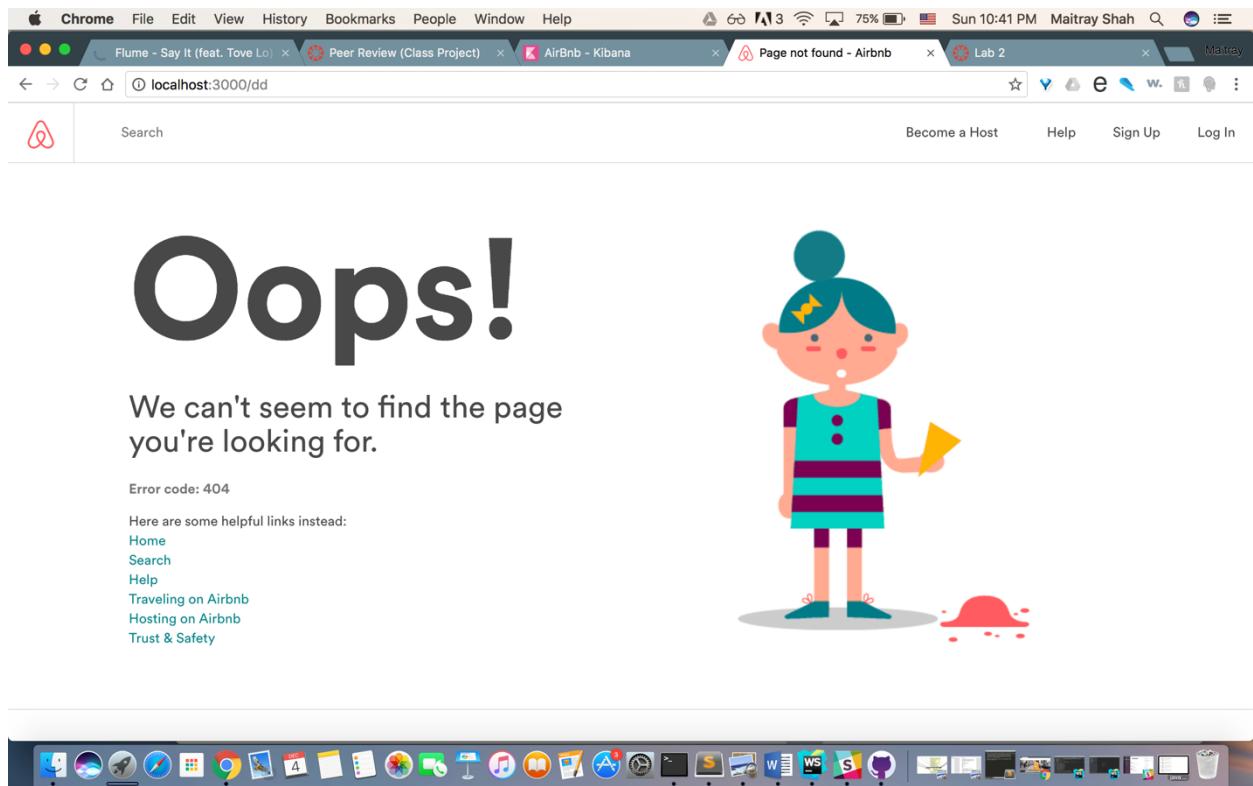
Code Snippet from server1.js (lines 145-184):

```

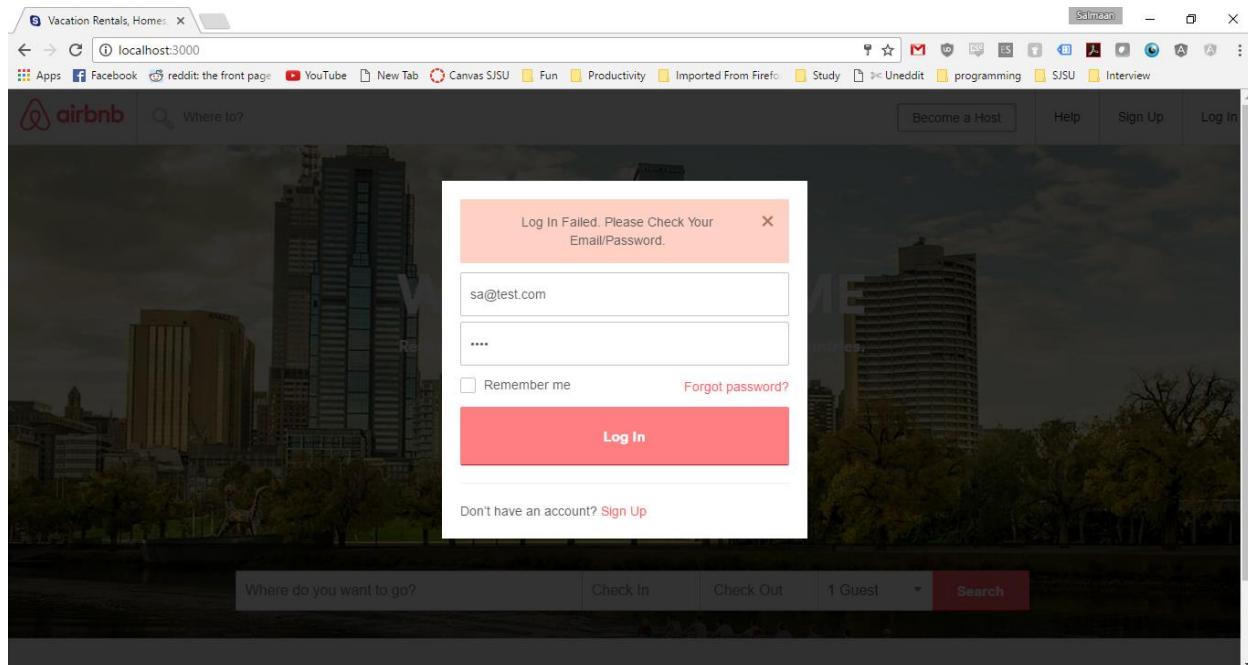
function saveInfo(req, resp) {
  var json_response;
  try {
    var msg_payload = {
      "firstname": req.body.firstname,
      "lastname": req.body.lastname,
      "birthdate": req.body.birthdate,
      "phonenumber": req.body.phonenumber,
      "address": req.body.address,
      "city": req.body.city,
      "zip": req.body.zip,
      "username": req.session.username
    };
    console.log("STORE EXTRA USER INFORMATION", "Sending message to extra info queue");
    mq_client.make_request('moreInfo_queue', msg_payload, function(err, results) {
      console.log(results.code);
      if (err) {
        console.log(err);
        resp.end();
      } else if (results.code == 200) {
        json_response = {"statusCode": 200};
        resp.send(json_response);
        resp.end();
      } else if (results.code == 400) {
        json_response = {"statusCode": 400};
        resp.send(json_response);
        resp.end();
      }
    });
  } catch (e) {
  }
}

```

Event Log: 185:24 LF: UTF-8



Validation in Login - If email and/or password are incorrect then the following screen will appear.



Validation in change password - If the new password and the confirm password do not match then the following screen will appear.

The screenshot shows the Airbnb account security page at localhost:3000/Account_Security. The user is attempting to change their password. The 'Old Password' field contains '....'. The 'New Password' field contains '....'. The 'Confirm Password' field is highlighted in yellow. A red error message 'Passwords do not match!' is displayed below the fields.

If any of the credit card details are not in valid format then the following screen will appear

The screenshot shows the Airbnb payment page at localhost:3000/getPaymentPage?propertyId=5844d9b1a7f87265a0ab34&checkin=4-12-2016&checkout=5-12-2016&guests=2.8. The user is entering payment information for a booking. In the 'Card number' field, the placeholder 'Please Enter Valid Card Details' is visible. To the right, there is a listing for '101 San Fernando' in San Jose, CA, showing a bedroom and listing details.

localhost:3000/getPaymentPage?propertyId=5844d9b1a7fb7265a0a0ab34&checkin=4-12-2016&checkout=5-12-2016&guests=2.8

Please Enter Valid Expire Dates

Please Enter Valid Security Code

Billing Information

First name: test Last name: test

Postal code: United States

Cancellation Policy	Flexible
House Rules	Read policy
Nights	1
\$200 x 1 per night	\$200
\$200 x 1 night	\$200
Airbnb Service Fees	\$35
Total	\$235

House Rules

By booking this space you're agreeing to follow House Rules..

By clicking on "Continue", you agree to pay the total amount shown, which includes Service Fees, on the right and to the [Terms of Service](#), [House Rules](#), [Cancellation Policy](#) and [Guest Refund Policy](#).

Confirm

- Mocha Testing:

```

1 /**
2  * Created by Salmaan on 10/16/2016.
3 */
4
5 var express = require('express');
6 var request = require('request');
7 var assert = require('assert');
8 var http = require('http');
9 var mocha = require('mocha');
10
11 describe('API tests', function () {
12
13     it('should display profile page', function(done){
14
15         http.get('http://localhost:3000/profile/398-30-8465', function(res){
16             assert.equal(200, res.statusCode);
17             done();
18         });
19     });
20
21
22     it('should display property page', function(done){
23
24         http.get('http://localhost:3000/property?propertyId=5844b06feb037a6060fe1ee7', function(res){
25             assert.equal(200, res.statusCode);
26             done();
27         });
28     });
29
30 });
31
32     it('should display home page', function(done){
33
34         http.get('http://localhost:3000/*', function(res){
35             assert.equal(200, res.statusCode);
36             done();
37         });
38     });
39
40
41     it('should display 404, as property doesnt exist', function(done){
42
43         http.get('http://localhost:3000/property?propertyId=423', function(res){
44             assert.equal(200, res.statusCode);
45             done();
46         });
47     });
48
49     it('should display 404 page', function(done){
50
51         http.get('http://localhost:3000/profile/salmaan', function(res){
52             assert.equal(200, res.statusCode);
53             done();
54         });
55     });
56
57 });
58
59 });
60

```

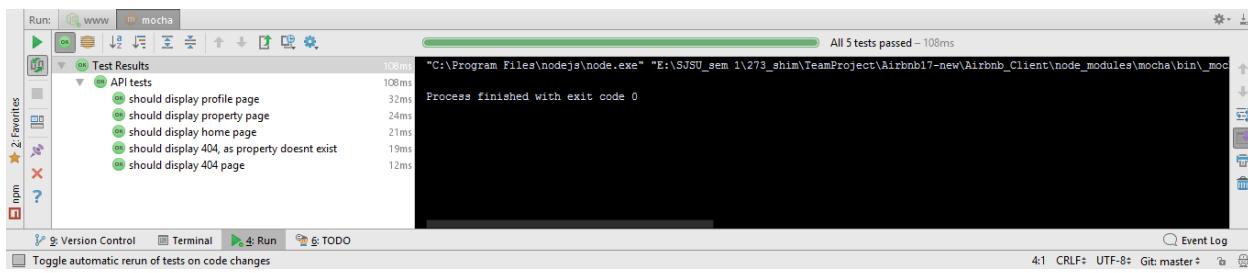
```

28         assert.equal(200, res.statusCode);
29         done();
30     });
31
32
33     it('should display home page', function(done){
34
35         http.get('http://localhost:3000/*', function(res){
36             assert.equal(200, res.statusCode);
37             done();
38         });
39     });
40
41     it('should display 404, as property doesnt exist', function(done){
42
43         http.get('http://localhost:3000/property?propertyId=423', function(res){
44             assert.equal(200, res.statusCode);
45             done();
46         });
47     });
48
49     it('should display 404 page', function(done){
50
51         http.get('http://localhost:3000/profile/salmaan', function(res){
52             assert.equal(200, res.statusCode);
53             done();
54         });
55     });
56
57 });
58
59 });
60

```

Output:

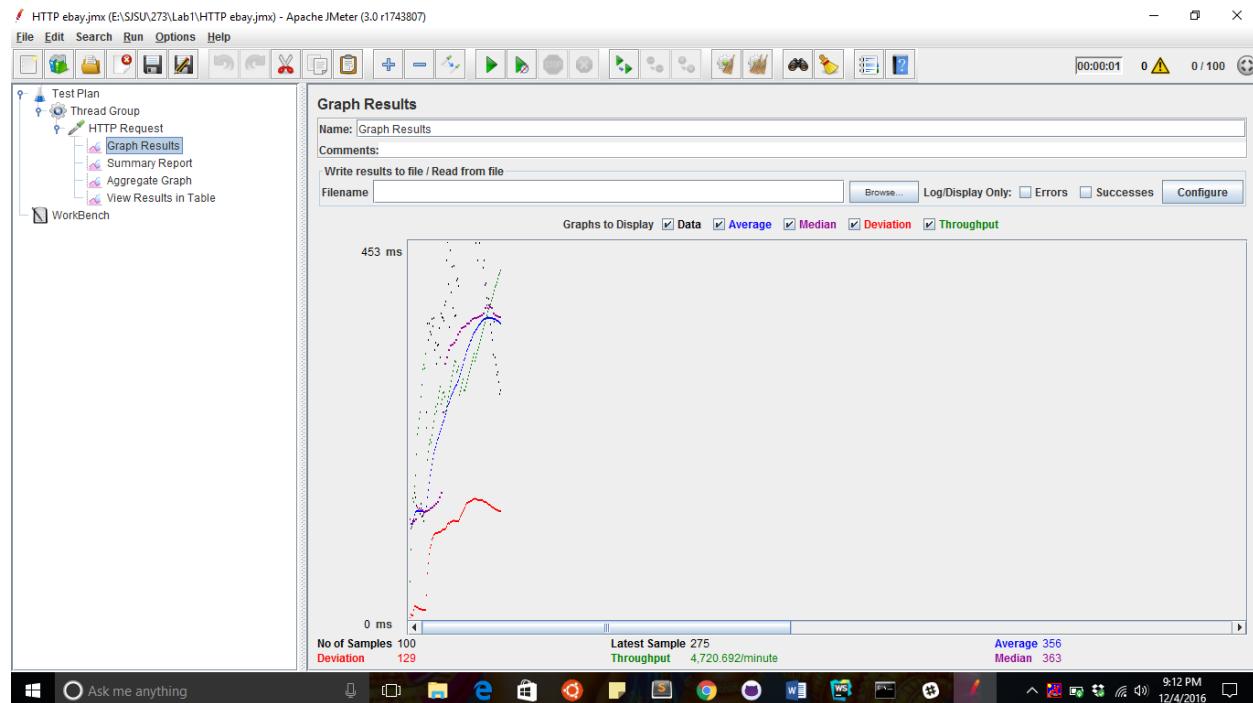
Mocha Testing screenshots



JMeter Testing:

1. Base

1) 100 concurrent users



HTTP ebay.jmx (E:\SJSU\273\Lab1\HTTP ebay.jmx) - Apache JMeter (3.0 r1743807)

File Edit Search Run Options Help

Test Plan Thread Group
HTTP Request
Graph Results Summary Report Aggregate Graph View Results in Table WorkBench

Summary Report

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

Label	# Samples	Average	Min	Max	Std. Dev.	Error %	Throughput	KB/sec	Avg. Bytes
HTTP Request	100	356	109	576	129.96	0.00%	78.7/sec	201.58	2623.5
TOTAL	100	356	109	576	129.96	0.00%	78.7/sec	201.58	2623.5

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

9:11 PM 12/4/2016

2) 200 concurrent users

HTTP ebay.jmx (E:\SJSU\273\Lab1\HTTP ebay.jmx) - Apache JMeter (3.0 r1743807)

File Edit Search Run Options Help

Test Plan Thread Group
HTTP Request
Graph Results Summary Report Aggregate Graph View Results in Table WorkBench

Graph Results

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

Graphs to Display Data Average Median Deviation Throughput

834 ms
0 ms
No. of Samples 200
Deviation 263
Latest Sample 673
Throughput 7,177.033/min
Average 465
Median 398

9:14 PM 12/4/2016

HTTP ebay.jmx (E:\SJSU\273\Lab1\HTTP ebay.jmx) - Apache JMeter (3.0 r1743807)

File Edit Search Run Options Help

Test Plan Thread Group
HTTP Request
Graph Results Summary Report Aggregate Graph View Results in Table WorkBench

Summary Report

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

Label	# Samples	Average	Min	Max	Std. Dev.	Error %	Throughput	KB/sec	Avg. Bytes
HTTP Request	200	465	118	893	263.81	0.00%	119.6/sec	306.47	2623.6
TOTAL	200	465	118	893	263.81	0.00%	119.6/sec	306.47	2623.6

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

9:13 PM 12/4/2016

3)300 concurrent users

HTTP ebay.jmx (E:\SJSU\273\Lab1\HTTP ebay.jmx) - Apache JMeter (3.0 r1743807)

File Edit Search Run Options Help

Test Plan Thread Group
HTTP Request
Graph Results Summary Report Aggregate Graph View Results in Table WorkBench

Graph Results

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

Graphs to Display Data Average Median Deviation Throughput

1311 ms
0 ms
No of Samples 300
Deviation 401
Latest Sample 1035
Throughput 8,759.124/min
Average 885
Median 1051

9:15 PM 12/4/2016

HTTP ebay.jmx (E:\SJSU\273\Lab1\HTTP ebay.jmx) - Apache JMeter (3.0 r1743807)

File Edit Search Run Options Help

Test Plan Thread Group HTTP Request Graph Results Summary Report Aggregate Graph View Results in Table WorkBench

Summary Report

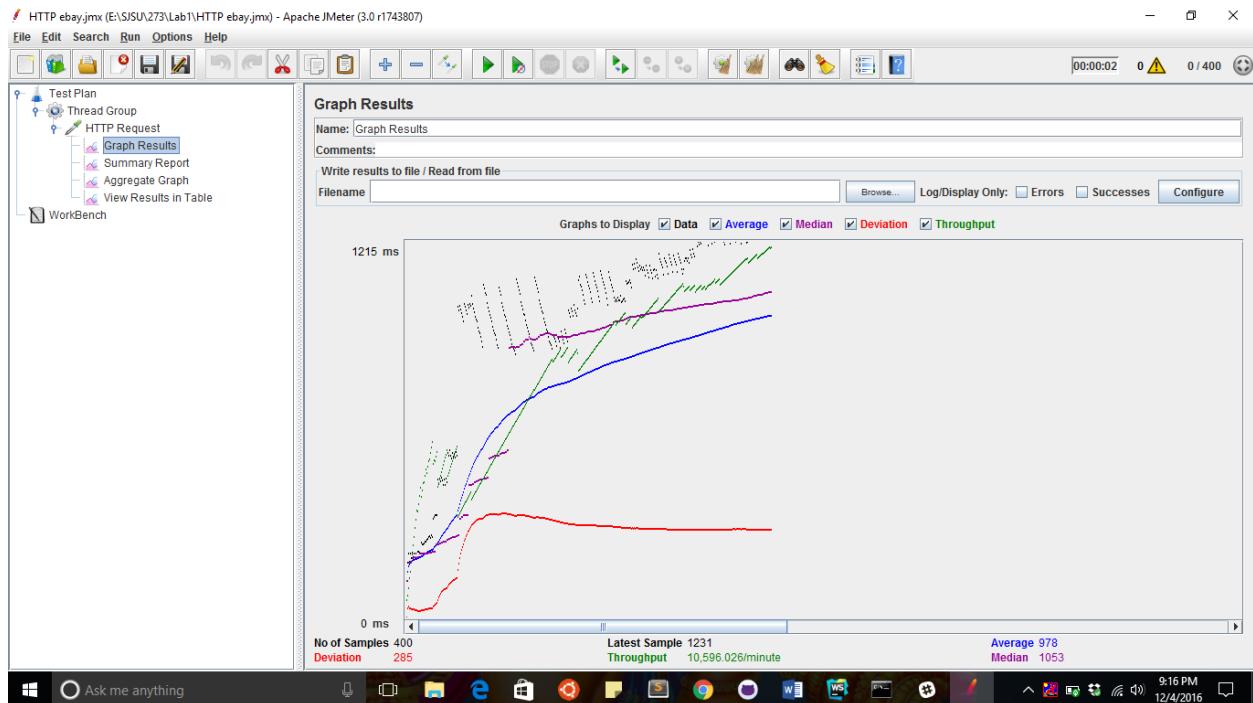
Name: Summary Report
Comments:
Write results to file / Read from file
Filename Browse... Log/Display Only: Errors Successes Configure

Label	# Samples	Average	Min	Max	Std. Dev.	Error %	Throughput	KB/sec	Avg. Bytes
HTTP Request	300	885	119	1382	401.75	0.00%	146.0/sec	374.03	2623.6
TOTAL	300	885	119	1382	401.75	0.00%	146.0/sec	374.03	2623.6

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

9:15 PM 12/4/2016

4) 400 concurrent users



HTTP ebay.jmx (E:\SJSU\273\Lab1\HTTP ebay.jmx) - Apache JMeter (3.0 r1743807)

File Edit Search Run Options Help

Test Plan Thread Group
HTTP Request
Graph Results Summary Report Aggregate Graph View Results in Table WorkBench

Summary Report

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

Label	# Samples	Average	Min	Max	Std. Dev.	Error %	Throughput	KB/sec	Avg. Bytes
HTTP Request	400	978	119	1281	285.67	0.00%	176.6/sec	452.49	2623.7
TOTAL	400	978	119	1281	285.67	0.00%	176.6/sec	452.49	2623.7

Include group name in label? Save Table Header

9:16 PM 12/4/2016

5)500 concurrent users

HTTP ebay.jmx (E:\SJSU\273\Lab1\HTTP ebay.jmx) - Apache JMeter (3.0 r1743807)

File Edit Search Run Options Help

Test Plan Thread Group
HTTP Request
Graph Results Summary Report Aggregate Graph View Results in Table WorkBench

Graph Results

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

Graphs to Display Data Average Median Deviation Throughput

2084 ms
0 ms
No of Samples 500
Deviation 612
Latest Sample 1811
Throughput 10,362.694/min
Average 1541
Median 1847

9:17 PM 12/4/2016

HTTP ebay.jmx (E:\SJSU\273\Lab1\HTTP ebay.jmx) - Apache JMeter (3.0 r1743807)

File Edit Search Run Options Help

Test Plan

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

Summary Report

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

Label	# Samples	Average	Min	Max	Std. Dev.	Error %	Throughput	KB/sec	Avg. Bytes
HTTP Request	500	1541	219	2169	612.81	0.00%	172.7/sec	442.53	2623.8
TOTAL	500	1541	219	2169	612.81	0.00%	172.7/sec	442.53	2623.8

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

9:17 PM 12/4/2016

2. Base + Connection Pooling

1) 100 concurrent users

HTTP ebay.jmx (E:\SJSU\273\Lab1\HTTP ebay.jmx) - Apache JMeter (3.0 r1743807)

File Edit Search Run Options Help

Test Plan

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

Graph Results

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

Graphs to Display: Data Average Median Deviation Throughput

483 ms
0 ms
No of Samples 100
Deviation 160
Latest Sample 440
Throughput 4,341.534/min
Average 262
Median 211

9:05 PM 12/4/2016

HTTP ebay.jmx (E:\SISU\273\Lab1\HTTP ebay.jmx) - Apache JMeter (3.0 r1743807)

File Edit Search Run Options Help

Test Plan

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

WorkBench

Summary Report

Name: [Summary Report]
Comments:
Write results to file / Read from file
Filename Browse... Log/Display Only: Errors Successes

Label	# Samples	Average	Min	Max	Std. Dev.	Error %	Throughput	KB/sec	Avg. Bytes
HTTP Request	100	262	90	563	160.52	0.00%	72.4/sec	185.37	2623.3
TOTAL	100	262	90	563	160.52	0.00%	72.4/sec	185.37	2623.3

Include group name in label? Save Table Header

9:05 PM 12/4/2016

2) 200 concurrent users

HTTP ebay.jmx (E:\SISU\273\Lab1\HTTP ebay.jmx) - Apache JMeter (3.0 r1743807)

File Edit Search Run Options Help

Test Plan

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

WorkBench

Graph Results

Name: [Graph Results]
Comments:
Write results to file / Read from file
Filename Browse... Log/Display Only: Errors Successes

Graphs to Display: Data Average Median Deviation Throughput

473 ms

0 ms

No of Samples 200
Deviation 133

Latest Sample 548
Throughput 7,722.008/min

Average 375
Median 368

9:06 PM 12/4/2016

HTTP ebay.jmx (E:\SJSU\273\Lab1\HTTP ebay.jmx) - Apache JMeter (3.0 r1743807)

File Edit Search Run Options Help

Test Plan Thread Group HTTP Request Graph Results Summary Report Aggregate Graph View Results in Table WorkBench

Summary Report

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

Label	# Samples	Average	Min	Max	Std. Dev.	Error %	Throughput	KB/sec	Avg. Bytes
HTTP Request	200	375	133	660	133.98	0.00%	128.7/sec	329.74	2623.5
TOTAL	200	375	133	660	133.98	0.00%	128.7/sec	329.74	2623.5

Include group name in label Save Table Data Save Table Header

9:06 PM 12/4/2016

3) 300 concurrent users

HTTP ebay.jmx (E:\SJSU\273\Lab1\HTTP ebay.jmx) - Apache JMeter (3.0 r1743807)

File Edit Search Run Options Help

Test Plan Thread Group HTTP Request Graph Results Summary Report Aggregate Graph View Results in Table WorkBench

Graph Results

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

Graphs to Display Data Average Median Deviation Throughput

980 ms
900 ms
800 ms
700 ms
600 ms
500 ms
400 ms
300 ms
200 ms
100 ms
0 ms

No of Samples 300
Deviation 314
Latest Sample 1353
Throughput 8,126.411/min
Average 742
Median 800

9:08 PM 12/4/2016

HTTP ebay.jmx (E:\SJSU\273\Lab1\HTTP ebay.jmx) - Apache JMeter (3.0 r1743807)

File Edit Search Run Options Help

Test Plan Thread Group HTTP Request Graph Results Summary Report Aggregate Graph View Results in Table WorkBench

Summary Report

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

Label	# Samples	Average	Min	Max	Std. Dev.	Error %	Throughput	KB/sec	Avg. Bytes
HTTP Request	300	742	127	1426	314.43	0.00%	135.4/sec	347.05	2623.9
TOTAL	300	742	127	1426	314.43	0.00%	135.4/sec	347.05	2623.9

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

9:07 PM 12/4/2016

4) 400 concurrent users

HTTP ebay.jmx (E:\SJSU\273\Lab1\HTTP ebay.jmx) - Apache JMeter (3.0 r1743807)

File Edit Search Run Options Help

Test Plan Thread Group HTTP Request Graph Results Summary Report Aggregate Graph View Results in Table WorkBench

Graph Results

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

Graphs to Display Data Average Median Deviation Throughput

1101 ms
0 ms

No of Samples 400
Deviation 371
Latest Sample 1288
Throughput 10,582.011/minute

Average 889
Median 957

9:08 PM 12/4/2016

HTTP ebay.jmx (E:\SJSU\273\Lab1\HTTP ebay.jmx) - Apache JMeter (3.0 r1743807)

File Edit Search Run Options Help

Test Plan Thread Group
HTTP Request
Graph Results Summary Report Aggregate Graph View Results in Table WorkBench

Summary Report

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

Label	# Samples	Average	Min	Max	Std. Dev.	Error %	Throughput	KB/sec	Avg. Bytes
HTTP Request	400	889	162	1456	371.77	0.00%	176.4/sec	451.82	2623.3
TOTAL	400	889	162	1456	371.77	0.00%	176.4/sec	451.82	2623.3

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

9:08 PM 12/4/2016

5)500 concurrent Users

HTTP ebay.jmx (E:\SJSU\273\Lab1\HTTP ebay.jmx) - Apache JMeter (3.0 r1743807)

File Edit Search Run Options Help

Test Plan Thread Group
HTTP Request
Graph Results Summary Report Aggregate Graph View Results in Table WorkBench

Graph Results

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

Graphs to Display Data Average Median Deviation Throughput

1796 ms
0 ms
No of Samples 500
Deviation 484
Latest Sample 1962
Throughput 9,842.52/minute
Average 1435
Median 1547

9:10 PM 12/4/2016

HTTP ebay.jmx (E:\SJSU\273\Lab1\HTTP ebay.jmx) - Apache JMeter (3.0 r1743807)

File Edit Search Run Options Help

Test Plan Thread Group HTTP Request Graph Results Summary Report Aggregate Graph View Results in Table WorkBench

Summary Report

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

Label	# Samples	Average	Min	Max	Std. Dev.	Error %	Throughput	KB/sec	Avg. Bytes
HTTP Request	500	1435	146	2078	484.22	0.00%	164.0/sec	420.28	2623.5
TOTAL	500	1435	146	2078	484.22	0.00%	164.0/sec	420.28	2623.5

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

9:09 PM 12/4/2016

3. Base + RabbitMQ

1) 100 concurrent users

HTTP ebay.jmx (E:\SJSU\273\Lab1\HTTP ebay.jmx) - Apache JMeter (3.0 r1743807)

File Edit Search Run Options Help

Test Plan Thread Group HTTP Request Graph Results Summary Report Aggregate Graph View Results in Table WorkBench

Graph Results

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

Graphs to Display Data Average Median Deviation Throughput

467 ms
0 ms

No of Samples 100
Deviation 102
Latest Sample 193
Throughput 5,089.059/minute
Average 287
Median 253

9:01 PM 12/4/2016

HTTP ebay.jmx (E:\SJSU\273\Lab1\HTTP ebay.jmx) - Apache JMeter (3.0 r1743807)

File Edit Search Run Options Help

Test Plan Thread Group
HTTP Request
Graph Results Summary Report Aggregate Graph View Results in Table WorkBench

Summary Report

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

Label	# Samples	Average	Min	Max	Std. Dev.	Error %	Throughput	KB/sec	Avg. Bytes
HTTP Request	100	287	154	487	102.20	0.00%	84.8/sec	217.29	2623.3
TOTAL	100	287	154	487	102.20	0.00%	84.8/sec	217.29	2623.3

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

Ask me anything 6:01 PM 12/4/2016

2) 200 concurrent users

HTTP ebay.jmx (E:\SJSU\273\Lab1\HTTP ebay.jmx) - Apache JMeter (3.0 r1743807)

File Edit Search Run Options Help

Test Plan Thread Group
HTTP Request
Graph Results Summary Report Aggregate Graph View Results in Table WorkBench

Graph Results

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

Graphs to Display Data Average Median Deviation Throughput

473 ms
0 ms
No. of Samples 200
Deviation 108
Latest Sample 420
Throughput 8,350.731/min
Average 383
Median 408

Ask me anything 6:02 PM 12/4/2016

HTTP ebay.jmx (E:\SJSU\273\Lab1\HTTP ebay.jmx) - Apache JMeter (3.0 r1743807)

File Edit Search Run Options Help

Test Plan Thread Group
HTTP Request
Graph Results Summary Report Aggregate Graph View Results in Table WorkBench

Summary Report

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

Label	# Samples	Average	Min	Max	Std. Dev.	Error %	Throughput	KB/sec	Avg. Bytes
HTTP Request	200	383	116	576	108.23	0.00%	139.2/sec	356.63	2623.9
TOTAL	200	383	116	576	108.23	0.00%	139.2/sec	356.63	2623.9

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

Ask me anything 6:01 PM 12/4/2016

3) 300 concurrent users

HTTP ebay.jmx (E:\SJSU\273\Lab1\HTTP ebay.jmx) - Apache JMeter (3.0 r1743807)

File Edit Search Run Options Help

Test Plan Thread Group
HTTP Request
Graph Results Summary Report Aggregate Graph View Results in Table WorkBench

Graph Results

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

Graphs to Display Data Average Median Deviation Throughput

849 ms
0 ms
No. of Samples 300
Deviation 204
Latest Sample 797
Throughput 10,204.082/min
Average 630
Median 685

Ask me anything 6:02 PM 12/4/2016

HTTP ebay.jmx (E:\SJSU\273\Lab1\HTTP ebay.jmx) - Apache JMeter (3.0 r1743807)

File Edit Search Run Options Help

Test Plan Thread Group
HTTP Request
Graph Results Summary Report Aggregate Graph View Results in Table WorkBench

Summary Report

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

Label	# Samples	Average	Min	Max	Std. Dev.	Error %	Throughput	KB/sec	Avg. Bytes
HTTP Request	300	630	132	959	204.27	0.00%	170.1/sec	435.75	2623.7
TOTAL	300	630	132	959	204.27	0.00%	170.1/sec	435.75	2623.7

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

Ask me anything 6:02 PM 12/4/2016

4) 400 concurrent users

HTTP ebay.jmx (E:\SJSU\273\Lab1\HTTP ebay.jmx) - Apache JMeter (3.0 r1743807)

File Edit Search Run Options Help

Test Plan Thread Group
HTTP Request
Graph Results Summary Report Aggregate Graph View Results in Table WorkBench

Graph Results

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

Graphs to Display Data Average Median Deviation Throughput

1070 ms
0 ms
No of Samples 400
Deviation 251
Latest Sample 1201
Throughput 10,820.559/min
Average 1027
Median 1061

Ask me anything 6:26 PM 12/4/2016

HTTP ebay.jmx (E:\SJSU\273\Lab1\HTTP ebay.jmx) - Apache JMeter (3.0 r1743807)

File Edit Search Run Options Help

Test Plan Thread Group HTTP Request Graph Results Summary Report Aggregate Graph View Results in Table WorkBench

Summary Report

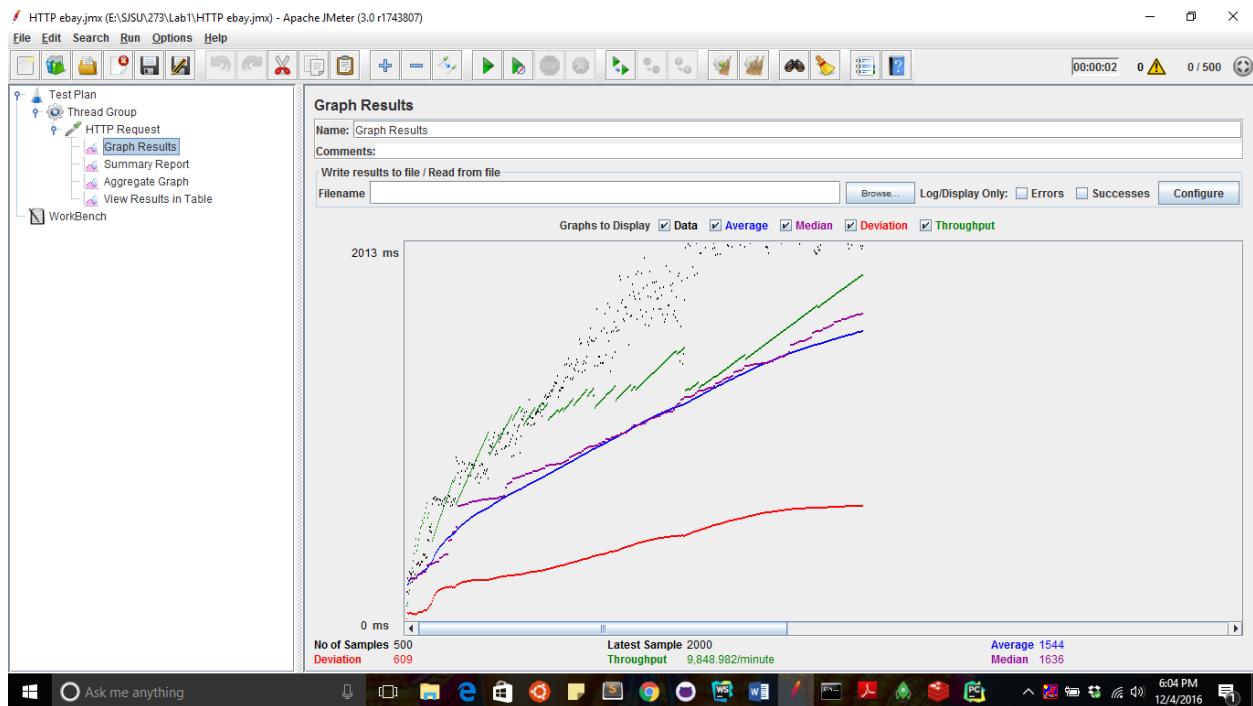
Name: Summary Report
Comments:
Write results to file / Read from file
Filename Browse... Log/Display Only: Errors Successes Configure

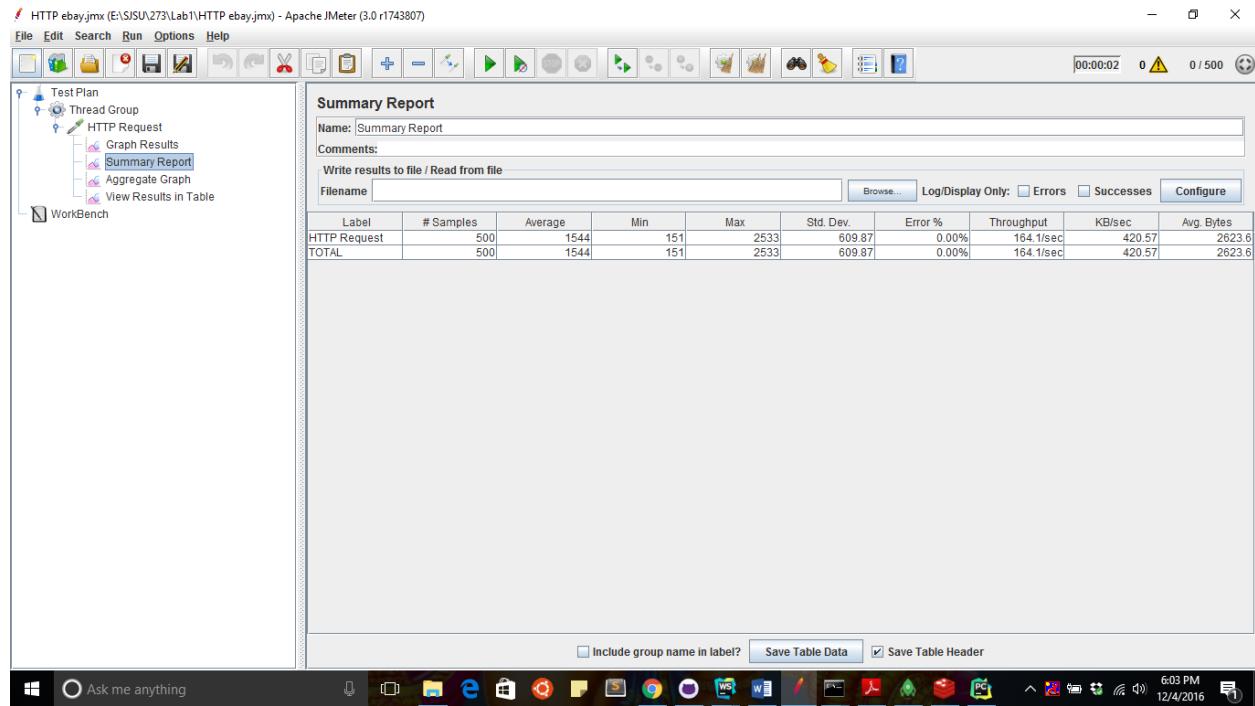
Label	# Samples	Average	Min	Max	Std. Dev.	Error %	Throughput	KB/sec	Avg. Bytes
HTTP Request	400	1027	145	1416	251.27	0.00%	180.3/sec	462.05	2623.6
TOTAL	400	1027	145	1416	251.27	0.00%	180.3/sec	462.05	2623.6

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

6:26 PM 12/4/2016

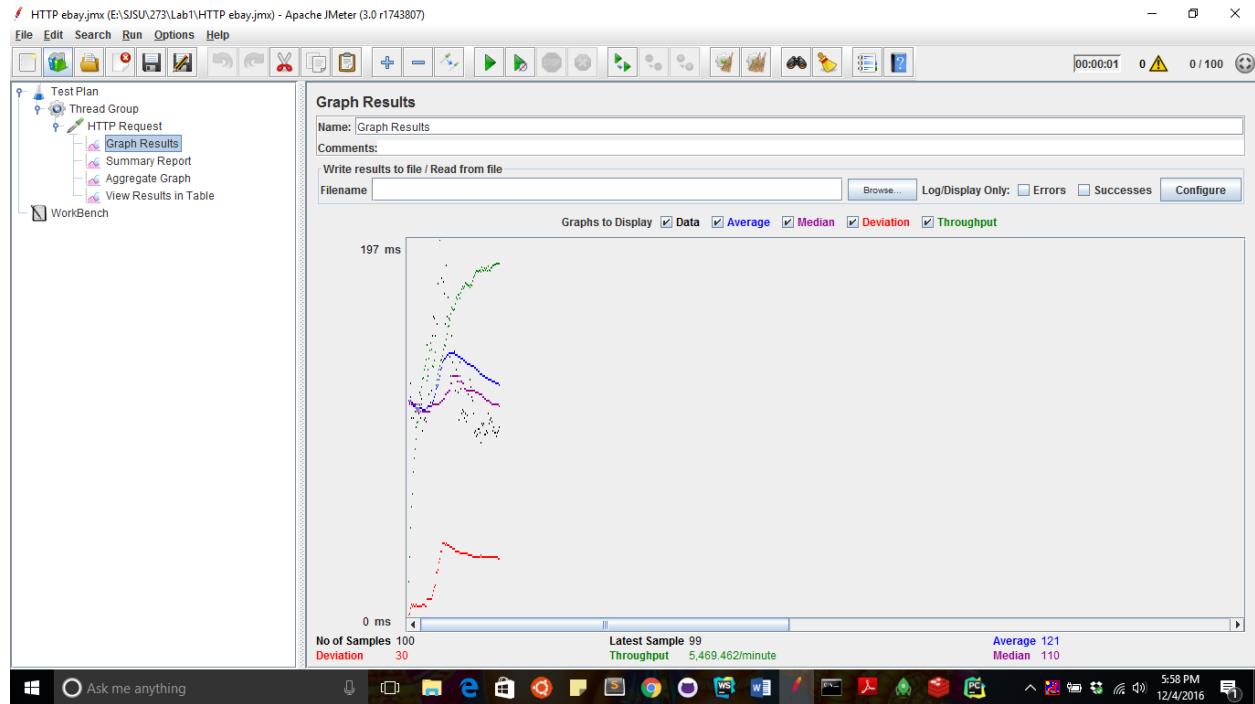
5)500 concurrent Users





4. Base + RabbitMQ+ Connection Pooling

1) 100 concurrent users



HTTP ebay.jmx (E:\SJSU\273\Lab1\HTTP ebay.jmx) - Apache JMeter (3.0 r1743807)

File Edit Search Run Options Help

Test Plan Thread Group HTTP Request Graph Results Summary Report Aggregate Graph View Results in Table WorkBench

Summary Report

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

Label	# Samples	Average	Min	Max	Std. Dev	Error %	Throughput	KB/sec	Avg. Bytes
HTTP Request	100	121	91	232	30.89	0.00%	91.2/sec	233.57	2623.7
TOTAL	100	121	91	232	30.89	0.00%	91.2/sec	233.57	2623.7

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

5:57 PM 12/4/2016

2) 200 concurrent users

HTTP ebay.jmx (E:\SJSU\273\Lab1\HTTP ebay.jmx) - Apache JMeter (3.0 r1743807)

File Edit Search Run Options Help

Test Plan Thread Group HTTP Request Graph Results Summary Report Aggregate Graph View Results in Table WorkBench

Graph Results

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

Graphs to Display Data Average Median Deviation Throughput

577 ms

0 ms

No of Samples 200
Deviation 199

Latest Sample 286
Throughput 9,302 326/minute

Average 355
Median 373

5:54 PM 12/4/2016

HTTP ebay.jmx (E:\SISU\273\Lab1\HTTP ebay.jmx) - Apache JMeter (3.0 r1743807)

File Edit Search Run Options Help

Test Plan Thread Group HTTP Request Graph Results Summary Report Aggregate Graph View Results in Table WorkBench

Summary Report

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

Label	# Samples	Average	Min	Max	Std. Dev.	Error %	Throughput	KB/sec	Avg. Bytes
HTTP Request	200	355	97	710	199.94	0.00%	155.0/sec	397.21	2623.5
TOTAL	200	355	97	710	199.94	0.00%	155.0/sec	397.21	2623.5

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

5:57 PM 12/4/2016

3) 300 concurrent users

HTTP ebay.jmx (E:\SISU\273\Lab1\HTTP ebay.jmx) - Apache JMeter (3.0 r1743807)

File Edit Search Run Options Help

Test Plan Thread Group HTTP Request Graph Results Summary Report Aggregate Graph View Results in Table WorkBench

Graph Results

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

Graphs to Display Data Average Median Deviation Throughput

724 ms

0 ms

No of Samples 300 Deviation 193

Latest Sample 748 Throughput 10,676.157/minute

Average 539 Median 584

5:58 PM 12/4/2016

HTTP ebay.jmx (E:\SISU\273\Lab1\HTTP ebay.jmx) - Apache JMeter (3.0 r1743807)

File Edit Search Run Options Help

00:00:01 0 0 / 300

Test Plan

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

Summary Report

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

Label	# Samples	Average	Min	Max	Std. Dev.	Error %	Throughput	KB/sec	Avg. Bytes
HTTP Request	300	539	149	933	193.94	0.00%	177.9/sec	455.87	2623.5
TOTAL	300	539	149	933	193.94	0.00%	177.9/sec	455.87	2623.5

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

Ask me anything 5:58 PM 12/4/2016

4) 400 concurrent users

HTTP ebay.jmx (E:\SISU\273\Lab1\HTTP ebay.jmx) - Apache JMeter (3.0 r1743807)

File Edit Search Run Options Help

00:00:01 0 0 / 400

Test Plan

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

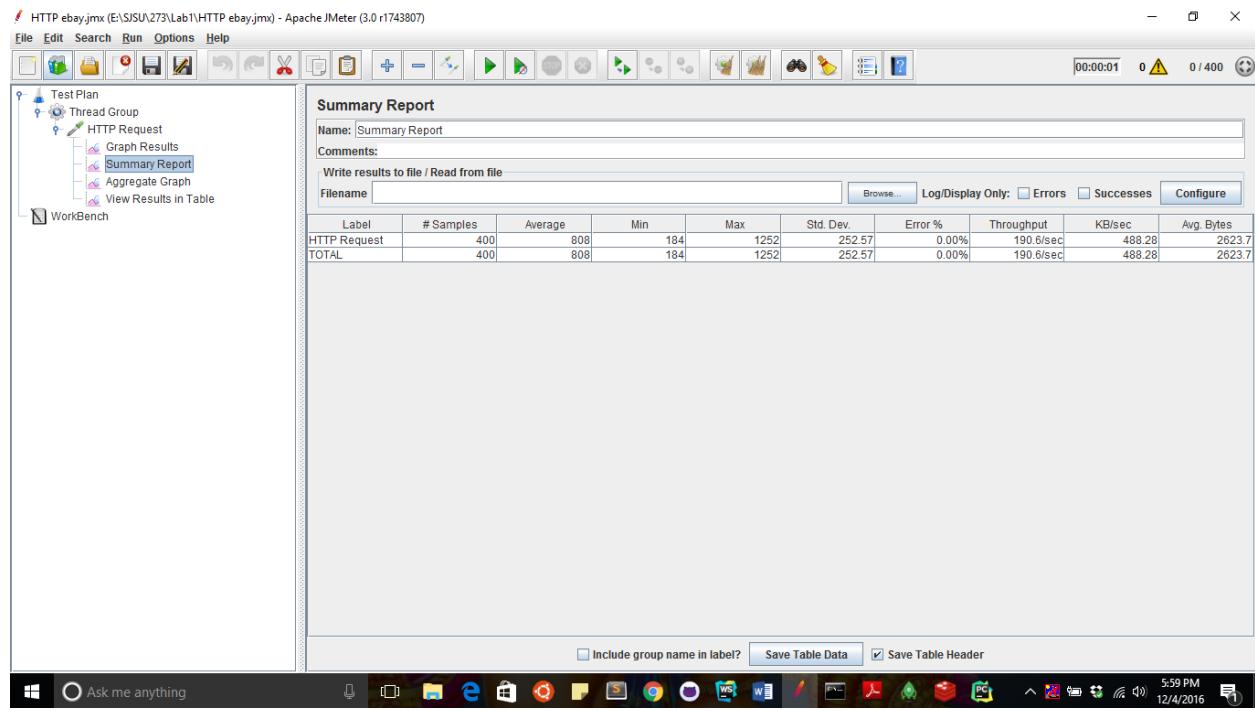
Graph Results

Name: [Graph Results]
Comments:
Write results to file / Read from file
Filename Browse... Log/Display Only: Errors Successes Configure

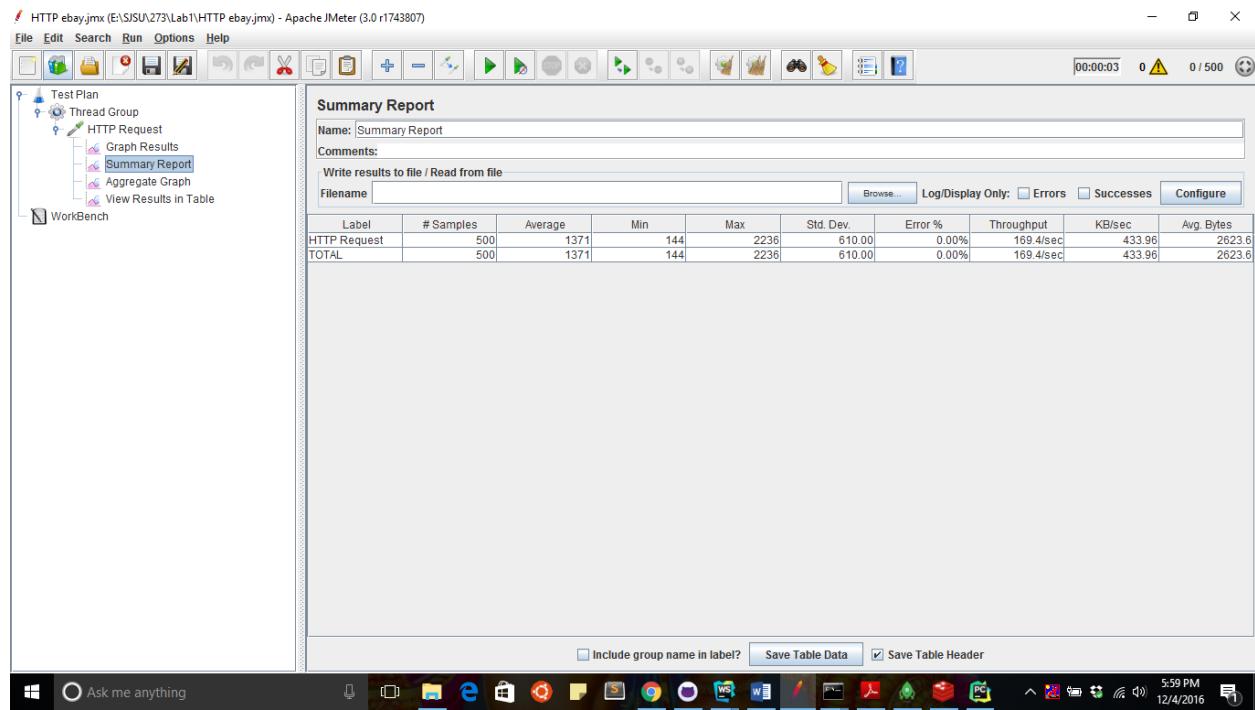
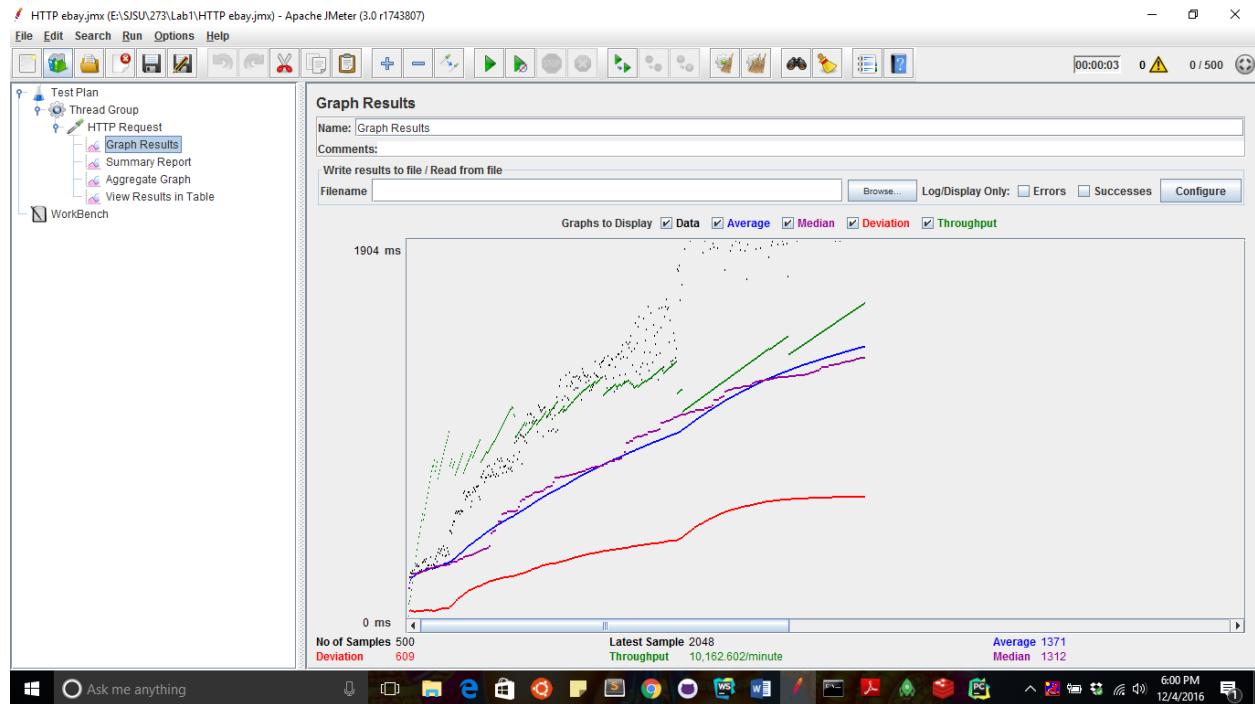
Graphs to Display: Data Average Median Deviation Throughput

1071 ms
0 ms
No of Samples 400
Deviation 252
Latest Sample 1178
Throughput 11,434.016/min
Average 808
Median 834

Ask me anything 5:59 PM 12/4/2016

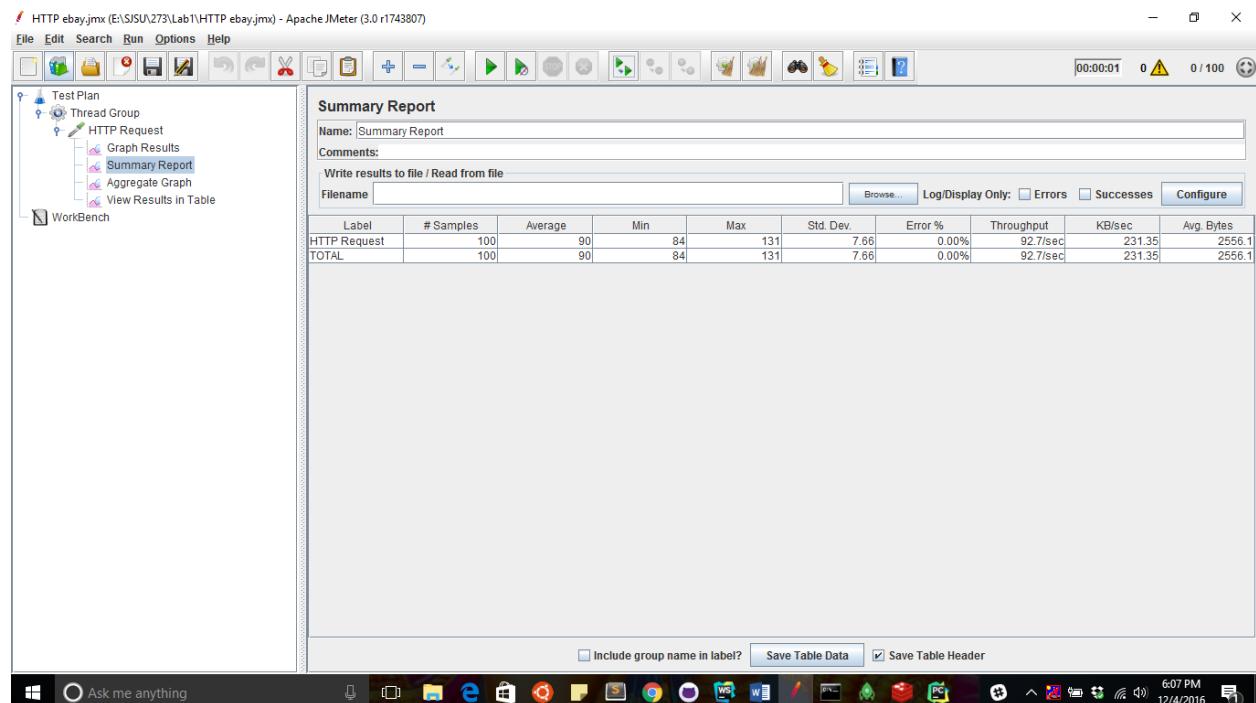
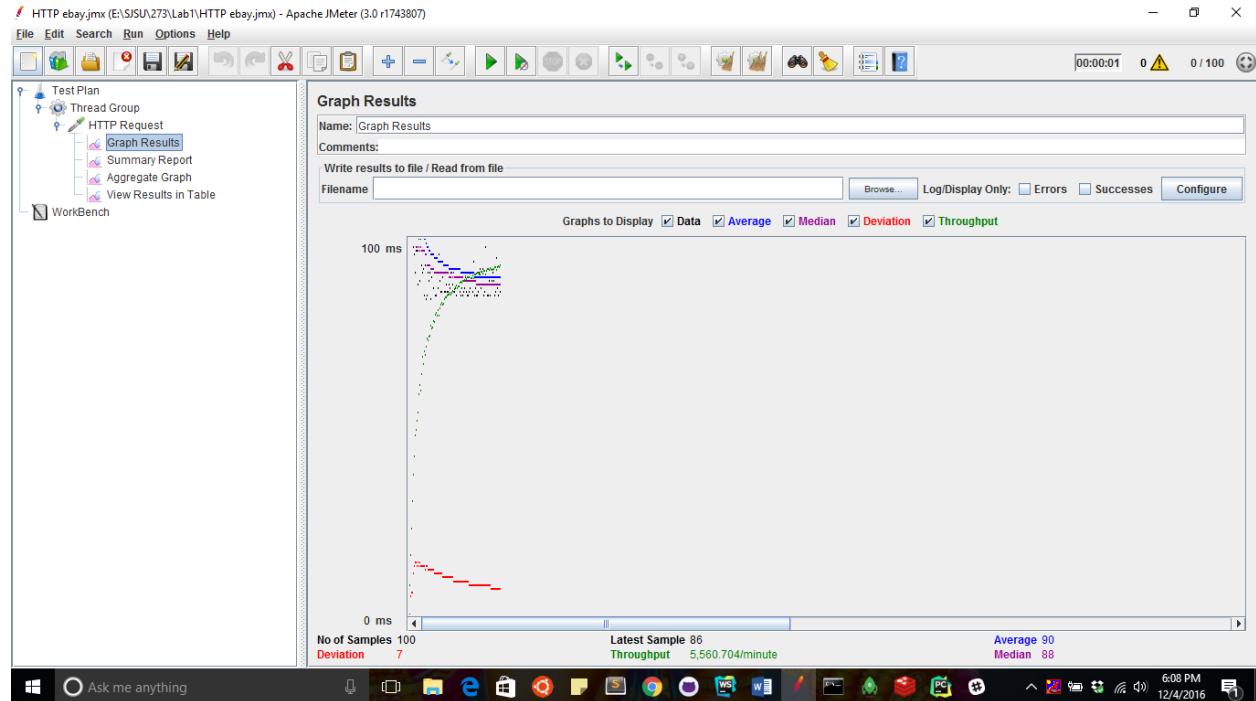


5)500 concurrent Users

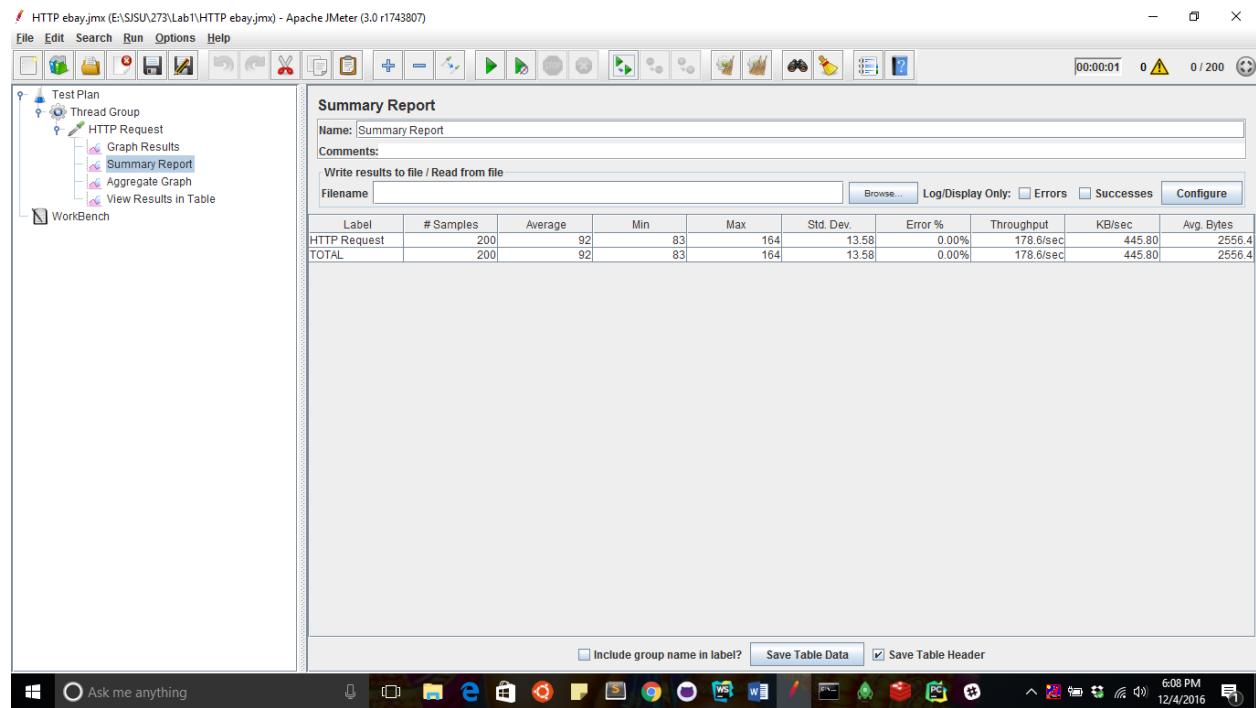
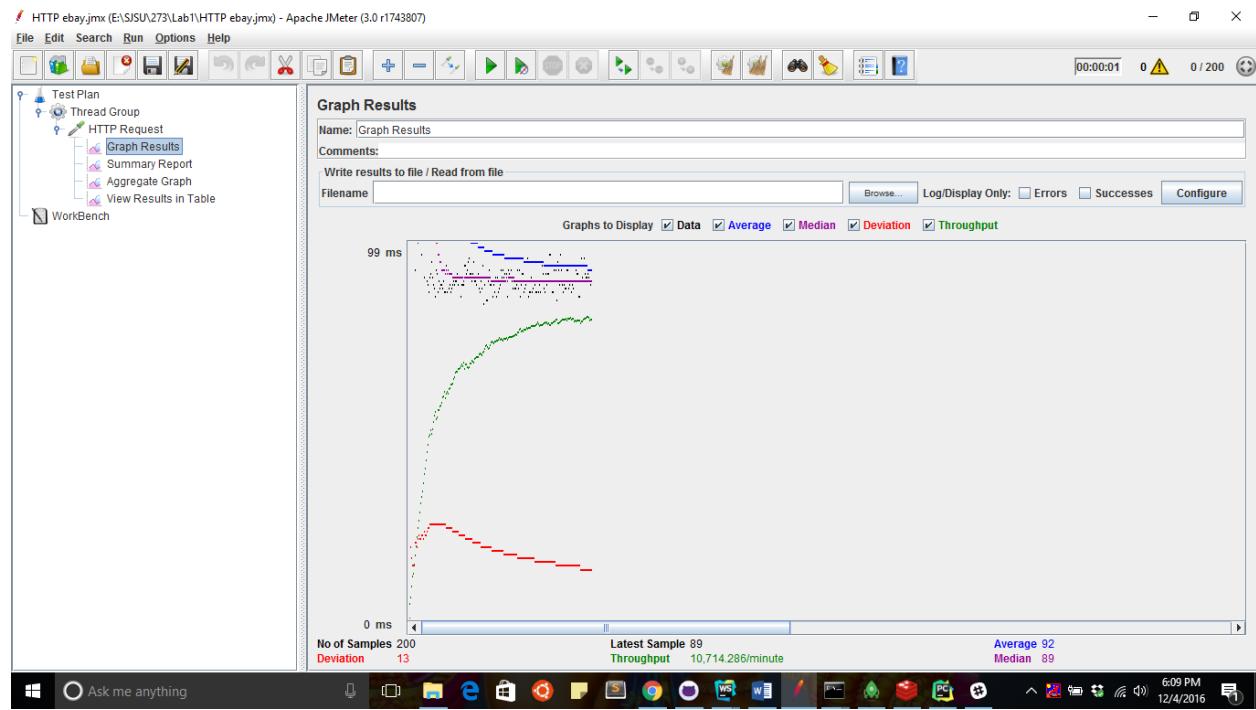


5. Base + RabbitMQ + Connection Pooling + Redis

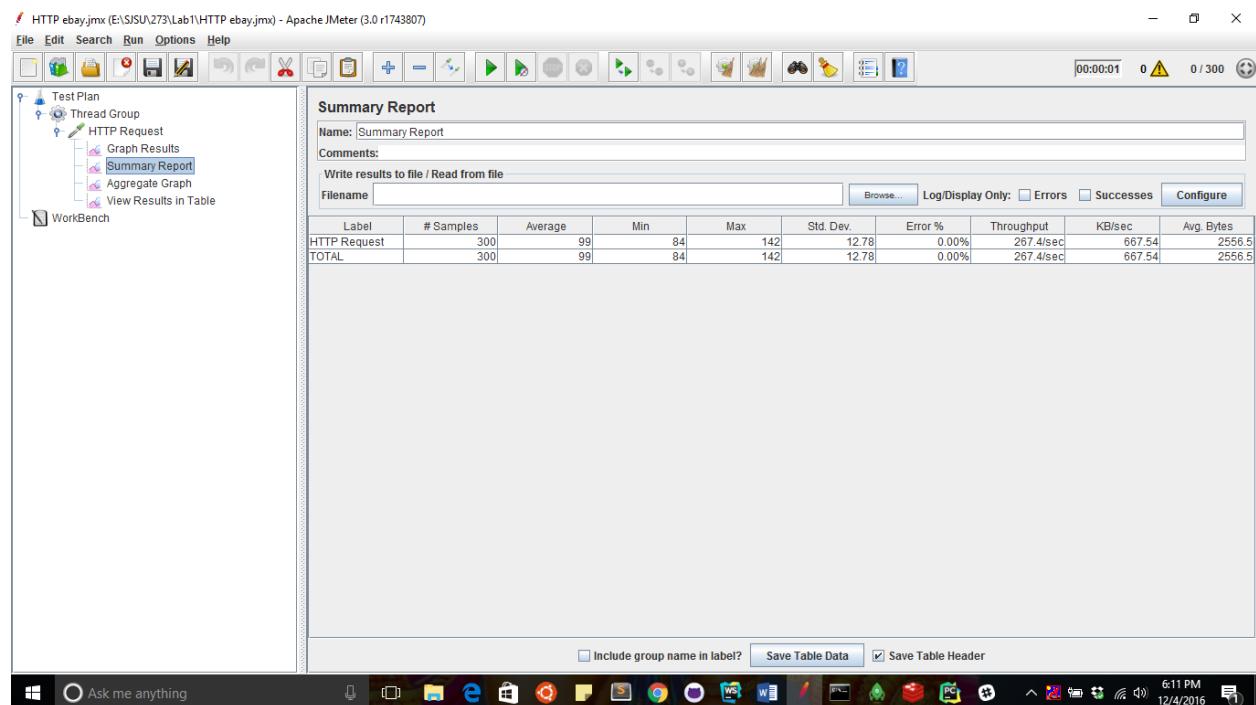
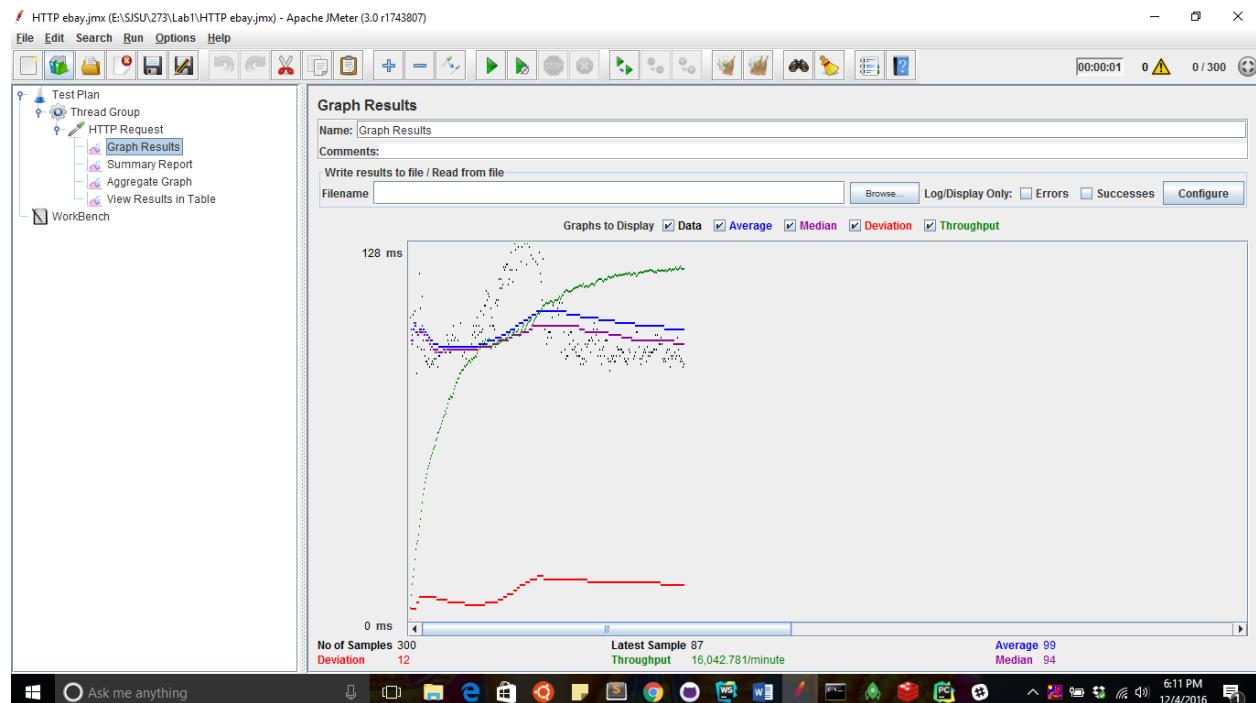
1) 100 concurrent users



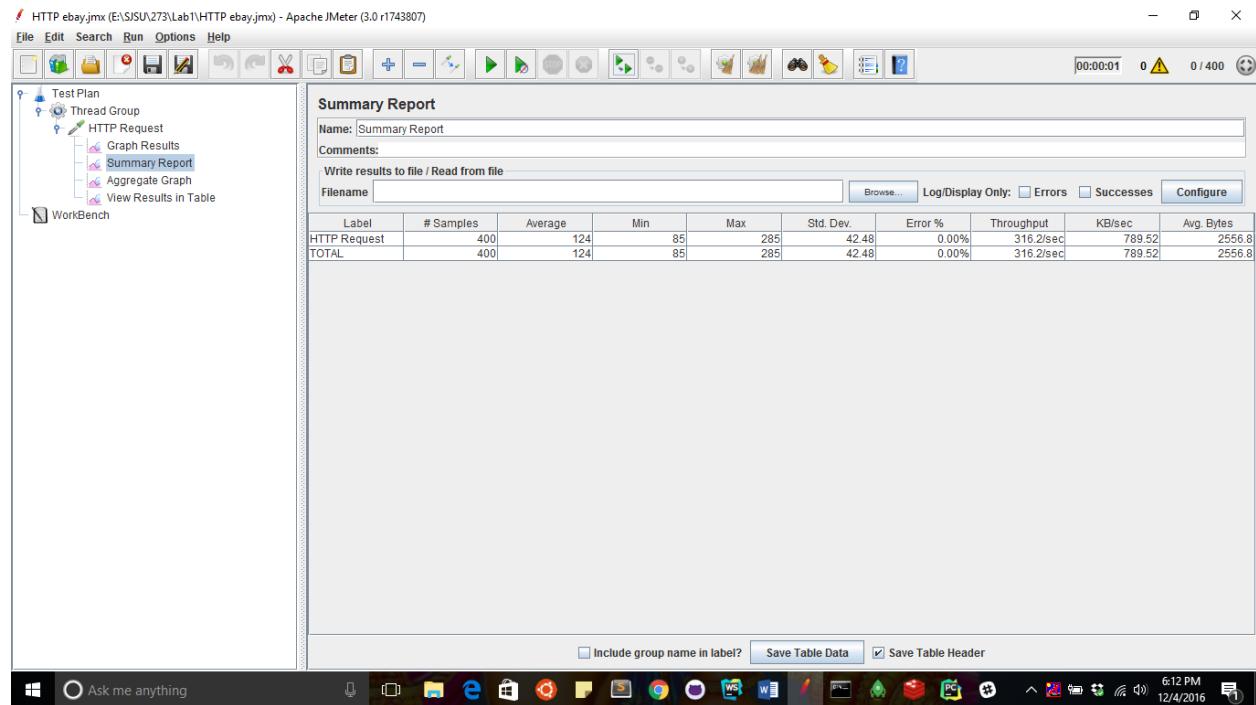
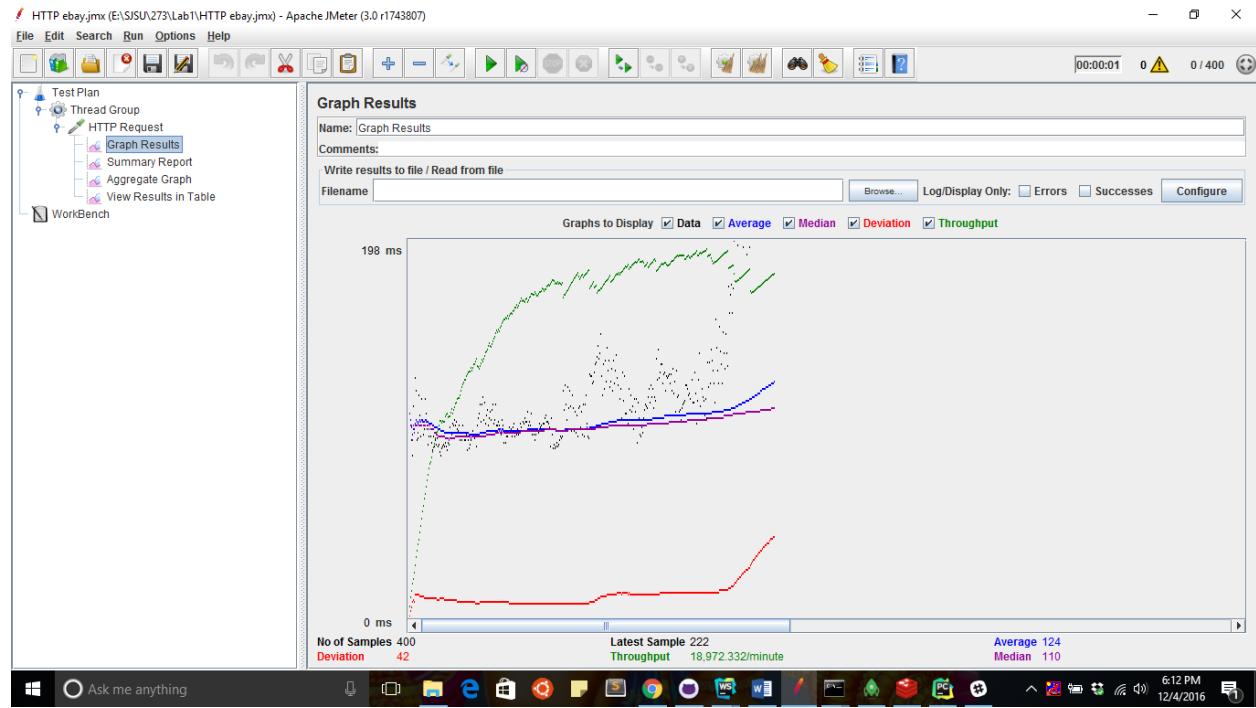
2) 200 concurrent users



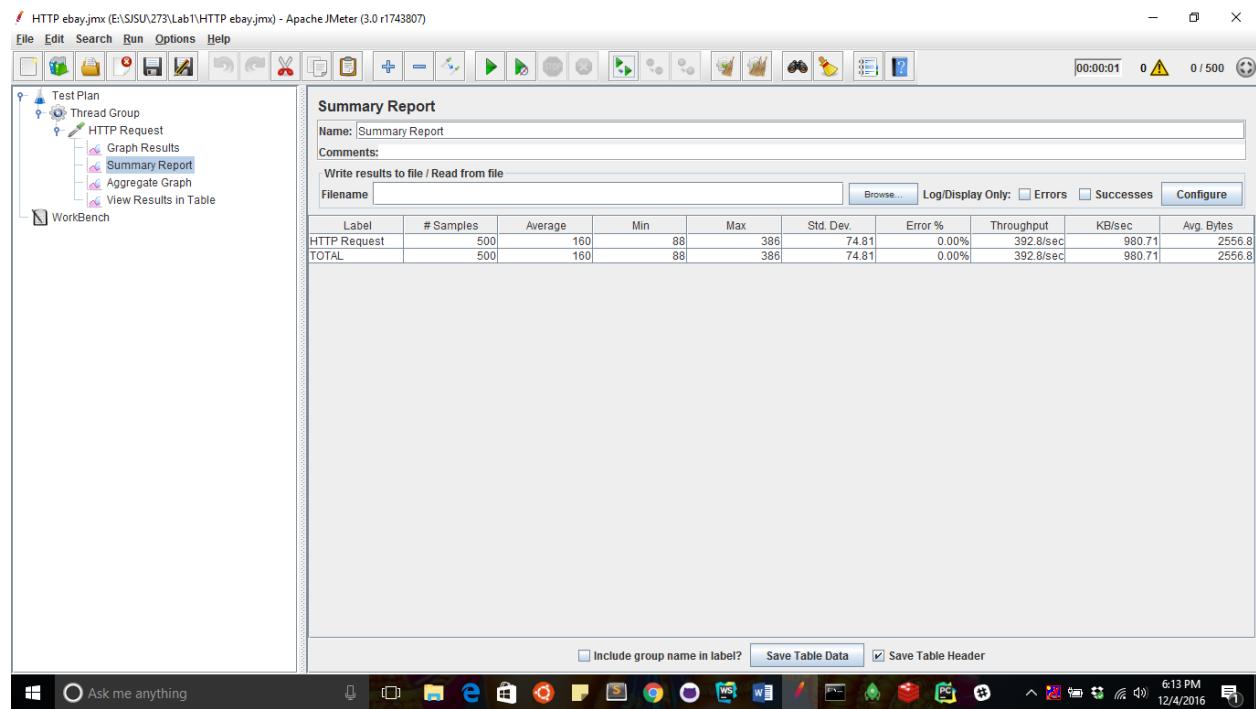
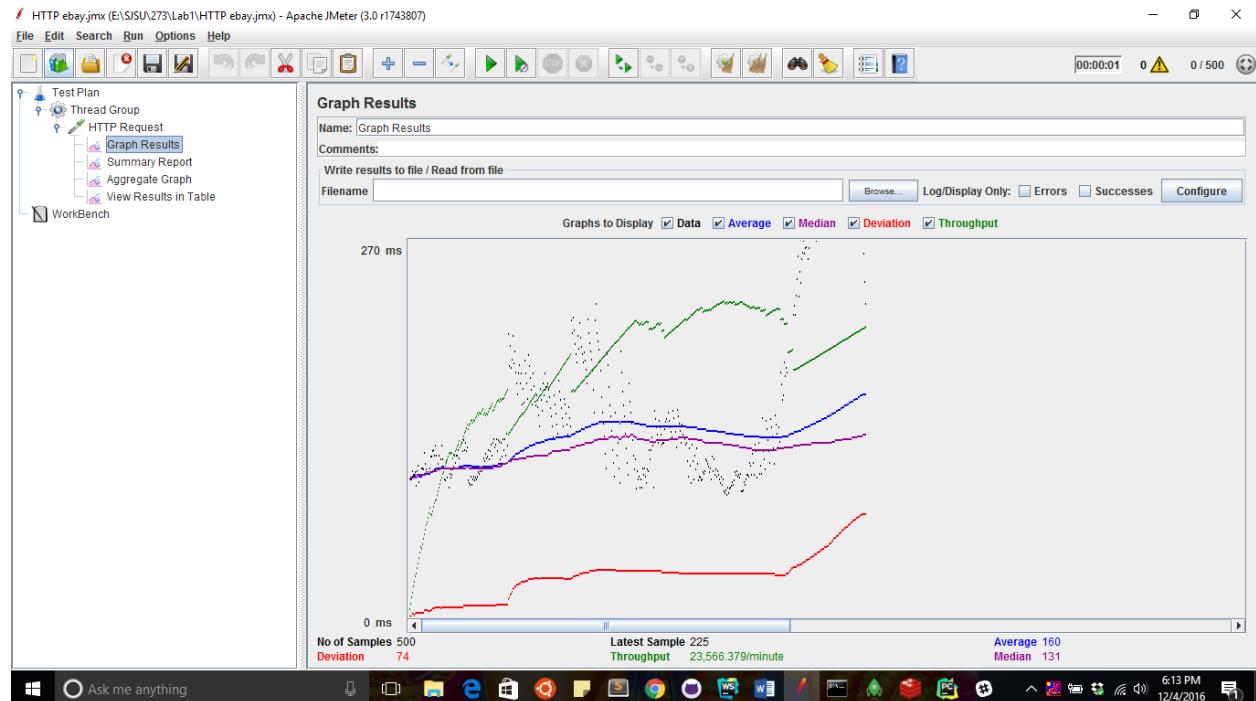
3) 300 concurrent users



4) 400 concurrent users



5) 500 concurrent Users

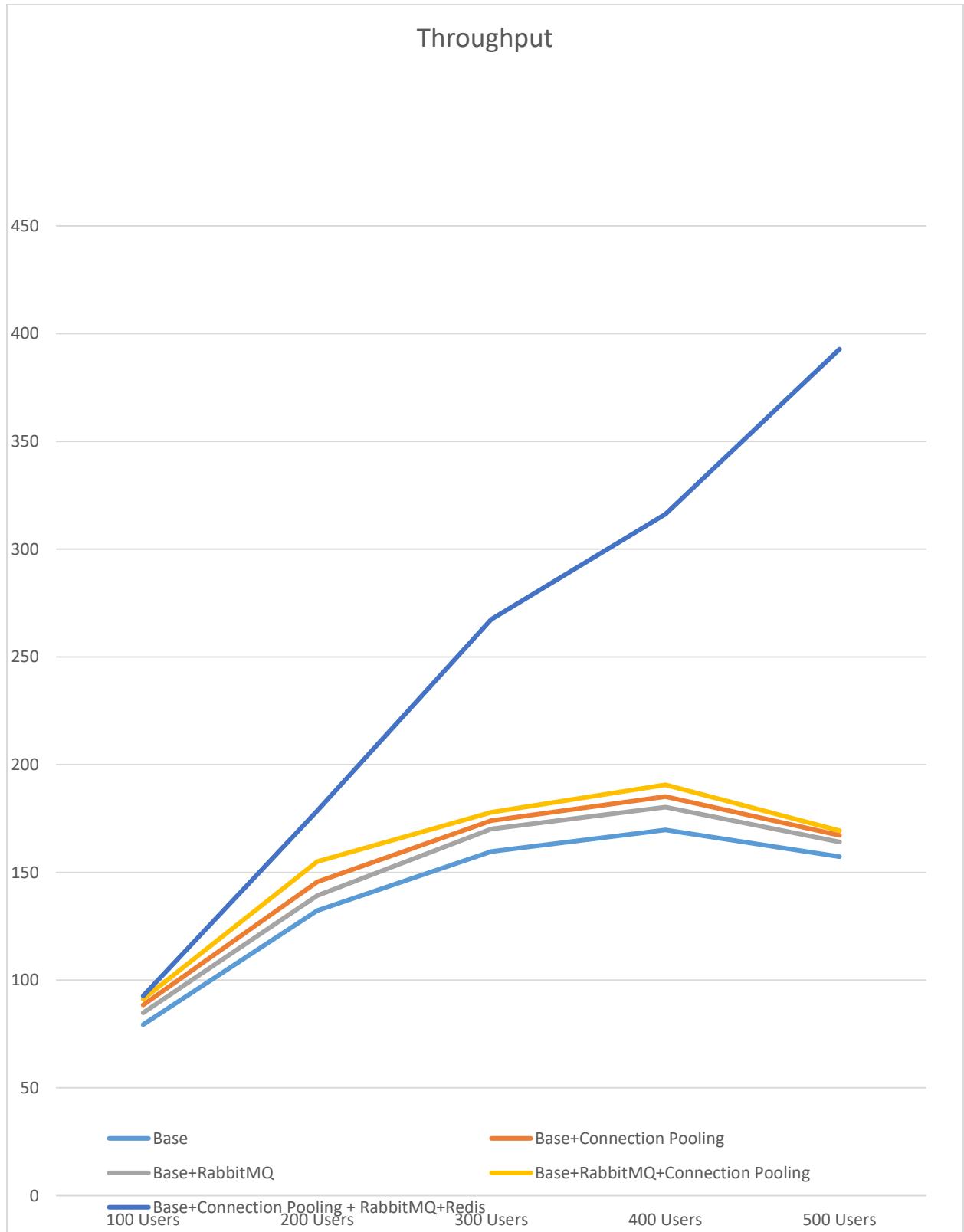


Performance Analysis

1. Throughput Analysis

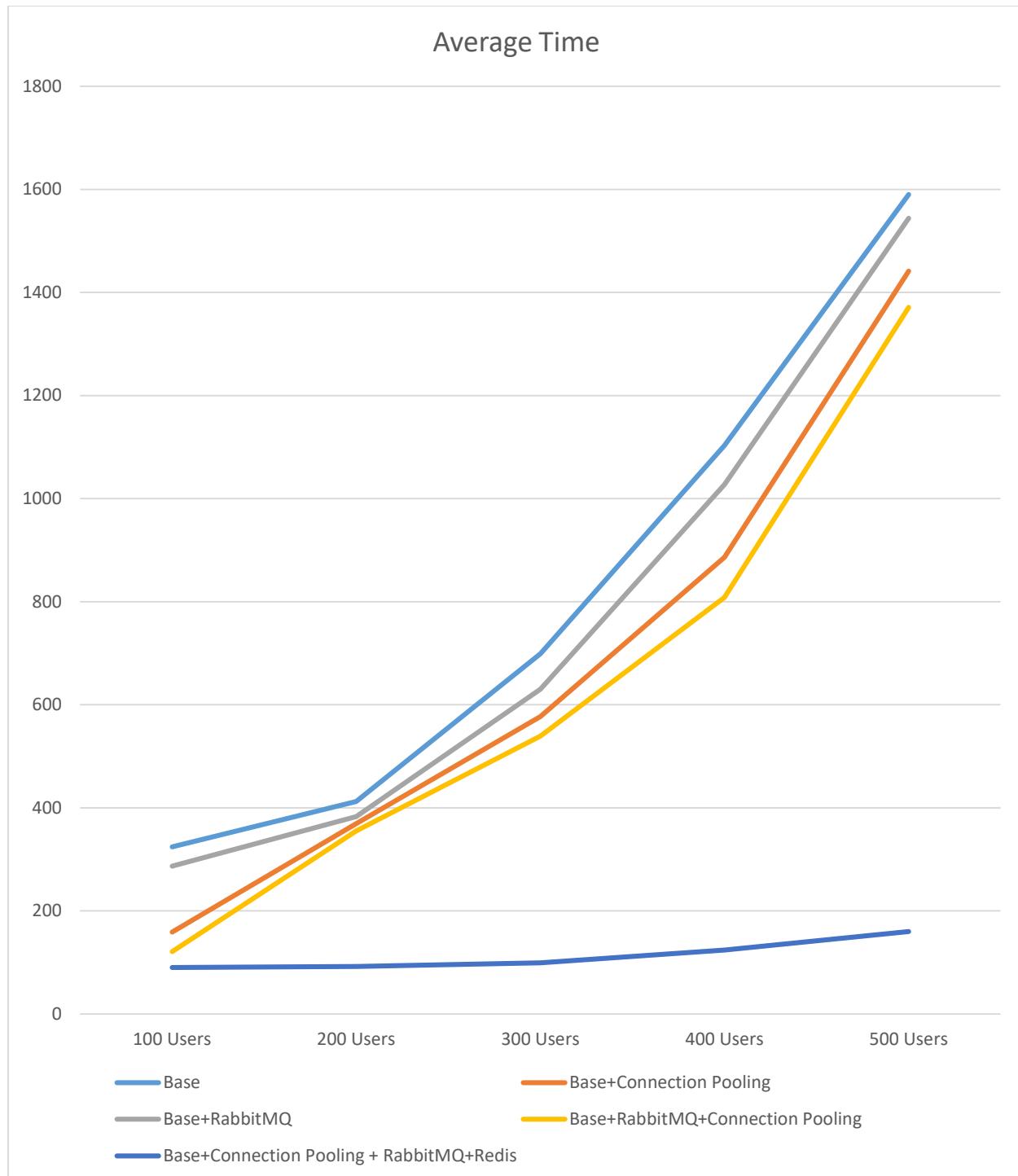
-Y axis shows throughput per second

-X axis shows number of concurrent users



2. Average Time Analysis

-Y axis shows Average Time in seconds
-X axis shows number of concurrent users



Observations and Lessons learnt from the Project

Airbnb Application helped us to develop an efficient, robust and a scalable web application.

- We developed RESTful stateless services using MEAN stack.
- The connection pooling in the MongoDB database helped us to optimize the performance.
- RabbitMQ helped us in providing scalability and handling multiple concurrent requests at the same time.
- We used Redis for cache management.
- We learned ELK stack to develop the Analytics module.
- JMeter and Mocha helped us to continuously analyze the performance in different scenarios for all the APIs.