

A dark blue vertical bar on the left side of the page. A blue arrow points to the right from the bar, containing the text 'CMPE 273', 'ENTERPRISE DISTRIBUTIVE', and 'SYSTEMS'.

CMPE 273

ENTERPRISE DISTRIBUTIVE  
SYSTEMS

# TEAM PROJECT

# KAYAK PROTOTYPE

## TEAM 11

<https://github.com/asbharadiya/kayak>

AMAN OJHA

ANKIT BHARADIYA

MAULIK BHATT

PALASH HEDAU

PRATEEK SHARMA

# Contents

ABOUT.....	3
KAYAK.....	3
CONTRIBUTION .....	4
1.    AMAN OJHA.....	4
2.    ANKIT BHARADIYA .....	4
3.    MAULIK BHATT .....	4
4.    PALASH HEDAU .....	4
5.    PRATEEK SHARMA .....	4
ABOUT THE PROJECT .....	5
1.    Object Management Policy .....	5
2.    “Heavyweight” Resource Handling .....	6
3.    Policy of writing data into the Database .....	7
ADMIN APPLICATION .....	8
Admin Section .....	8
CUSTOMER APPLICATION .....	18
Client Section .....	18
CODE LISTING.....	30
Client Frontend .....	30
Admin Frontend.....	32
Node Backend .....	33
Kafka Backend.....	34
Mocha Testing .....	35
JMETER TESTING .....	36
1.    Listing .....	36
2.    User Registration .....	37
3.    Bookings .....	39
4.    JMeter conclusion .....	40
MOCHA TESTING.....	41
DATABASE SCHEMA .....	43
1.    MongoDB:.....	44
2.    MySQL: .....	44

OBSERVATIONS & LESSONS LEARNED .....	45
Observations .....	45
Lessons Learned.....	45

# ABOUT

## KAYAK

This is a prototype of the website, Kayak.com which is quite popular for all the travel needs of a customer. The website can be used to book flight, car or hotels from other service providers. Kayak acts as an aggregator in between and can also be used for comparing rates on different websites.



KAYAK was founded in 2004 by Steve Hafner and Paul M. English. Being available in over 18 languages, this website is widely used across many countries. It is particularly used as a meta search engine for finding flight, cars and hotel bookings.

In our application, the front end is created using React, HTML5, Bootstrap and CSS. After logging in, the user gets various options like search cars, flight and hotels along with booking and payment options. All these requests are sent to the server where various RESTful APIs are present to handle the requests. The backend is designed using NodeJS and ExpressJS.

From the RESTful api, the requests are further sent to the Messaging Queue backend where the exact logic for data handling is present. For our application, we are using Kafka messaging queue.

# CONTRIBUTION

## 1. AMAN OJHA

- Developed APIs for profile page in frontend as well as backend.
- Flight listing and API development for admin section.
- Report creation.
- JMeter load testing along with Mocha testing.
- Bug fixing and UI testing.

## 2. ANKIT BHARADIYA

- Set up git repository and base project structure
- Set up Kafka queue and topics
- Defining API endpoints and main flow of data of entire project
- Handling common CSS classes and responsiveness of webpages
- Developed user Home module with search bar, payment module and common modals
- Connection pooling implementation

## 3. MAULIK BHATT

- Hotel listing and API development for admin section.
- Analytics charts and user tracking in admin section.
- Listing and API development of Billing section in admin side.
- Research and implementation of Mongoose ODM.
- Result sorting in each type of listings on user side.

## 4. PALASH HEDAU

- Car listing and API development for admin section.
- Admin's user listing module with search.
- Search and Filters functionality across client and admin module.
- Handling complex queries which involved nested updating for billing and booking module.
- Cars booking frontend and billing interface. User past booking module
- Redis setup and integration.

## 5. PRATEEK SHARMA

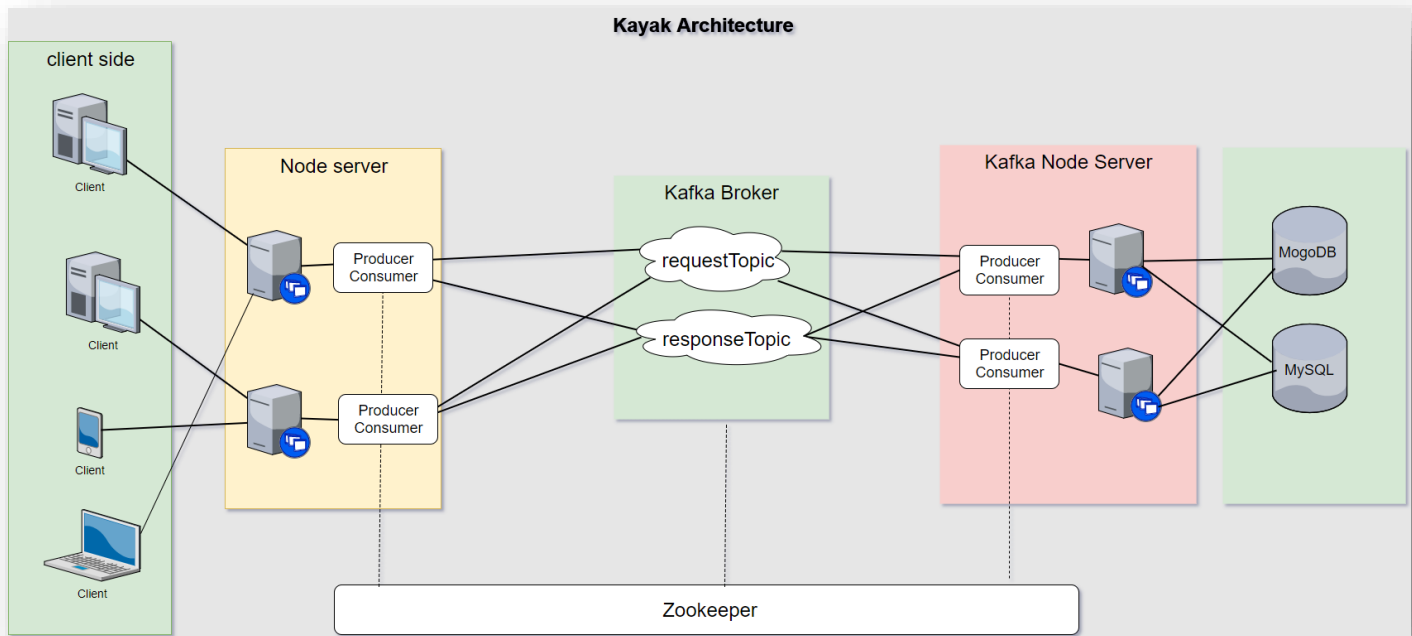
- Passport authentication setup for admin as well as clients.
- Hotel listing and API development for client website.
- Implementation of user analytics to be displayed on client side.
- Developing CRON jobs and data logging which was later used in analytics section.
- Bug fixing and performance improvement on the go.

# ABOUT THE PROJECT

## 1. Object Management Policy

Kayak is a vast project in its own with various functionalities like managing listing of flights, cars and hotels. It also handles booking information, managing user's information, billing etc. Because of the multi offerings, we made the project modular.

The project's architecture is as follows:



## Admin Frontend:

The first module is admin frontend which can be only accessed with admin privileges. It is a centralized console from where an admin can control all the data to be displayed on the website. The admin will have rights to add, delete, modify data of flights, cars and hotels. While adding car, flight or hotel admin can add different details of each category i.e. name, address, city, source, destination, room availability, booking dates etc and upload pictures also.

The admin will also have access to an analytics page which will provide various analysis of gathered data of how the website is being used. Through analytics report and charts, admin can see share of each category hotel, flight and car in total revenue and orders. Admin can also view highest revenue generating companies in last year and last month, and city wise revenue. Admin also access track of user and most viewed or clicked listing.

The admin will also have access to listing of all the users registered on the website and can delete them if needed. It will also have information regarding billings.

## Client Frontend:

The client frontend is the actual web application which will be used by the end user. After landing on the home page, the user will have options to search for cars, hotels and flights. While searching we provide suggestions of the cities of USA according to entered string in input box. The results will be displayed according to the user's selection criteria. User can change filter criteria on the fly and new results will be displayed. User can also sort the results (ascending/descending) based on price, rating, stars, departure time. Also, if more than 20 results come then load more option becomes visible to access results on next page.

After selecting a option, the user will be redirected to the booking page where he has to fill in all the information required according to listing category. Once all the information is validated, user will proceed to checkout. On successful payment, billing will be done.

User can also see his/her profile, past bookings and register payment methods. User also can update his/her person details and can also upload profile picture if want to. User also can register multiple payment methods which will be provided next while billing.

## Node backend:

The node backend server is responsible for handling all the requests made by the above two front end services. It is the first point of contact which routes all the api calls to its respective handler. The first thing after receiving a request from the front end is to check whether the request is from a validated user or not. All these requests are handled by passport. After validation, the requests are passed on to the kafka node backend using kafka topics.

## Kafka server:

The kafka server maintains topics which are also known as messaging queues. All the requests are passed via the request topic and received via the response topic. All the requests sent in the topic are associated with a unique id for identification purpose. These requests are further sent to the kafka node back end where the actual logic for data handling is present.

## Kafka node backend server:

Once a request is received by the kafka back end, we perform the desired operation or fetch the required data from databases such as MongoDB or MYSQL. The required data is then further sent to the node back end via kafka server in response topic. Once the node back end receives the data, it sends the data back to the front end where the data is displayed to the user.

## 2. "Heavyweight" Resource Handling

As it is known that performing queries on the database requires costly resources, it is very essential to limit these kinds of activities. We managed our resources in the following way to reduce data fetch time, keeping in mind the necessary security measures.

- All the user's login information is stored in MySQL as it is more secured.

- Various other information like data related to cars, flights and hotels along with booking, billing and analytics data is stored in MongoDB as it supports faster data retrieval and easy to store data in modular format with no relation.
- All the images are upload on the amazon server and only the links are stored in the database. This helps in avoiding unnecessary data dumping in the database which could have hampered retrieval time.
- We used Redis to implement caching, so we need not to perform actual DB operation if we already found same in key in our cache. It gives better performance and less resource utilization.
- We implemented connection pooling, so from beginning we keep fix number of connection open so when heavy load comes we need not to open and close many connections. We just take one connection from pool, use it and then return it back to pool for reuse. It helps us to perform DB operation faster.
- For implementation of kafka messaging queue, we created two topics: kayak and response\_topic. First one is for sending message from node server to kafka backend server and later one is for return flow to get response.

### 3. Policy of writing data into the Database

As we discussed above, that performing queries on the database requires costly resources, it is very essential to limit these kinds of activities. We followed a few policies which helped us keeping our databases optimized.

- No unnecessary data writing in the database. We only updated the database when a attribute is changed.
- All the images are upload on the amazon server and only the links are stored in the database. This helps in avoiding unnecessary data dumping in the database which could have hampered retrieval time.
- We used SQL caching which helped us retrieving data faster.
- We have created different api for different kind operations on each resource like GET, PUT, POST, DELETE. Because of this it helps us keep data integrity.
- We used MySQL for maintaining user login information as MySQL is more secured.
- Other data of the user as well as flights, hotels and cars with billing and booking information is stored in MongoDB because MongoDB is faster for data retrieval and security is not the actual need in this case.
- We also have many heavyweight images for user profile, hotel images, car images which we are directly saving on the cloud. This is done to avoid excessive data to be stored in the database which will also increase the hamper the performance.

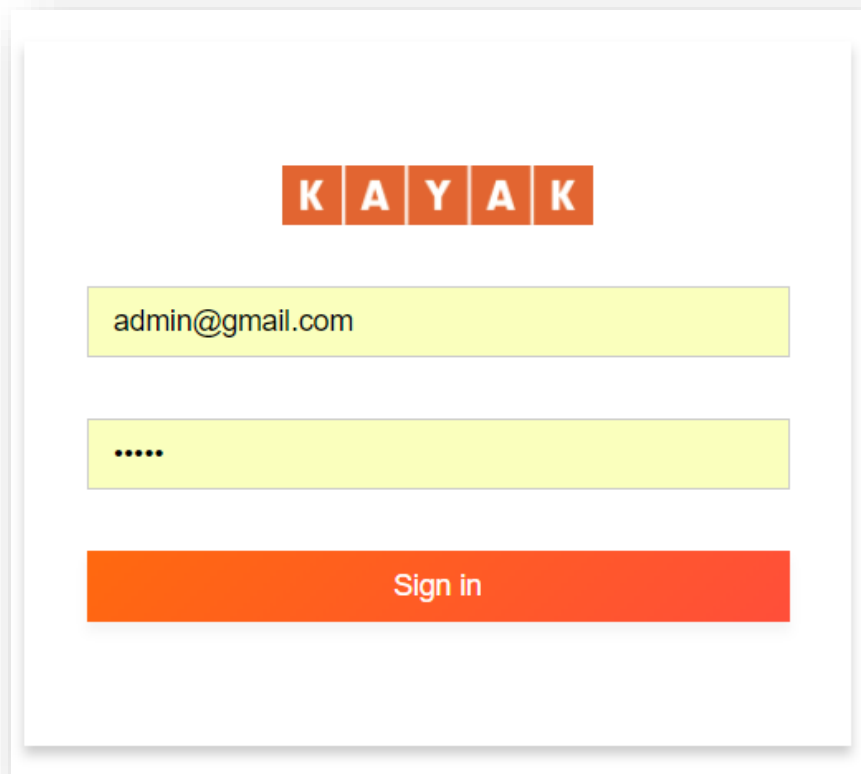


# ADMIN APPLICATION

## Admin Section

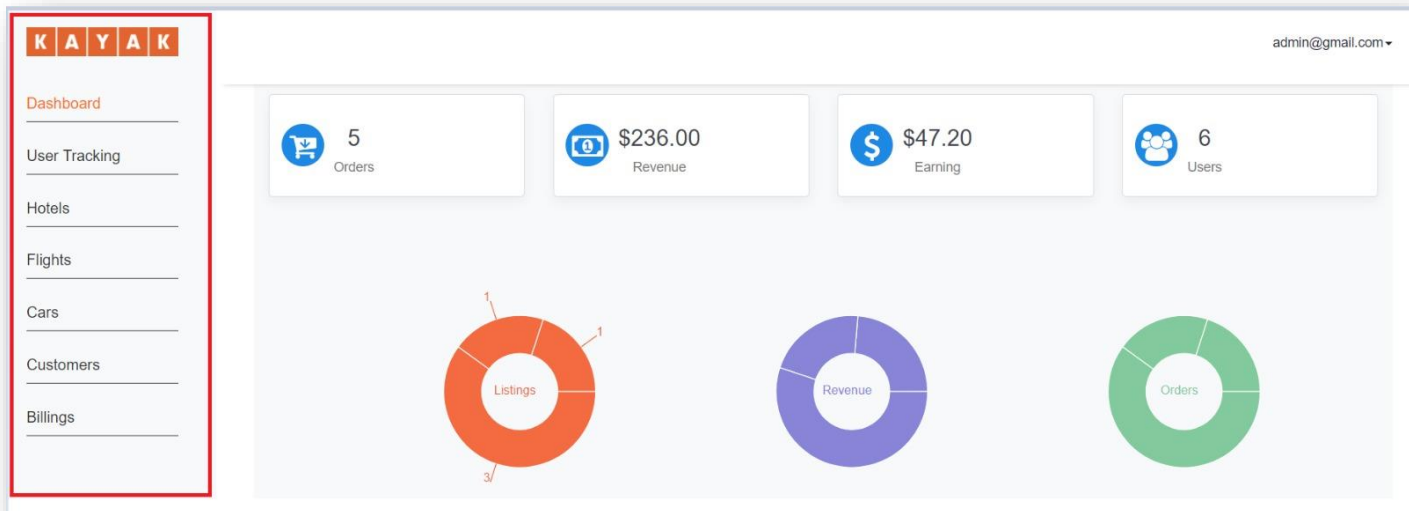
An admin application is made which can be only accessed by the admin credentials. Admin application has all the special privileges of adding, deleting, editing a flight, hotel or car. All the data added by the admin will be stored in MongoDB and will be displayed in the main customer application based on the search results.

A customer will not have any option to add, delete or modify the flights, hotels or cars data.

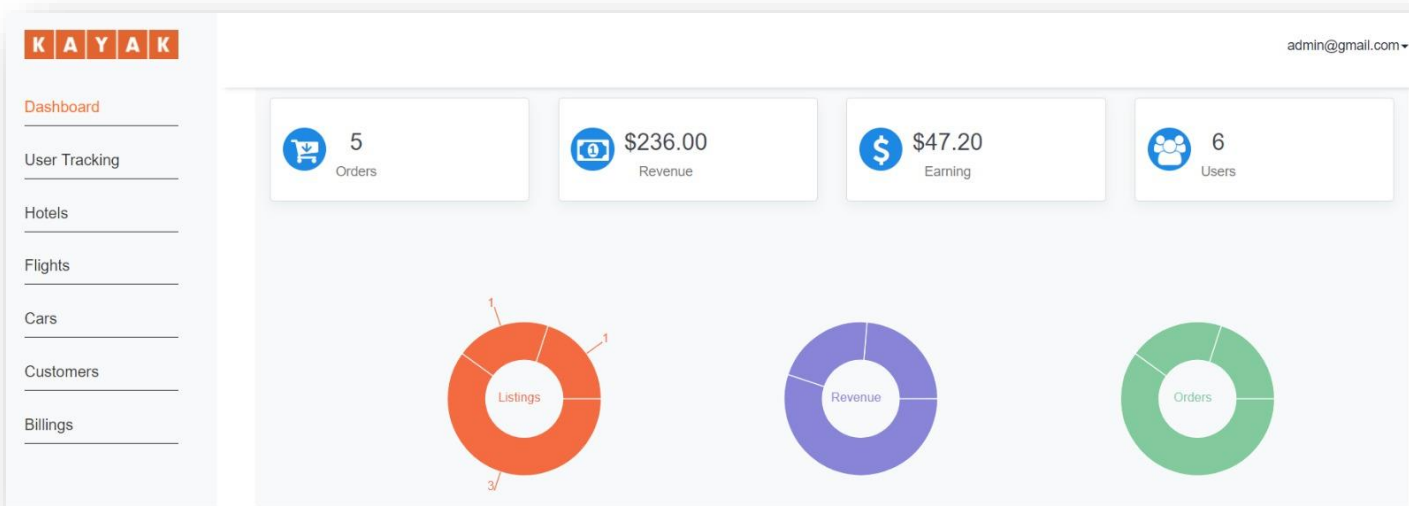


The image shows a sign-in form for the Kayak Admin application. At the top, the Kayak logo is displayed, consisting of five orange squares with the letters K, A, Y, A, and K in white. Below the logo, there are two yellow input fields. The first field contains the email address 'admin@gmail.com'. The second field contains five dots, indicating a password. Below these fields is a large orange button with the text 'Sign in' in white.

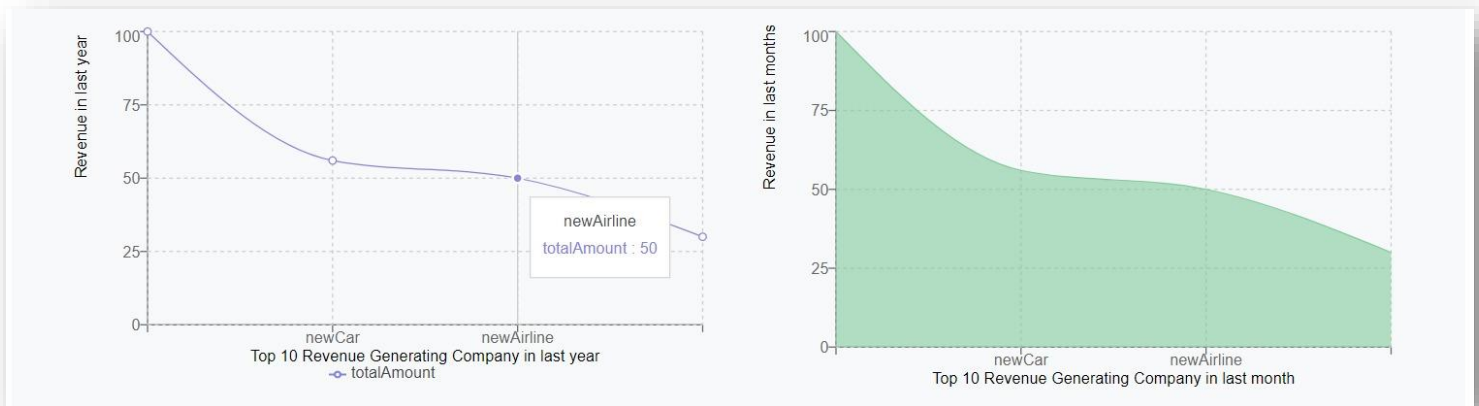
After login, a admin will get various options in the side bar (marked in red), Admin can use the links to go to different tabs and access data accordingly.



On the dashboard and User Tracking page, the admin can see various interactive charts and figures which can be used for data analysis. The charts can be used to analyze the total revenue, number of users registered, number of visitors, total earning, total orders etc. Using pie charts, admin can see share of each category in revenue, orders, and total listings.



Admin can see the highest revenue generating listings in last year and last month in two different graphs shown below. On hover of each point in line chart we can see value of revenue value and name of listing at that point. This data is generated from data of past year and past moth billings.



Admin also can see top revenue generating cities in bar graph below. On hover of each bar admin can see the revenue amount and number of orders from that city. On x-axis the name of cities is given. For example, we have taken San Jose and New York here.

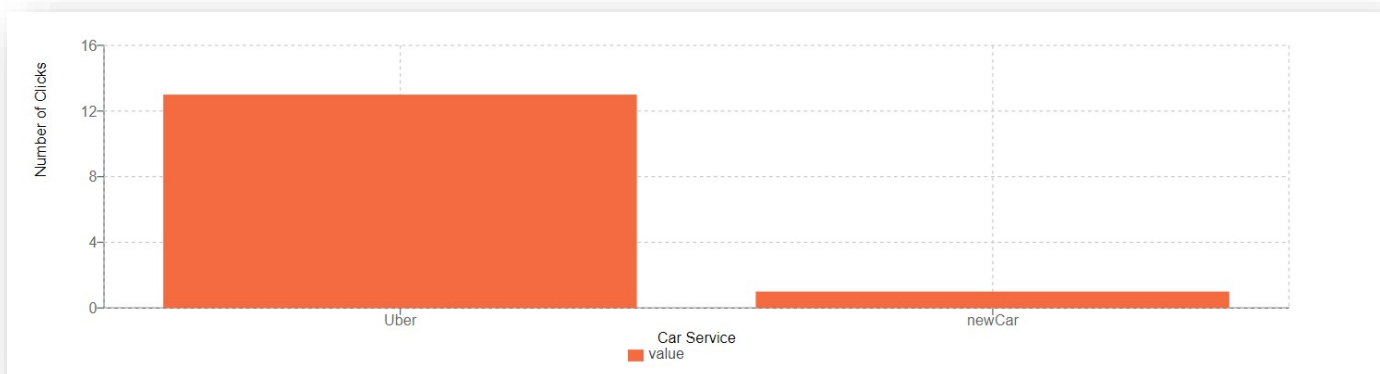


We have created click tracking services that fire on click of different html elements and log data in log file. A cron job is set up and performs operations on log file and fill data in analytics collection in DB from this file. Later we use this data to perform various analytical operations and generate attractive graphs.

Using this click count data we can show most popular hotel, flight and car services on our user tracking page. We also showed which pages are accessed most by user in form of bar graph like below.

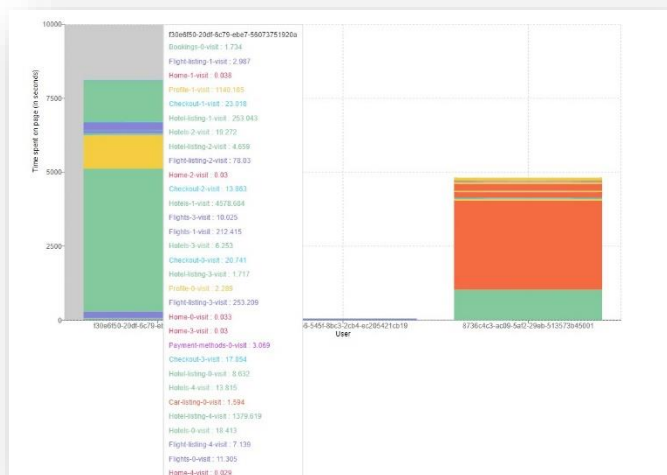


As we have click count data for each listing we are showing top car services which are popular among users. Same way we are showing top airlines and top hotels. On hover of each bar admin can see click counts for that listing. The name of listing are shown on x-axis.





For user tracking purpose we used bar graph to show amount of time user spent on each page in chronological order, starting from bottom of bar. On hover of bar whole track of user access is show like below screenshot. Id of user is shown on x-axis to maintain anonymity. If user accessed same page for multiple times by navigating from different pages, each visit is mark with a number and time spent in each visit, that is also visible in screenshot. This way we can track any user very easily.



All the pages have options to add data, below is an example to add a flight details.

The screenshot shows the Kayak admin interface with a modal form for adding flight details. The left sidebar contains navigation links: Home, Hotels, Flights (highlighted), Cars, Customers, Billings, Admin Console, and Host Console. The main content area displays a table of flight details with columns for Airline Name, Source, Destination, Departure, and Arrival. The modal form is open, allowing the user to input the following information:

- Flight Number: 241B
- Airline Name: Southwest
- Source: SJC
- Destination: LAX
- Departure: 12:11 PM
- Arrival: 12:12 PM
- Meals Included: ☒ Yes
- Luggage Count: 2
- Flight Service Start Date: (empty)

The modal form includes 'Close' and 'Submit' buttons at the bottom. The background table shows existing flight data with edit and delete icons.

Option to add Hotel

The screenshot shows the Kayak admin interface with a modal form for adding hotel details. The left sidebar contains navigation links: Dashboard, User Tracking, Hotels (highlighted), Flights, Cars, Customers, and Billings. The main content area displays a table of hotel details with columns for Hotel Name, Address, City, Zip, Phone Number, Email, Amenities, and Overall Rating. The modal form is open, allowing the user to input the following information:

- Hotel Name: (empty)
- Address: (empty)
- City: (empty)
- Zip: (empty)
- Phone Number: (empty)
- Email: (empty)
- Amenities: ☐ Pool, ☐ Bar, ☐ Jacuzzi, ☐ Lunch, ☐ Dinner, ☐ A/C, ☐ Parking, ☐ Wifi, ☐ TV
- Star: (empty)
- Overall Rating: (empty)

The modal form includes 'Close' and 'Submit' buttons at the bottom. The background table shows existing hotel data with edit and delete icons.

## Option to add Hotel

KAYAK

Dashboard

User Tracking

Hotels

Flights

Cars

Customers

Billings

Car Name

Honda

Merci

Car Type

Select

Car Name

Upload

Browse

Occupancy

No of Occupants

No Of Cars To Add

Service Start Date

mm/dd/yyyy

Service End Date

mm/dd/yyyy

Car City





Close

Submit





admin@gmail.com

Add Car



After adding data, the details are shown as below. Admin will also have an option to edit or delete any of the entries.

Add Flight				
Airline Name	Flight Number	Source & Departure	Destination & Arrival	
Jet	111A	SJC	LAX	 
		14:22	00:12	
Delta	22B	LAX	SJC	 
		00:12	11:22	



Hotel entries in admin section.

Add Hotel				
Hotel Name	Address	Contact Info	Rating	
Hilton	150 ABC Street, San Jose, CA, 95112	9891121122, hilton@hilton.com	4	 
Westin	200 Bay, San Jose, CA, 97667	6556656565, westin@westin.com	5	 

Car entries in admin section.

Add Car				
Car Name	Type	# of Cars/day	Rental Value	
Honda12	Standard	21	22	 

Registered User info in admin section. The admin will also have option to search users using the search box.

Search				
	Name	Address	Contact Info	
	Aman Ojha	0	aa@aa.comnull	



## Delete Options

Are you sure to delete Hilton ?

YES

NO

Are you sure to delete Honda ?

YES

NO

Billing section in admin. All the past billing information can be found in the billing section.

Bill Id	Booking Id	Listing Type	Total Amount
5a245b967a70d4820427d684	5a245b967a70d4820427d683	flights	1
5a245c077a70d4820427d687	5a245c077a70d4820427d686	cars	25

After clicking on a bill, Admin will further see details of a bill.

**Billing Information**

<b>Bill Id</b>	5a245b967a70d4820427d684
<b>Billing Date</b>	2017-12-03T20:16:22.274Z
<b>User Name</b>	Aman Ojha
<b>Total Amount</b>	1

**Booking Information**

<b>Booking Type</b>	flights
<b>Passengers's Email</b>	aa@aa.com
<b>Passengers's Contact Number</b>	1212121212
<b>Flight Source</b>	San Jose, California
<b>Flight Destination</b>	Los Angeles, California
<b>Flight Cabin</b>	Economy
<b>No of Passengers</b>	1
<b>Journey Date</b>	12-05-2017

Close

# CUSTOMER APPLICATION

## Client Section

The first page all the clients will be greeted with is shown below.

The screenshot shows the Kayak website's search interface for hotels. The background is a vibrant blue aerial view of a tropical island with several small, modern buildings. At the top, the Kayak logo is on the left, and navigation links for 'Hotels', 'Flights', and 'Cars' are in the center. A 'My Account' link with a user icon is on the right. Below the navigation, the text 'Search hundreds of travel sites at once' is centered. Underneath this, there are three tabs: 'HOTELS' (highlighted in orange), 'FLIGHTS', and 'CARS'. The main search form is a light gray box with four input fields: 'City' (empty), 'Check-in date' (12/03/2017), 'Check-out date' (12/03/2017), and 'No of guests' (1). To the right of these fields is an orange button with a white right-pointing arrow. Below the input fields, there is a 'Room type' section with four radio button options: 'Standard' (selected), 'Premium', 'Honeymoon', and 'Conference Room'.

All the users will be able to book hotels, flights or cars.

The screenshot shows the Kayak website's search interface for flights. The background is the same vibrant blue aerial view of a tropical island. At the top, the Kayak logo is on the left, and navigation links for 'Hotels', 'Flights', and 'Cars' are in the center. A 'My Account' link with a user icon is on the right. Below the navigation, the text 'Search hundreds of travel sites at once' is centered. Underneath this, there are three tabs: 'HOTELS', 'FLIGHTS' (highlighted in orange), and 'CARS'. The main search form is a light gray box with four input fields: 'Source' (empty), 'Destination' (empty), 'Date of travel' (12/03/2017), and 'No of travelers' (1). To the right of these fields is an orange button with a white right-pointing arrow. Below the input fields, there is a 'Cabin class' section with three radio button options: 'Economy' (selected), 'Business', and 'First'.

KAYAK Hotels Flights Cars My Account

Search hundreds of travel sites at once

HOTELS FLIGHTS CARS

City Pick-up date Drop-off date

12/03/2017 12/03/2017

Also, search suggestions will be provided to the users to help them select the desired city easily.

KAYAK Hotels Flights Cars My Account

Search hundreds of travel sites at once

HOTELS FLIGHTS CARS

Source Destination Date of travel No of travelers

San 12/03/2017 1

San Antonio, Texas

San Diego, California

San Jose, California

San Francisco, California



A user will also get an option to login or signup.

This screenshot shows the Kayak website's login interface. A modal window is centered on the screen, featuring two input fields for 'Email' and 'Password', followed by an orange 'Sign in' button. Below the modal, the main search form is visible, including fields for 'City', 'Check-in', and 'No of guests', along with radio buttons for 'Room type' (Standard, Premium, Honeymoon, Conference Room). The Kayak logo and navigation links (Hotels, Flights, Cars) are at the top.

KAYAK Hotels Flights Cars My Account

Sea nce

City Check-in 12/0

No of guests 1

Room type: ☒ Standard ☐ Premium ☐ Honeymoon ☐ Conference Room

Don't have an account? Sign up

This screenshot shows the Kayak website's signup interface. A modal window is centered on the screen, featuring four input fields for 'First Name', 'Last Name', 'Email', and 'Password', followed by an orange 'Sign up' button. Below the modal, the main search form is visible, including fields for 'City', 'Check-in', and 'No of guests', along with radio buttons for 'Room type' (Standard, Premium, Honeymoon, Conference Room). The Kayak logo and navigation links (Hotels, Flights, Cars) are at the top.

KAYAK Hotels Flights Cars My Account

Sea nce

City Check-in 12/0

No of guests 1

Room type: ☒ Standard ☐ Premium ☐ Honeymoon ☐ Conference Room

Already have an account? Sign in

Error on wrong login credentials

A login form with a white background and a dark blue border. It contains two input fields: the first has the text "sample@a.com" and the second has six dots. Below the fields is a red error message "Invalid email or password" and an orange "Sign in" button. At the bottom, there is a grey bar with the text "Don't have an account?" and a white "Sign up" button.

After selecting the desired cities and date of travel, the users will be shown the available options they have for booking.

The KAYAK website interface for flight searches. The top navigation bar includes the KAYAK logo and links for Hotels, Flights, and Cars. A user account link "My Account" is in the top right. The search section has four input fields: "Source" (San Jose, California), "Destination" (Los Angeles, California), "Date of travel" (12/03/2017), and "No of travelers" (1). Below these is a "Cabin class" section with radio buttons for Economy (selected), Business, and First. A "Filters" sidebar on the left includes a "Price Range" slider from \$0 to \$5000, "Meals" (YES/NO), and "Luggage Bag" (0, 1, 2, 3). The main results area displays two flight options. The first option is a Jet flight from San Jose to Los Angeles, departing at 23:21 and arriving at 00:12, with a 51m duration. It shows prices for First Class (\$1), Business (\$1), and Economy (\$1), with a "Book" button. The second option is a Delta flight from LAX to SJC, departing at 00:12 and arriving at 02:12, with a 2h duration. It also shows prices for First Class (\$1), Business (\$1), and Economy (\$1), with a "Book" button.

All the users will further have option to filter their search based on price range, or various other amenities available in the hotel, flight or car.

The screenshot shows the Kayak Cars search interface. At the top, there's a navigation bar with 'KAYAK' and tabs for 'Hotels', 'Flights', and 'Cars'. The 'Cars' tab is selected. Below the navigation bar, there are input fields for 'City' (San Jose, California), 'Pick-up date' (12/03/2017), and 'Drop-off date' (12/03/2017), followed by a search button. On the left, there's a 'Filters' sidebar with sections for 'Price Range' (a slider from \$0 to \$500), 'Luggage' (YES/NO checkboxes), 'Capacity' (checkboxes for 2, 4, 5, 7, 8), and 'Category' (checkboxes for Standard, Premium, Full Size, and Luxury). The main area displays two car rental options: 'Premium' (Honda or Similar) and 'Luxury' (Mercury or Similar). Each option shows a car image, a 'Special Rate' button, and a 'Book' button with a price of \$22. The 'Premium' option also shows 4 passengers, luggage YES, and 2 cars.

A user will also get an option to sort the available listing based on multiple parameters. Also, when the available listings are more than 20, we will get an option of 'Load More' to see more available options.

The screenshot shows the Kayak Flights search interface. At the top, there's a navigation bar with 'KAYAK' and tabs for 'Hotels', 'Flights', and 'Cars'. The 'Flights' tab is selected. Below the navigation bar, there are input fields for 'Source' (San Jose, California), 'Destination' (Los Angeles, California), 'Date of travel' (12/12/2017), and 'No of travelers' (1), followed by a search button. Below these fields, there's a 'Cabin class' section with radio buttons for 'Economy' (selected), 'Business', and 'First'. On the left, there's a 'Filters' sidebar with sections for 'Price Range' (a slider from \$0 to \$5000), 'Meals' (YES/NO checkboxes), and 'Luggage Bag'. The main area displays a flight listing for a 2A Jet from San Jose, California to Los Angeles, California, with a duration of 23:21. Below the flight listing, there's a table with columns for 'First Class', 'Business', and 'Economy', each with a price of \$1. A 'Sort by' dropdown menu is open, showing options: 'Price: Low to high', 'Price: Low to high', 'Price: High to low', and 'Departure time'. A 'Load more' button is visible at the bottom.

The user can select any of the available option and can further proceed for billing.

KAYAK

HotelsFlightsCars

My Account


Details

Honda or Similar | Premium | 4 Seats | Luggage YES

San Jose, California

12-03-2017

12-03-2017



Booking Information

Aman

Ojha

Mobile

aa@aa.com

License

Summary

Total Days	1
Vendor Base Fare	\$22
Total Base Fare	\$22
Tax	\$3
<b>Total</b>	<b>\$25</b>

Pay now

## Payment Options

Payment

☒ Select a payment method

No saved payment methods

☐ Or use a new one

Card number (XXXX XXXX XXXX XXXX)

Name on card

Expiry date (MM/YY)

CVV

☐ save this method for future use



After filling the form with all correct information and selecting the pay option, the users selected option is booked and billed.

KAYAKHotelsFlightsCars

My Account

Details

Honda or Similar | Premium | 4 Seats | Luggage YES

12-03-2017

Congratulations!

✔ Your booking was successful

My BookingsHome

Summary

Total Days1

Vendor Base Fare\$22

Total Base Fare\$22

Tax\$3

Total\$25

Pay now

Booking Information

Aman

Ojha

4212121212

aa@aa.com

1212

## Flight Listing page

Source

San Jose, California

Destination

Los Angeles, California

Date of travel

12/06/2017

No of travelers

1

→

Cabin class:

☒ Economy ☐ Business ☐ First

Filters

Price Range

\$0\$5000

Meals

☐ YES ☐ NO

Luggage Bag

☐ 0 ☐ 1 ☐ 2 ☐ 3

Sort by

Price: Low to high

2A  
Jet

23:21  
San Jose, California

00:12  
Los Angeles, California

51m

\$1

Book

First Class  
\$1

Business  
\$1

Economy  
\$1

222X  
Delta

00:12  
LAX

02:12  
SJC

2h

\$1

Book

First Class  
\$1

Business  
\$1

Economy  
\$1

## Flight Booking Page

KAYAKHotelsFlightsCarsMy Account

Details

Date of travel: 12-05-2017

2A  
Jet

23:21  
San Jose, California

00:12  
Los Angeles, California

51m

Cabin: ECONOMY

No of baggage: 0

Free meal included: No

Booking Information

Traveler #1

Sample

S

Sample

aa@aa.com

1212121221

Summary

Base fare\$1

No of travellers1

Subtotal\$1

Total\$1

Pay now

## Hotel Listing Page

KAYAKHotelsFlightsCarsMy Account

City

San Jose, California

Check-in date

12/06/2017

Check-out date

12/08/2017

No of guests

1

→

Room type:

☒ Standard

☐ Premium

☐ Honeymoon

☐ Conference Room

Filters

Stars

★★★★★


Rating

05

Amenities

Sort by

Price: Low to high



Hilton

San Jose

★★★★★

5.0 Excellent

Conference - \$200

\$200


Book

## Hotel Booking Page

KAYAKHotelsFlightsCarsMy Account

Details

Westin San Jose, California - 99999



Check-In

12-17-2017

Check-Out

12-17-2017

Room Type

Standard

Guests

1

Address

ppp  
San Jose, California  
99999

Contact

9879879877

Summary

Total Days

1

Vendor Base Fare

\$50

Total Base Fare

\$50

Total

\$50

Pay now

Booking Information

Aman

Mobile

Ojha

aa@aa.com

Payment

☒ Select a payment method

☐ 3413 - 4134 - 2413 - 4134


The user will also have option to check and update his profile.

KAYAKHotelsFlightsCarsMy Account

Profile

Payment methods

Past bookings



First name

Aman

Email

aa@aa.com

Street address

State

Last name

Ojha

Phone number

City

Zip code

Edit profile

A User will also get option to edit his profile

The screenshot shows the Kayak website's 'My Account' section. A modal form titled 'Edit profile' is open, allowing a user to update their personal information. The form includes fields for First Name, Last Name, Email, Address, City, State, Zip Code, and Phone Number. The 'Profile Image' field has a 'Browse' button. The background shows the 'My Account' page with links to Profile, Payment methods, and Past bookings, and a sidebar with a tree view of account settings.

**KAYAK** Hotels Flights Cars My Account

**Profile**  
Payment methods  
Past bookings

**Edit profile**

First Name: Aman  
Last Name: Ojha  
Email: aa@aa.com  
Address: 190 RYLAND ST, APT 1114  
City: SAN JOSE  
State: CA  
Zip Code: 998989  
Phone Number: 545445154  
Profile Image: Browse

Close Update

A user will also have option to add different payment methods.

The screenshot shows the Kayak website's 'My Account' section. A modal form titled 'Add card' is open, allowing a user to add a new payment method. The form includes fields for Card number, Name on card, Expiry date (MM/YY), and CVV. The background shows the 'My Account' page with links to Profile, Payment methods, and Past bookings, and a sidebar with a tree view of account settings.

**KAYAK** Hotels Flights Cars My Account

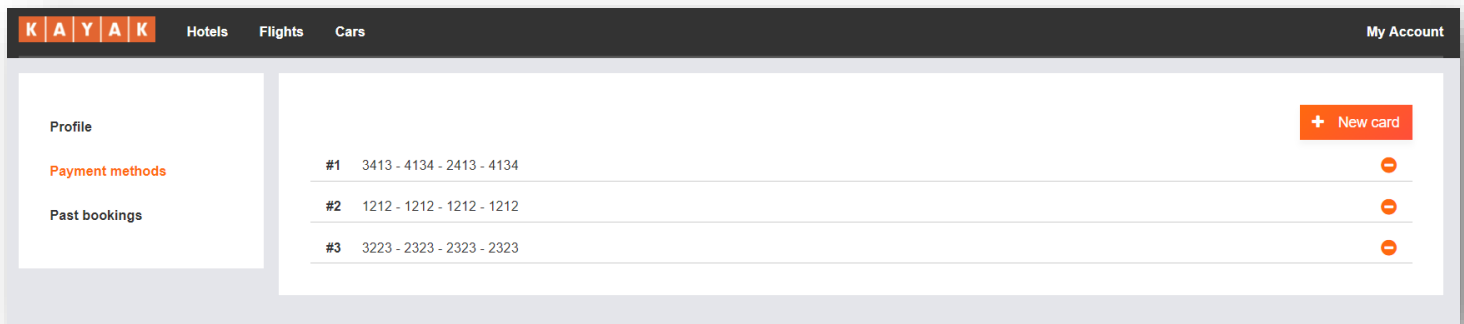
Profile  
**Payment methods**  
Past bookings

**Add card**

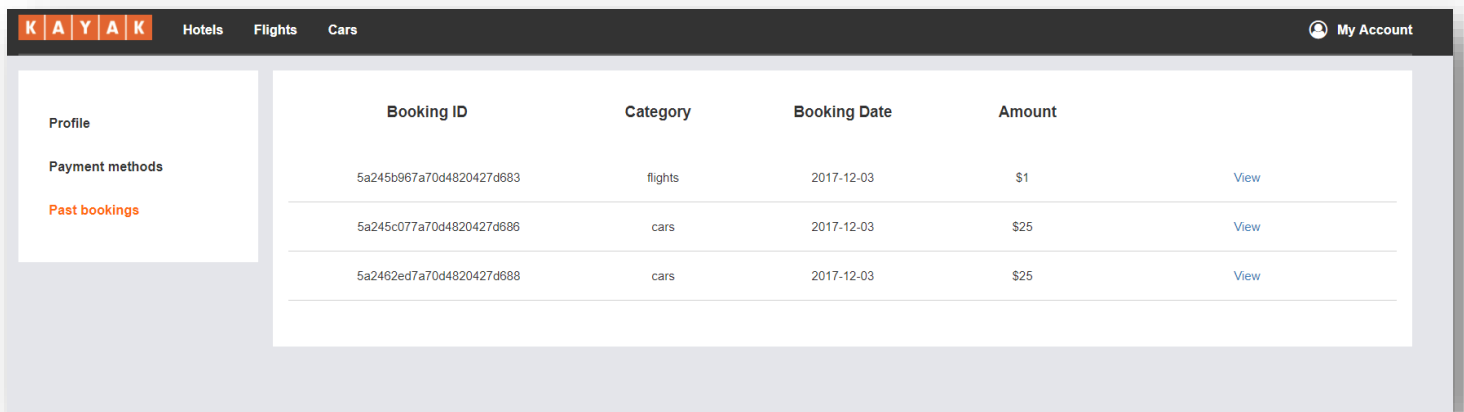
Card number (XXXX XXXX XXXX XXXX)  
Name on card  
Expiry date (MM/YY) CVV

Add card  
Cancel

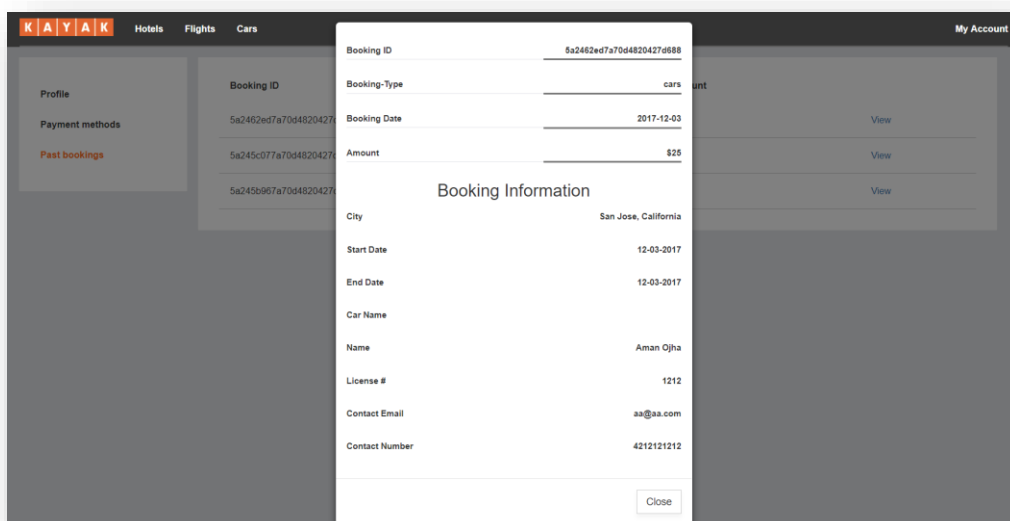
## Listing of all added payment options



All the previous booking history can be viewed on the Past Bookings tab.



## Car Past Booking receipts



Flight Past booking receipt

KAYAK

HotelsFlightsCars

My Account

Profile

Payment methods

Past bookings

Booking ID

5a2462ed7a70d48204276

5a245c077a70d48204276

5a245b967a70d48204276

Booking ID

5a245b967a70d4820427d683

Booking-Type

flights

Booking Date

2017-12-03

Amount

\$1

View

View

View

Booking Information

Source

San Jose, California

Destination

Los Angeles, California

Airline

Jet

Traveller #1

a a

Email

aa@aa.com

Contact

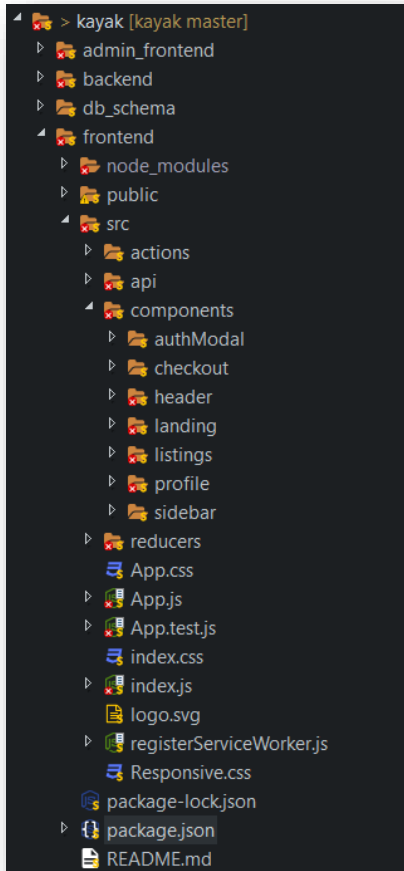
1212121212

Close

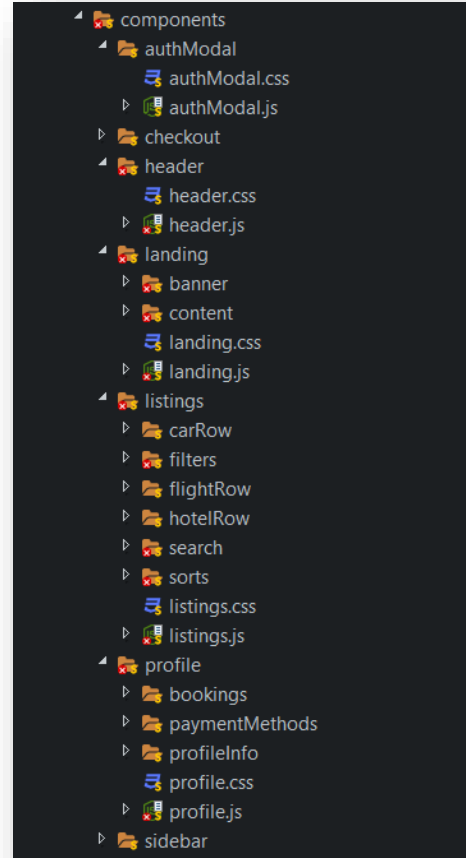
# CODE LISTING

## Client Frontend

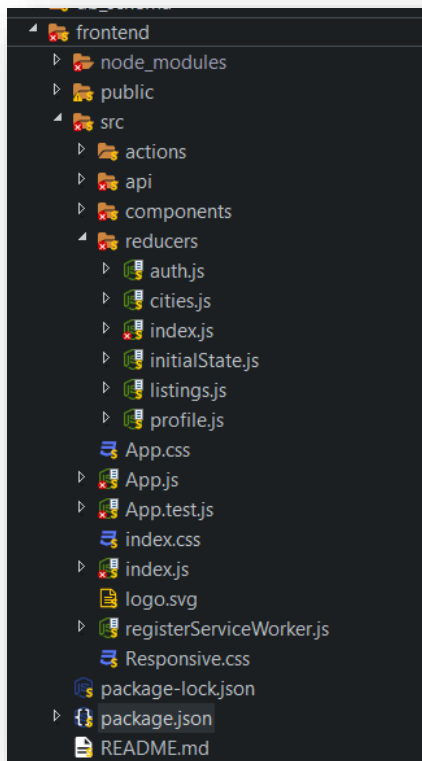
### Base directory structure



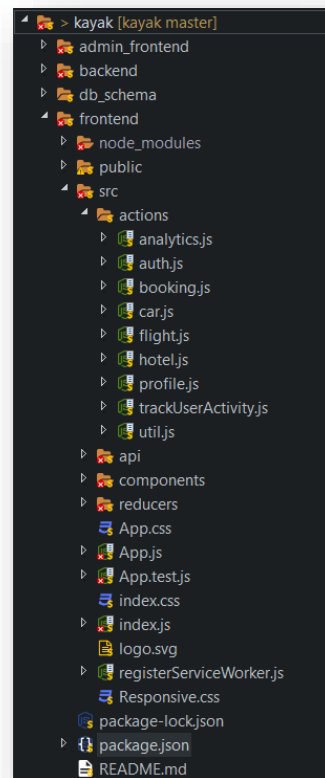
### Components in frontend



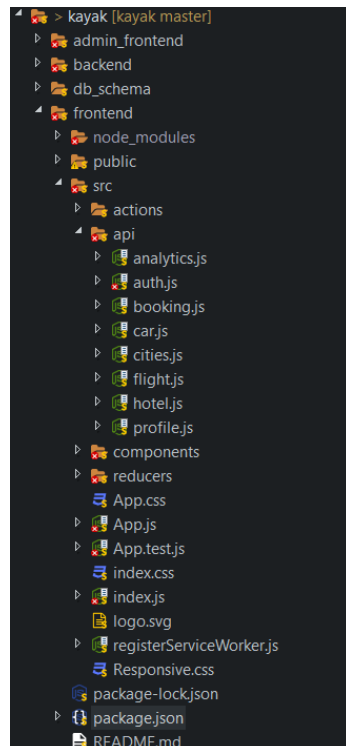
## Reducer



## Action Functions



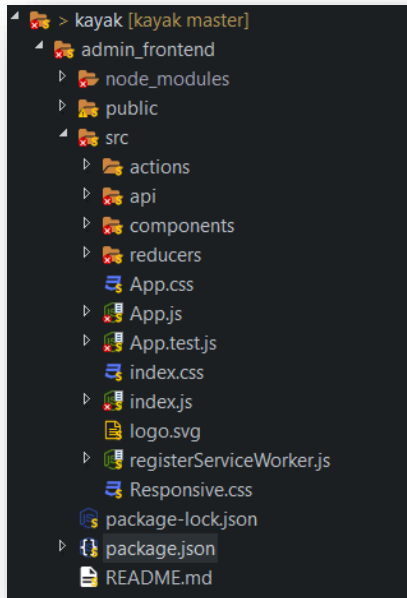
## APIs



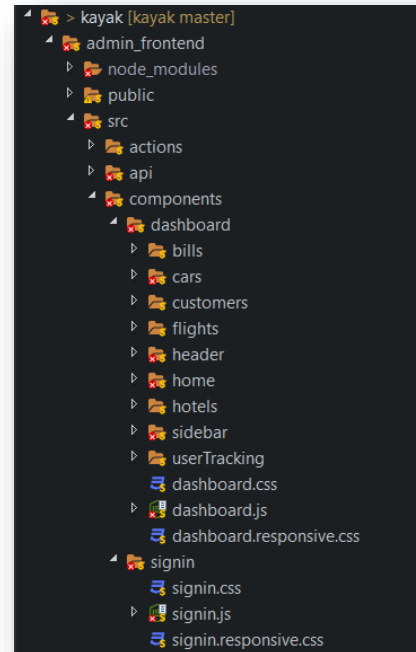


## Admin Frontend

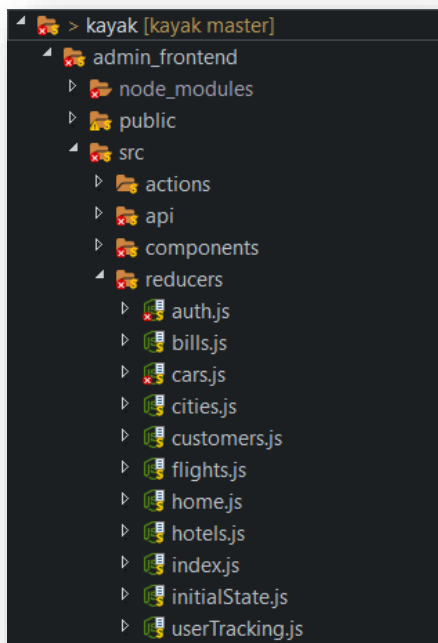
### Base directory structure



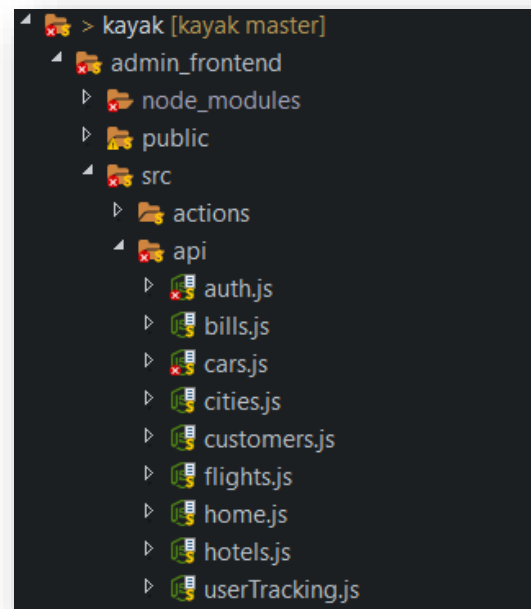
### Components in admin



### Reducer

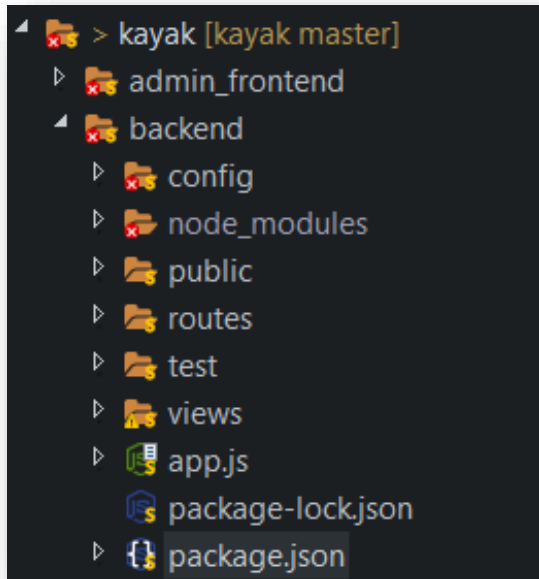


### APIs

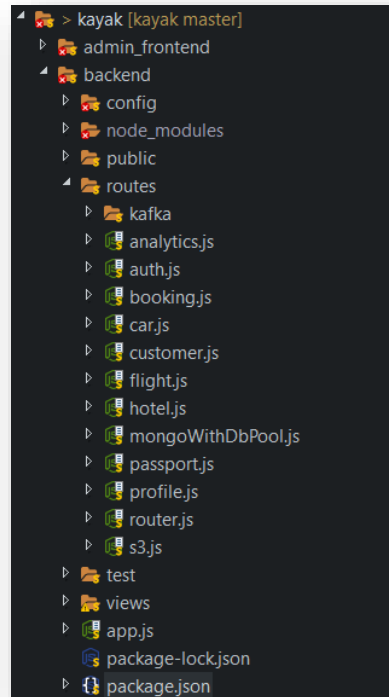


## Node Backend

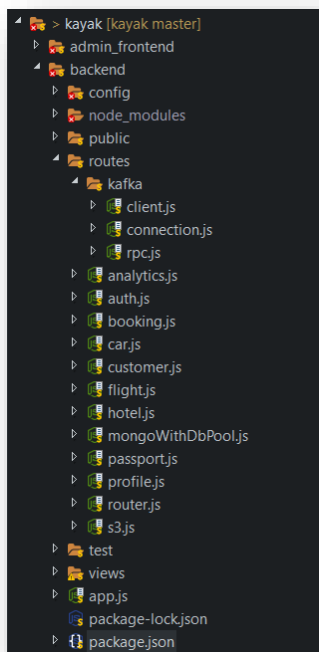
### Base directory structure



### Routes

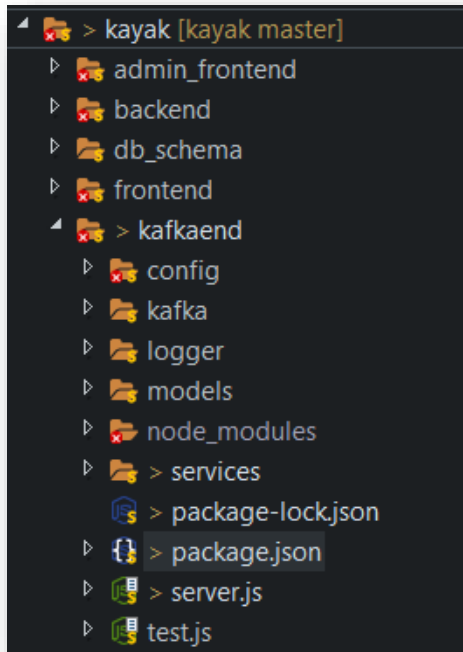


### Kafka routes

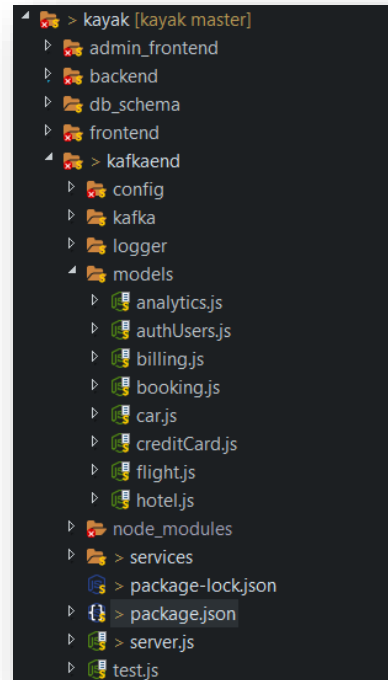


## Kafka Backend

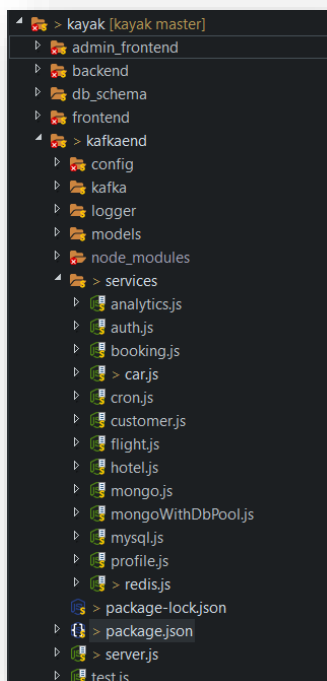
### Base directory structure



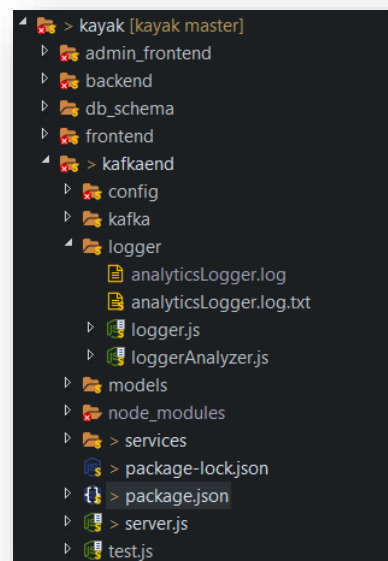
### Models



### Kafka services

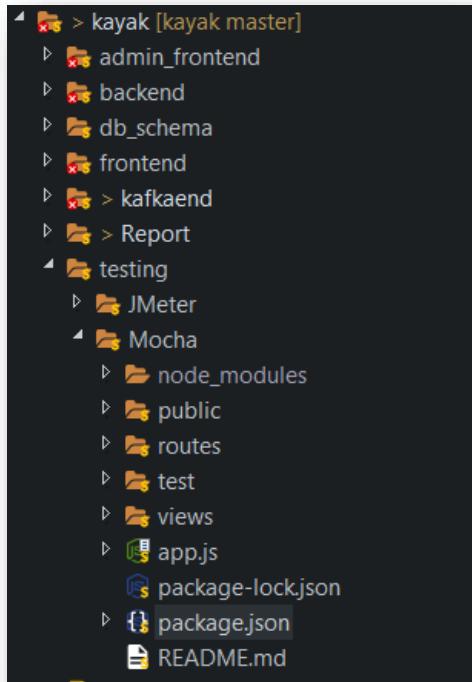


### Analytics logging

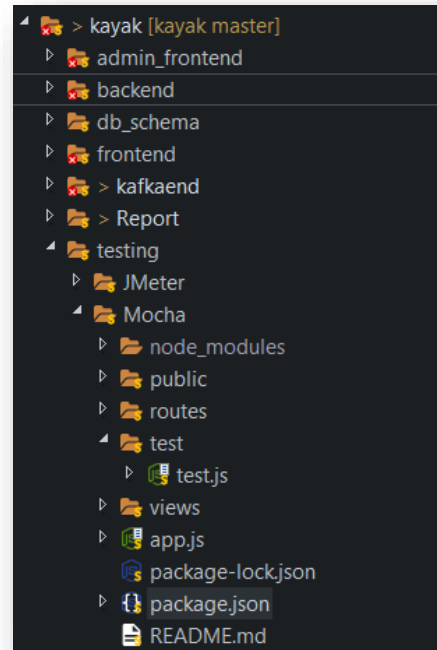


## Mocha Testing

### Base directory structure



### Test Cases

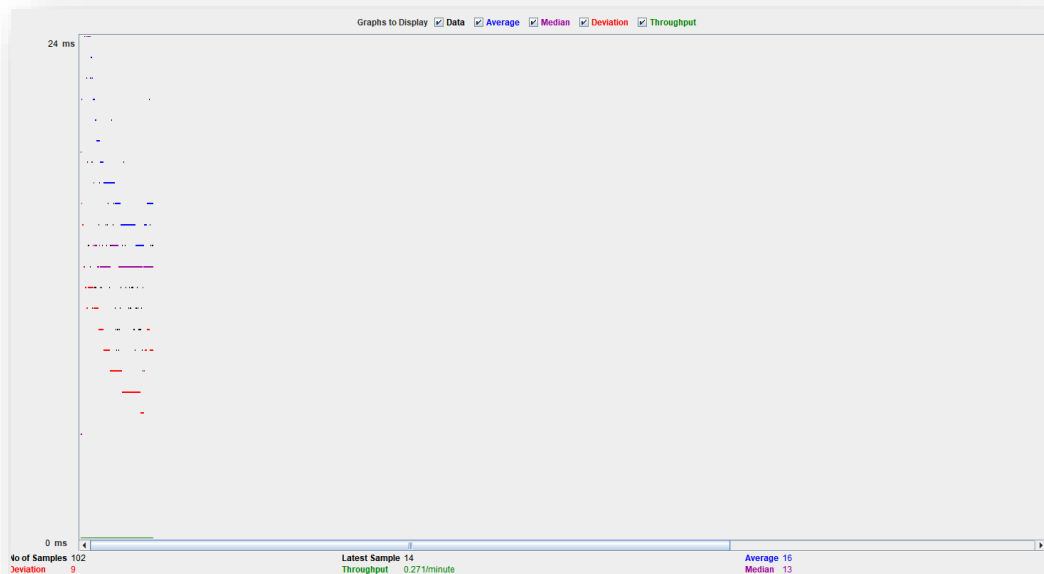


# JMETER TESTING

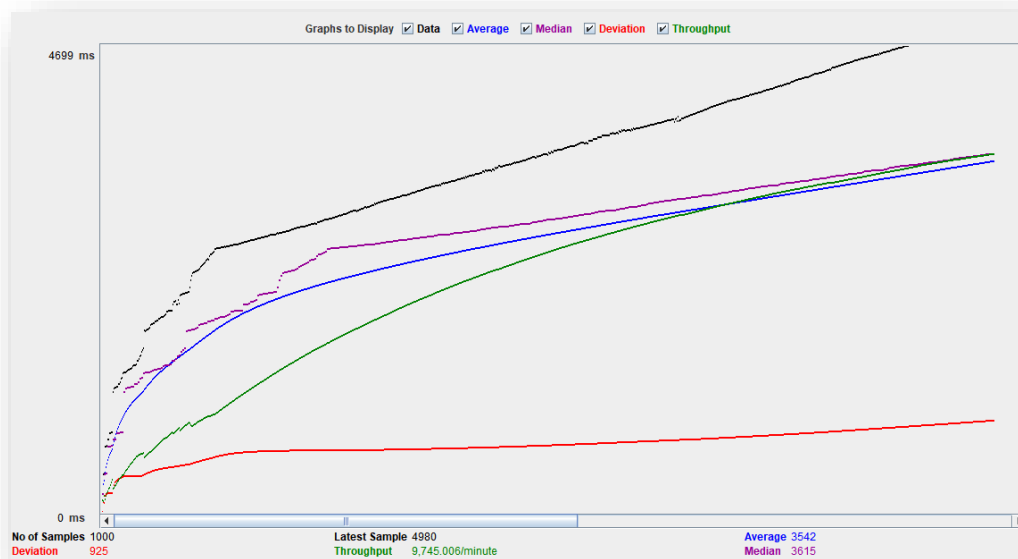
Apache JMeter is a testing utility which is commonly used as load testing tool for analyzing and measuring various aspects of a project. It is primarily focused on testing web applications. JMeter offers a variety of connection policies like HTTP, LDAP, JDBC etc.

## 1. Listing

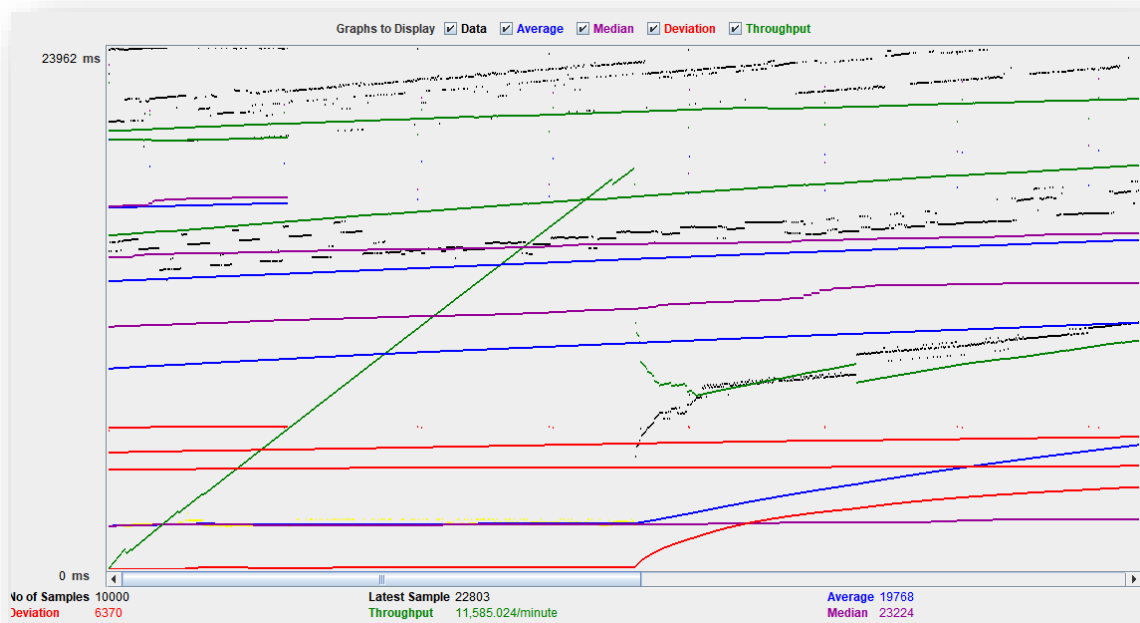
100 Users with average time of 16ms.



1000 User with average time of 3542 ms



10,000 User with average time of 19768 ms

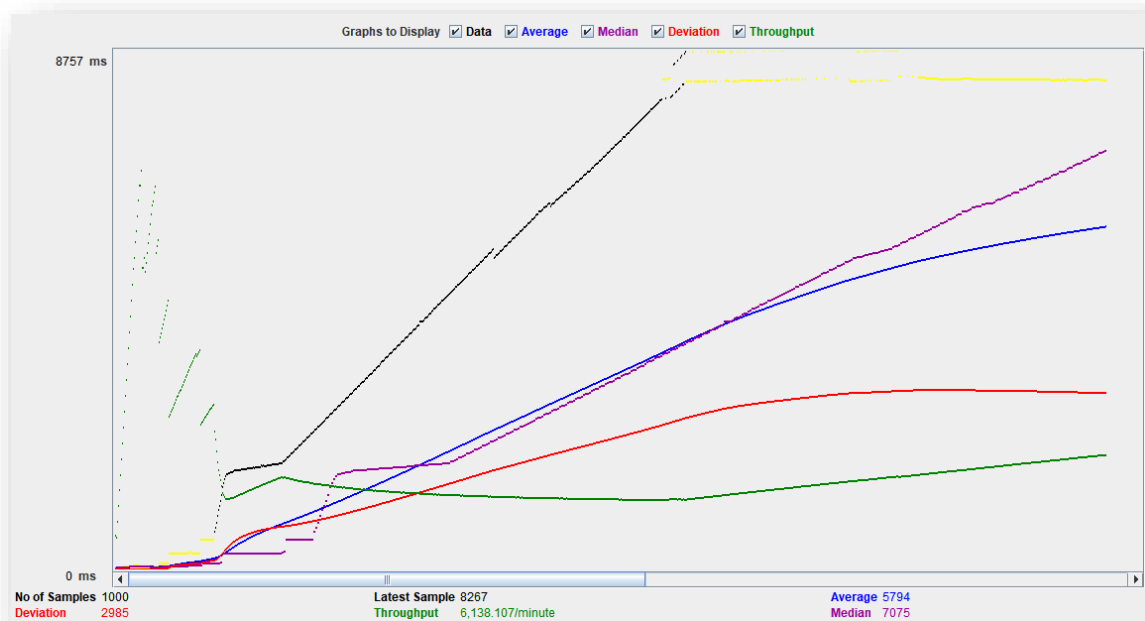


## 2. User Registration

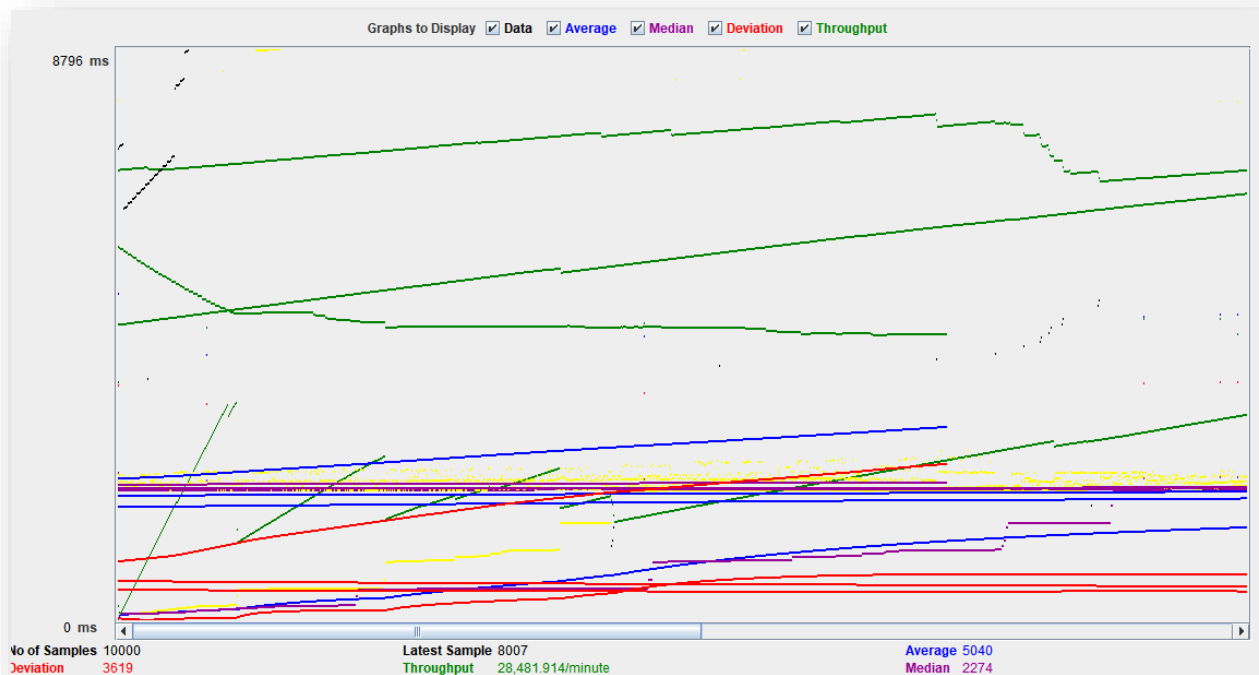
100 Users with average time of 518 ms



1000 Users with average time of 5794 ms

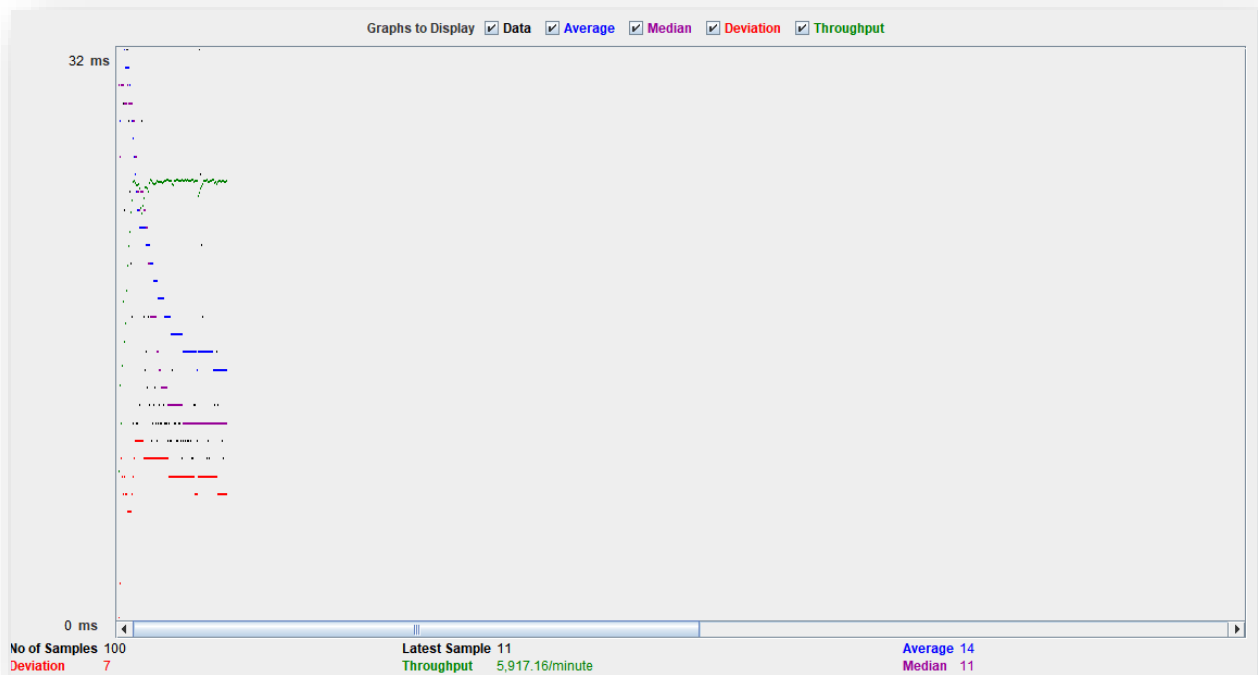


10,000 Users with average time of 5040 ms

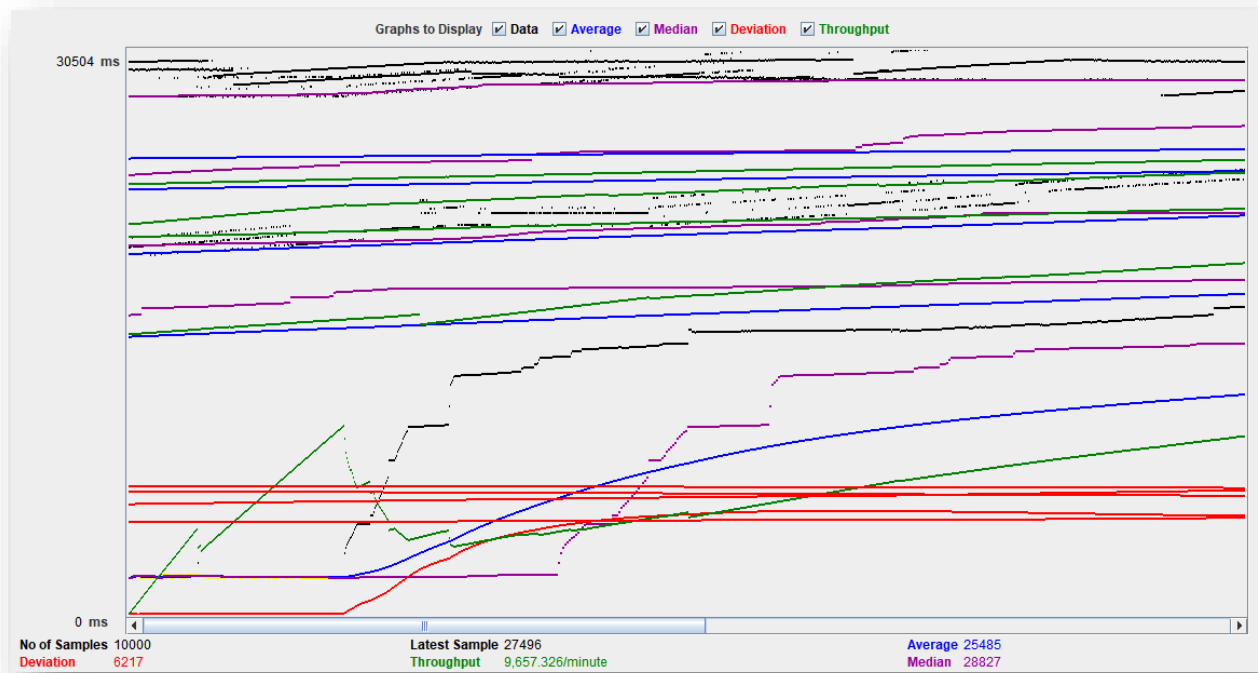


### 3. Bookings

100 Users with average time of 14 ms

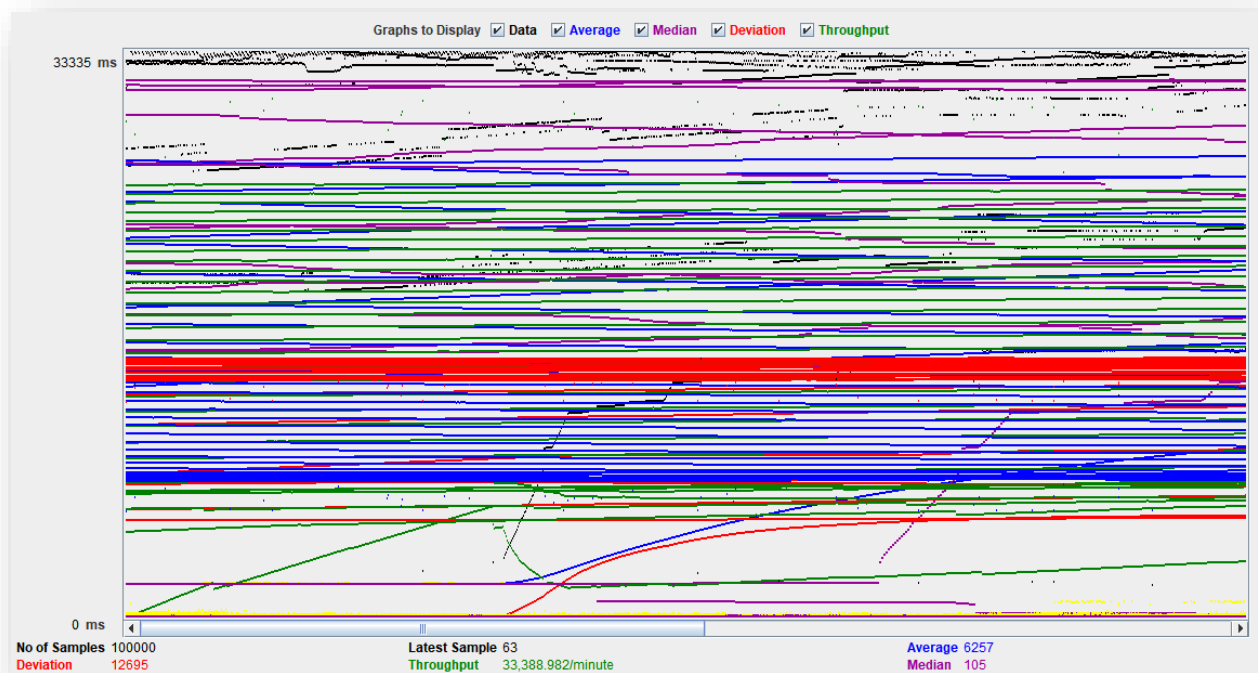


10,00 Users with average time of 25485 ms





100,000 Users with average time of 6257 ms



#### 4. JMeter conclusion

##### Connection Pooling:

A connection pool is a cache of database connections maintained so that the connections can be reused when future requests to the database are required. Connection pools are used to enhance the performance of executing commands on a database. In connection pooling, after a connection is created, it is placed in the pool and it is used again so that a new connection does not have to be established. If all the connections are being used, a new connection is made and is added to the pool. Connection pooling also cuts down on the amount of time a user must wait to establish a connection to the database.

We observed that with connection pooling in place, the average time is reduced. This helps in improving the performance of our application.

##### Performance:

Performance is very important for a system. We must check our systems performance to check our applications responsibility and stability under workload. It also helps us to analyze various attributes like scalability, reliability and resource usage.

In the graphs above, for each application, we can see the average time taken by the server api to serve the request. We performed various activity like 100 requests by a single user, 1000 requests, 10,000 requests by a user and so on. It replicates a real-life scenario where a web app has to serve multiple requests at once which affects the performance of a system.

# MOCHA TESTING

Mocha is a feature-rich JavaScript test framework running on Node.js and in the browser.

APIs were tested using Mocha successfully. Few API test cases are shown below:

```
1 var assert = require('assert');
2 var express = require('express');
3 var app = express();
4 var request = require('supertest');
5 var chai = require('chai').assert;
6
7 let chai = require('chai');
8 let chaiHttp = require('chai-http');
9
10 let should = chai.should();
11
12 chai.use(chaiHttp);
13
14 describe('http tests without auth', function(){
15
16     before(function() {
17
18     });
19
20     after(function() {
21
22     });
23
24     it('ADMIN should be able to login with correct details', function (done) {
25         chai.request('http://localhost:3002/api/v1/a')
26             .post('/signin')
27             .send({
28                 email:"admin@gmail.com",
29                 password:"ankit"
30             })
31             .end(function(err, res) {
32
```

```
38         it('ADMIN should not be able to login with wrong details', function (done) {
39             chai.request('http://localhost:3002/api/v1/a')
40                 .post('/signin')
41                 .send({
42                     email:"admin@gmail.com",
43                     password:"abc"
44                 })
45                 .end(function(err, res) {
46
47                     assert.equal(res.status, 400);
48                     done();
49                 });
50         });
51
52         it('CLIENT should be able to login with correct details', function (done) {
53             chai.request('http://localhost:3002/api/v1/c')
54                 .post('/signin')
55                 .send({
56                     email:"aa@aa.com",
57                     password:"abc@123"
58                 })
59                 .end(function(err, res) {
60
61                     assert.equal(res.status, 200);
62                     done();
63                 });
64         });
65
```

```

66     it('CLIENT should not be able to login with wrong details', function (done) {
67       chai.request('http://localhost:3002/api/v1/a')
68         .post('/signin')
69         .send({
70           email:"aa@aa.com",
71           password:"abc"
72         })
73       .end(function(err, res) {
74
75         assert.equal(res.status, 400);
76         done();
77       });
78     });
79
80     it('CLIENT should get error while signup with wrong data', function (done) {
81       chai.request('http://localhost:3002/api/v1/c')
82         .post('/signup')
83         .send({
84           email:"aa@aa.com",
85           password:"abc"
86         })
87       .end(function(err, res) {
88
89         assert.equal(res.status, 400);
90         done();
91       });
92     });
93

```

Output:

```

C:\SJSU\GitHub\kayak\testing\Mocha (Aman_5)
λ npm test

> kayak_server@0.0.0 test C:\SJSU\GitHub\kayak\testing\Mocha
> mocha

http tests without auth
  ✓ ADMIN should be able to login with correct details (169ms)
  ✓ ADMIN should not be able to login with wrong details
  ✓ CLIENT should be able to login with correct details (89ms)
  ✓ CLIENT should not be able to login with wrong details
  ✓ CLIENT should get error while signup with wrong data
  ✓ CLIENT should not get error while signup with correct data (88ms)
  ✓ CLIENT/ADMIN should logout

ADMIN http tests with auth
Cookie : connect.sid=s%3A68qdJr4LFKqF7j96YhhnBq5-sdtWh7-x.nKa1V78Gv16z5%2B3IcpxEihoQck5dzzw2d8PUFu0eQR4
  ✓ ADMIN LOGIN (99ms)
  ✓ ADMIN Passport check session
  ✓ ADMIN should get listing of all cars
  ✓ ADMIN should get listing of all flights
  ✓ ADMIN should get listing of all hotels
  ✓ ADMIN should get listing of all customers
  ✓ ADMIN should get revenue based on type
  ✓ ADMIN should get revenue based on Top Companies
  ✓ ADMIN should get revenue based on City
  ✓ ADMIN should get User Analytics data

CLIENT http tests with auth
Cookie : connect.sid=s%3A68qdJr4LFKqF7j96YhhnBq5-sdtWh7-x.nKa1V78Gv16z5%2B3IcpxEihoQck5dzzw2d8PUFu0eQR4
  ✓ CLIENT LOGIN (87ms)
  ✓ CLIENT should get All Credit card details
  ✓ CLIENT should get Hotel listings

20 passing (750ms)

```

# DATABASE SCHEMA

Hotel Schema
id : String
hotelName : String
hotelAddress : String
hotelCity : String
hotelState : String
hotelZip : Number
hotelPhoneNumber : Number
hotelEmail : String
hotelStar : Number
hotelRating : Number
hotelAmenities : [String],
hotelRooms : [{ roomType : String, priceTotal : Number, totalAvailable : Number, personPerRoom : Number}]
is_deleted : Boolean
serviceStartDate : Date
serviceEndDate : Date
availability : [{ availableDate : Date, hotelRooms : [{ roomType : String, priceTotal : Number, totalAvailable : Number, personPerRoom : Number}] } ]
images : [String]

AuthUser Schema
id : String
is_deleted: Boolean
auth_user_id : String
firstName : String
lastName : String
address: String
city: String
state: String
zip_code: String
phone_number: Number
profile_image: String
role: String
email: String

Car Schema
id : String
carType : String
carName : String
carQuantity : String
serviceStartDate : Date
serviceEndDate : Date
occupancy : Number
luggage : String
dailyRentalValue : Number
createdDate: Date
updatedDate : Date
availability : [{ availableDate : Date, availableCars : Number }]
is_deleted : Boolean
images : [String]

Flight Schema
id : String
flightNumber : String
airline : String
source : String
destination : String
arrival : String
departure : String
createdDate: Date
updatedDate : Date
is_deleted : Boolean
serviceStartDate : Date
availability : [{availableDate : Date , sections : [{class : String , price : Number , available : Number}] }]
serviceEndDate : Date
firstClassPrice : Number
firstClassSeats : Number
economyClassPrice : Number
economyClassSeats : Number
businessClassPrice : Number
businessClassSeats : Number
meals : Boolean
luggage : Number

Billing Schema
id : String
listingType: String
listingId:mongoose.Schema.ObjectId
bookingId:mongoose.Schema.ObjectId
userId:String
totalAmount:String
createdDate:Date
creditCardId:mongoose.Schema.ObjectId

CreditCard Schema
id : String
userId: String
cardNumber:String
nameOnCard:String
cvv:String
expiryDate:String

Booking Schema
id : String
listingType: String
listingId:mongoose.Schema.ObjectId
userId:String
bookingInfo:Object
createdDate:Date

## 1. MongoDB:

MongoDB is a document-oriented database. Instead of storing your data in tables made from individual rows, like a relational database does, it stores your data in collections made of individual documents. In MongoDB, a document is a big JSON blob with no particular format or schema.

MongoDB should be used in the following cases:

- High Write Load  
MongoDB by default prefers high insert rate over transaction safety. If you need to load tons of data lines with a low business value for each one, MongoDB should fit.
- High Availability in an Unreliable Environment
- Data is Location Based  
MongoDB has built in special functions, so finding relevant data from specific locations is fast and accurate.

In our application, MongoDB can be used to store files and user activity log data on the database.

## 2. MySQL:

MySQL is a free, open-source database management system. A DBMS is a system that manages databases and connects them to software. MySQL is a powerful, free open-source database management system that has been around for years. It is very stable and has a big community that helps maintain, debug and upgrade it.

MySQL should be used in the following cases:

- When we need to define relations between data present in different tables.
- Data to be stored is not large.
- Large Scalability is not required.

# OBSERVATIONS & LESSONS LEARNED

## Observations

The implementation of the whole Kayak website along with the analytics part was a large scale project which needed proper planning and coordination. Few of the points which we observed are as follows:

- Defining all the APIs before starting the project helped us in understanding the whole skeleton of the project.
- We divided the project into different modules internally which helped us to develop different sections parallelly. This resulted in a lot of time saving because one module did not affect the others.
- While developing the UI, we made various components like sidebars etc which was reused in various web pages. This reduced the number of lines of code we had to write.
- The analytics page helped us gaining data from end user, which can be later used to improve the services.
- We used Mongoose ODM for managing data in MongoDB by pre-defining the data models. It helped in reducing the code to perform CRUD operations.
- We followed Agile methodology in our project development phase. The daily standup meeting benefited us with keeping track of everyone's tasks and planning ahead for the remaining tasks.

## Lessons Learned

Below are few of the points we think could have been improved.

- Develop code keeping the analytics page in mind. This would have helped in developing the analytics page faster.
- JMeter helped us in testing of scalability and load balancing. After testing, we improved our code to make our website more stable.
- We used mocha to test various APIs with random data. It helped us in killing a few bugs as well.
- Feedback is an important part of project development. While deciding solutions to various functionalities, we kept a brainstorming session where everyone came up with suggestions and inputs.