CMPE 273

ENTERPRISE DISTRIBUTIVE SYSTEMS

TEAM PROJECT

KAYAK PROTOTYPE

TEAM 11

AMAN OJHA

ANKIT BHARADIYA

MAULIK BHATT

PALASH HEDAU

PRATEEK SHARMA

Contents

[ABOUT 2](#_Toc500038198)

[KAYAK 2](#_Toc500038199)

[CONTRIBUTION 3](#_Toc500038200)

[ABOUT THE PROJECT 4](#_Toc500038201)

[1. Object Management Policy 4](#_Toc500038202)

[2. Resource Handling 5](#_Toc500038203)

[3. Policy of writing data into the Database 5](#_Toc500038204)

[ADMIN APPLICATION 7](#_Toc500038205)

[Admin Section 7](#_Toc500038206)

[CUSTOMER APPLICATION 10](#_Toc500038207)

[JMETER TESTING 11](#_Toc500038208)

[1. Listing 11](#_Toc500038209)

[2. User Registration 13](#_Toc500038210)

[3. Bookings 14](#_Toc500038211)

[MOCHA TESTING 16](#_Toc500038212)

[DATABASE SCHEMA 17](#_Toc500038213)

[OBSERVATIONS & LESSONS LEARNED 18](#_Toc500038214)

[Observations 18](#_Toc500038215)

[Lesson Learned 18](#_Toc500038216)

# 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 Logo 2017.png

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 an meta search engine for finding flight, cars and hotel bookings.

# 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.

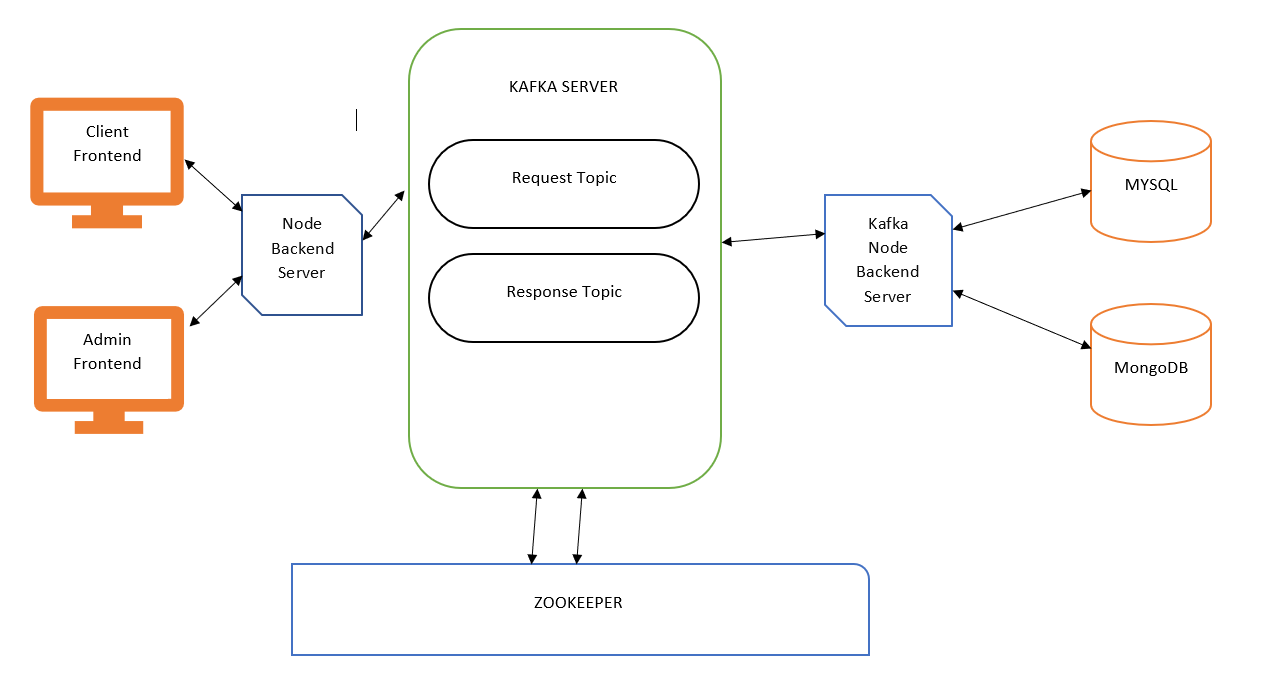
1. ANKIT BHARADIYA
2. MAULIK BHATT
3. PALASH HEDAU
4. PRATEEK SHARMA

# ABOUT THE PROJECT

## 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. The admin will also have access to an analytics page which will provide various real-time data of how the website is being used. It will provide information about revenues generated, popular flights, hotels and cars etc.

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. The results will be displayed according to the user’s selection criteria. After selected a particular option, the user will be redirected to the booking page where he has to fill in all the information required. Once all the information is validated, billing will be done.

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 particular 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.

## 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 users 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.

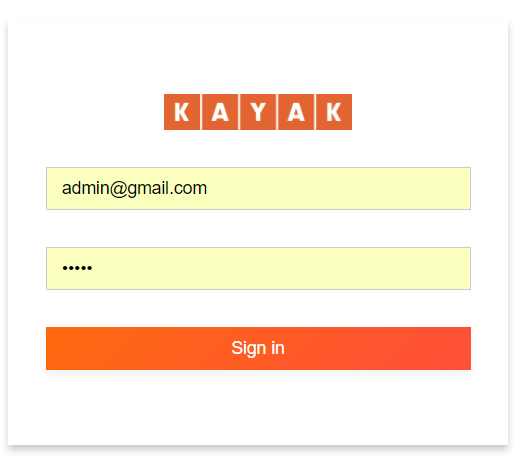
## Policy of writing data into the Database

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.

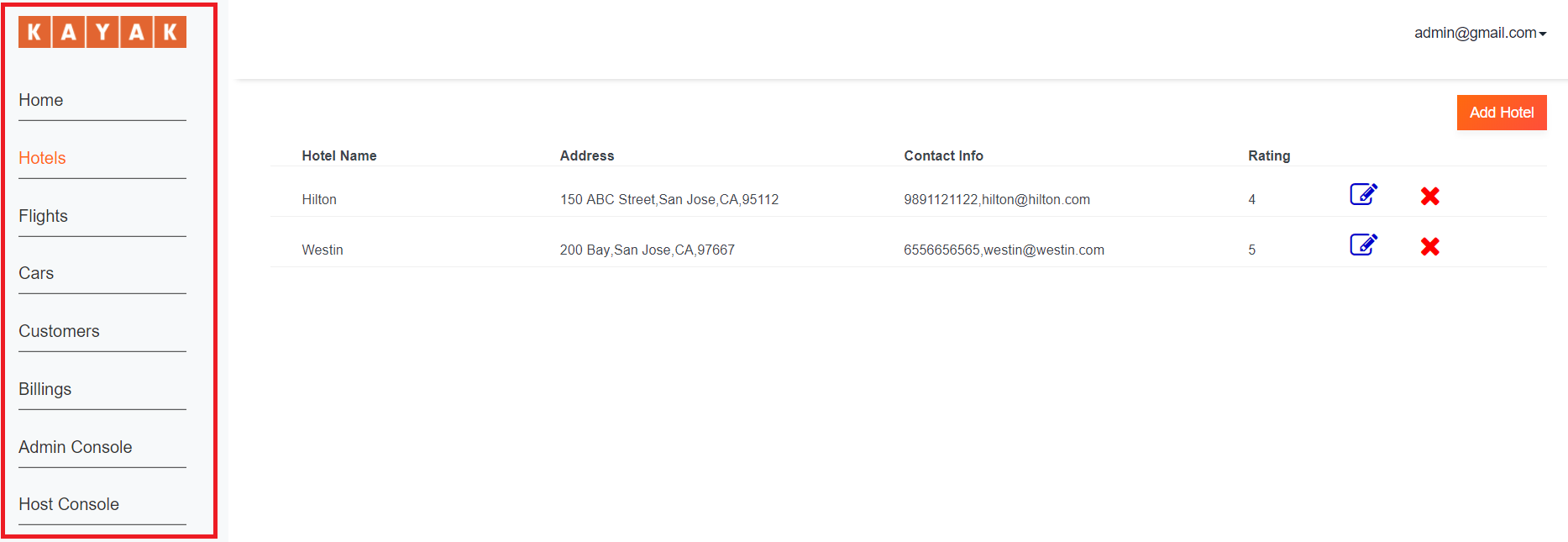
# ADMIN APPLICATION

## Admin Section

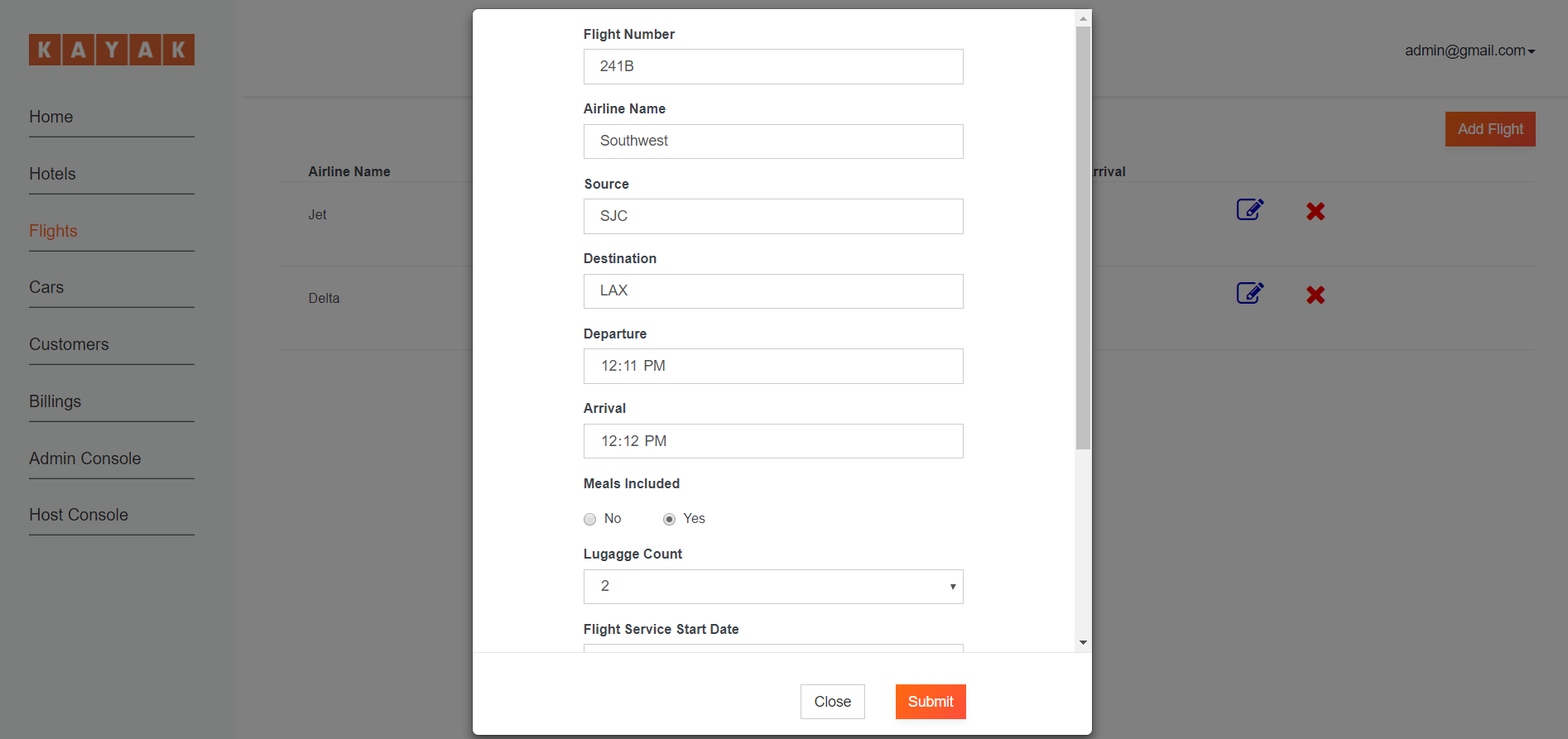


An admin application is made which can be only accessed by the admin credentials. Admin application has all the special privileges of adding, deleting, editing a flight, hotel or car. All the data added by the admin will be stored in MongoDB and will be displayed in the main customer application based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

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



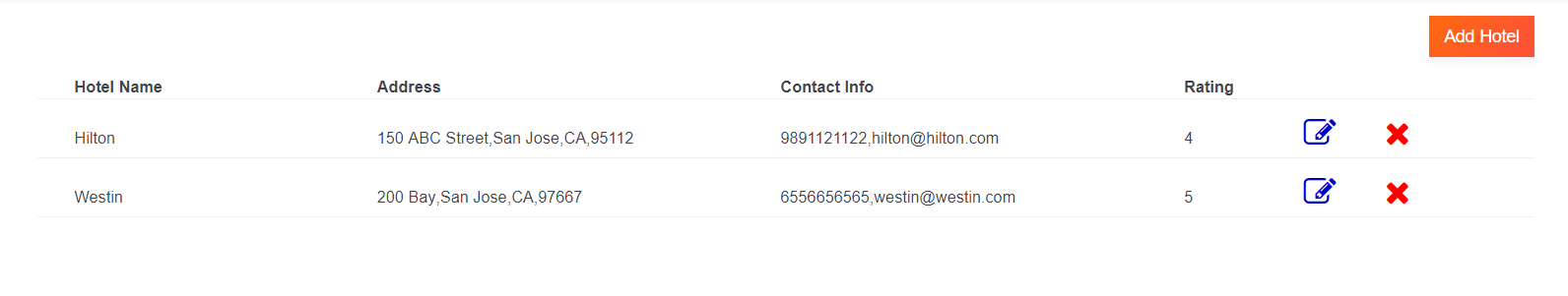
After login, a admin will get various options in the side bar (marked in red), Admin can use the links to go to different tabs and access data accordingly.



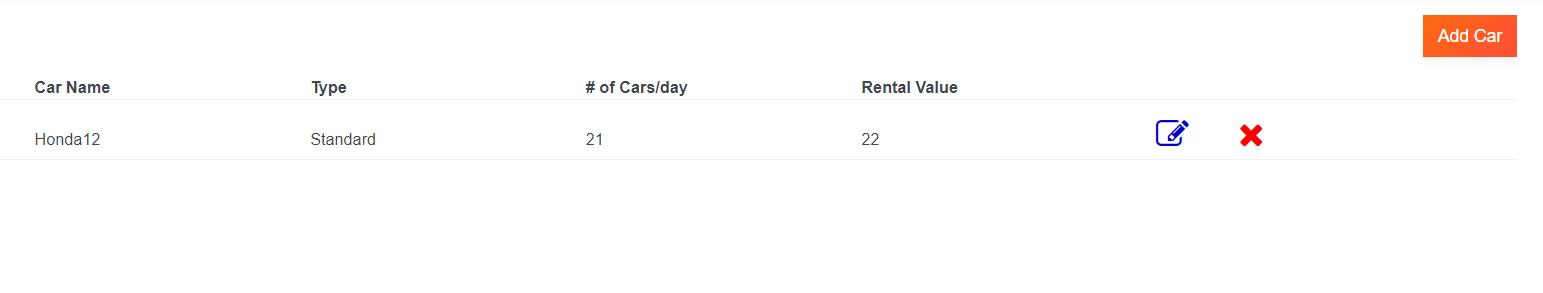
All the pages have options to add data, above is an example to add a flight details.



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



Hotel entries in admin section.



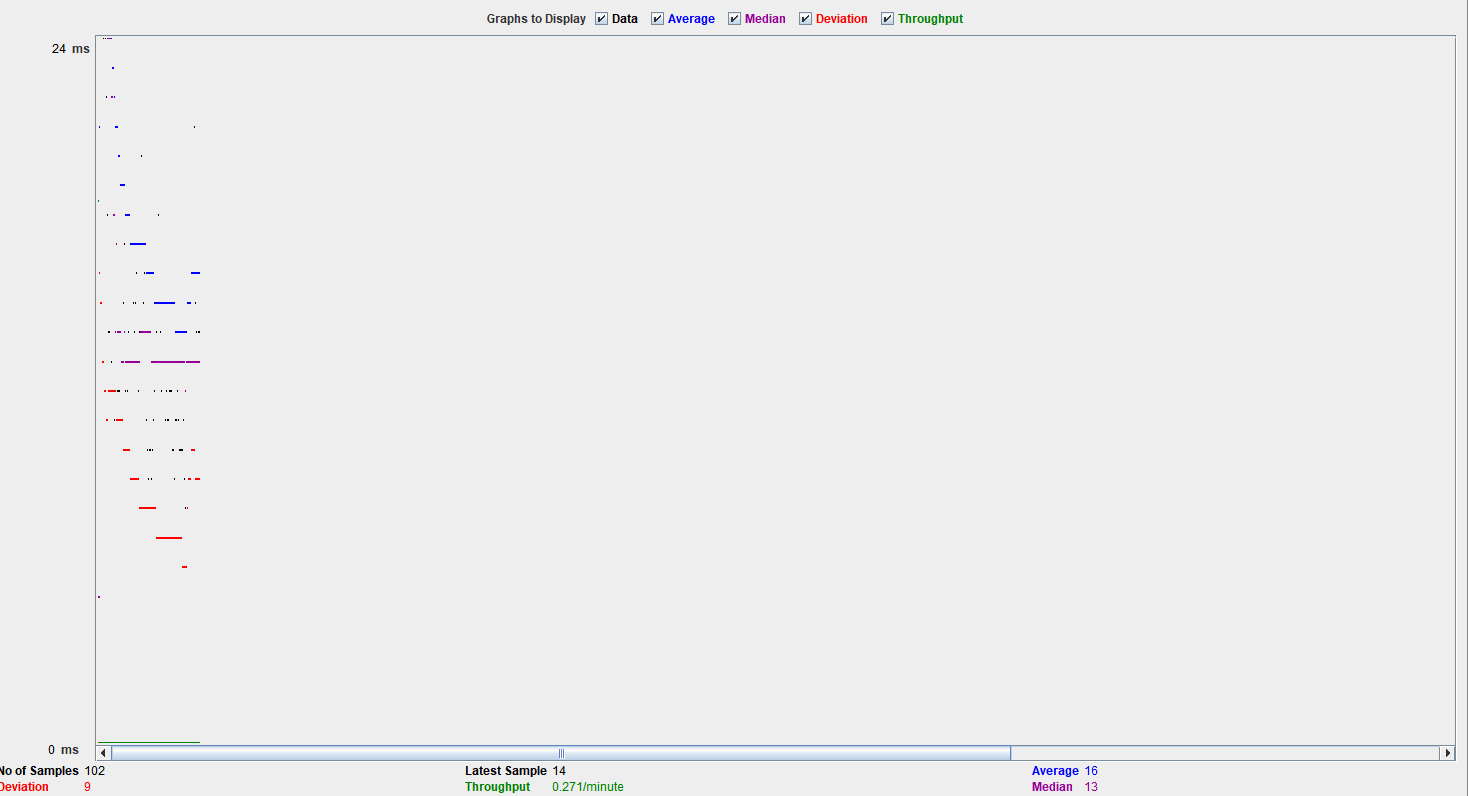
Car entries in admin section.

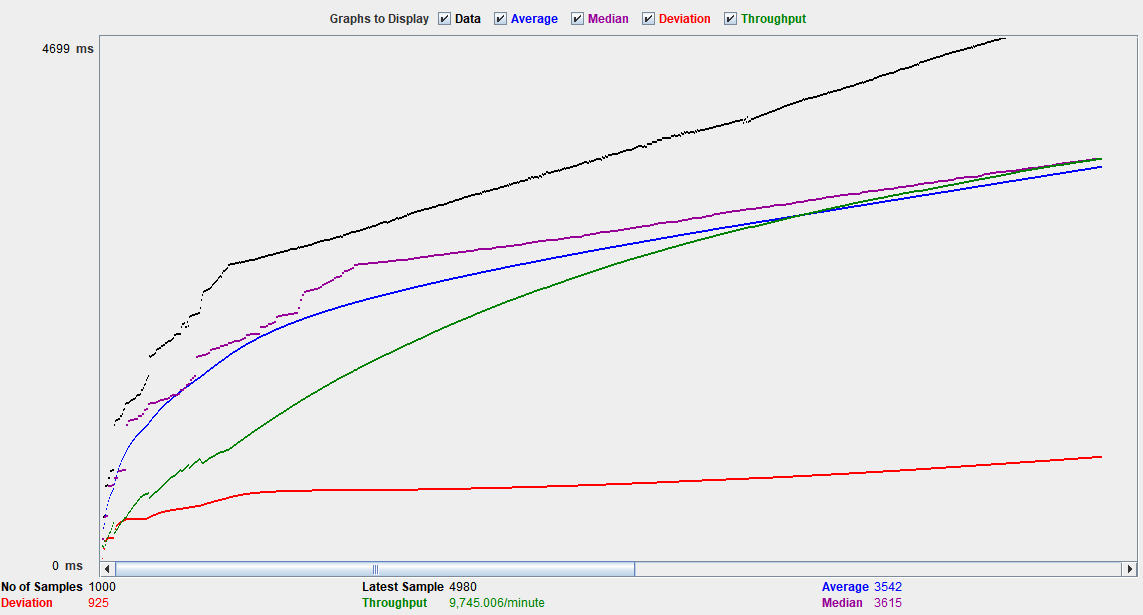
# CUSTOMER APPLICATION

# JMETER TESTING

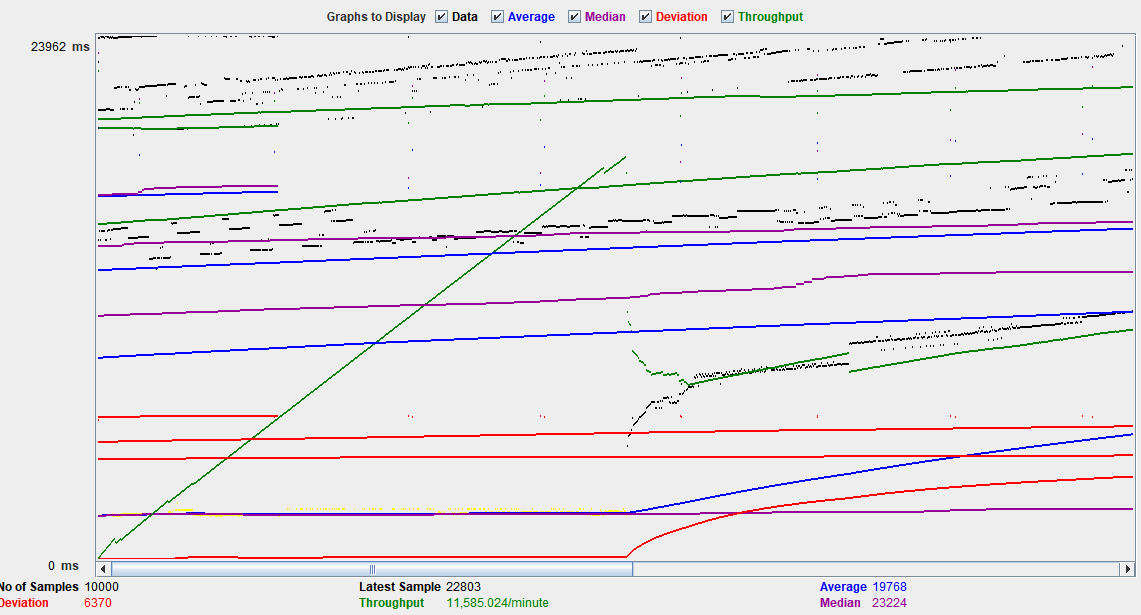
## Listing

100 Users with average time of 16ms.



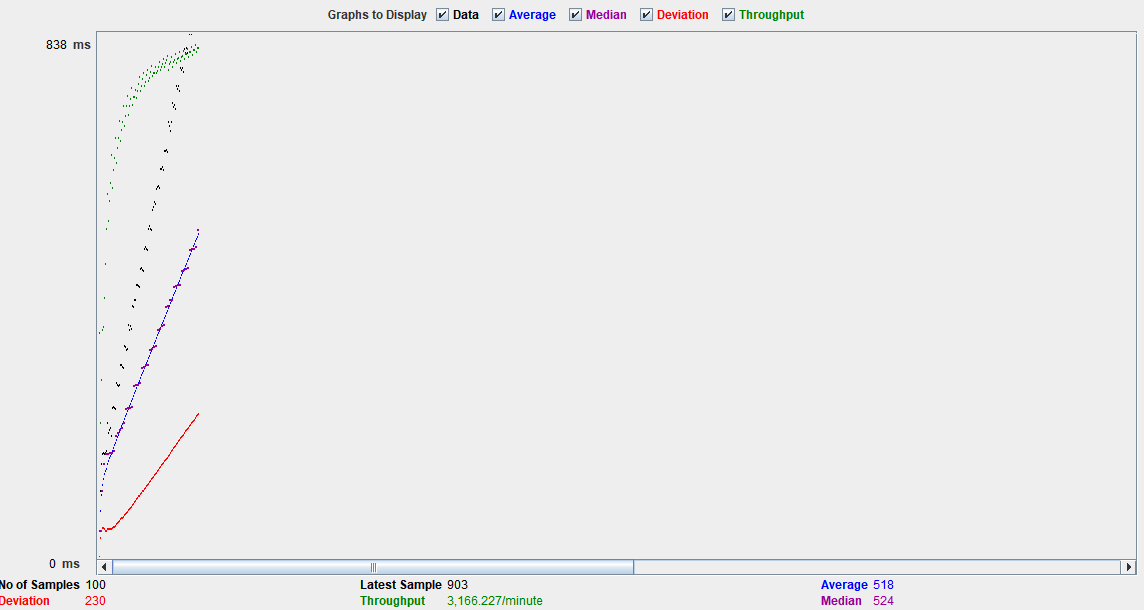
1000 User with average time of 3542 ms

10,000 User with average time of 19768 ms

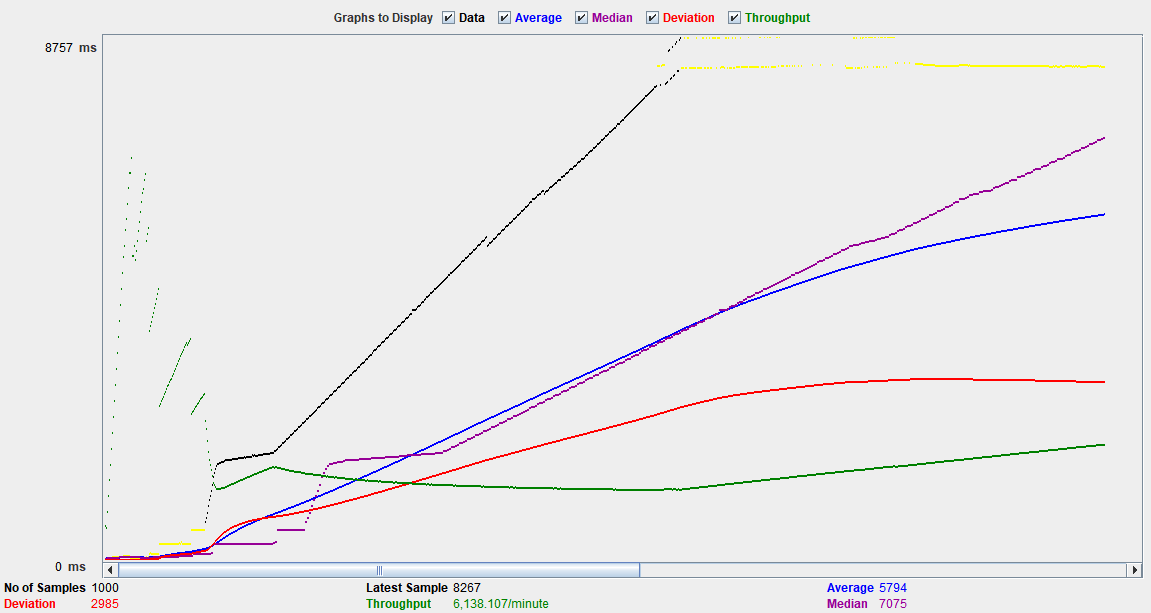


## User Registration

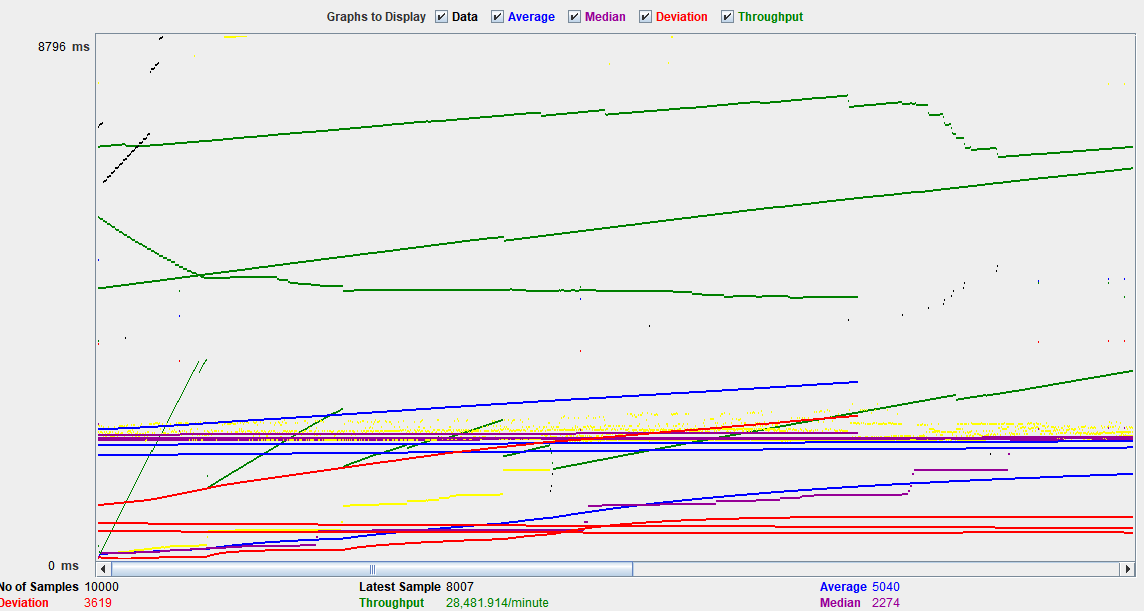
100 Users with average time of 518 ms



1000 Users with average time of 5794 ms

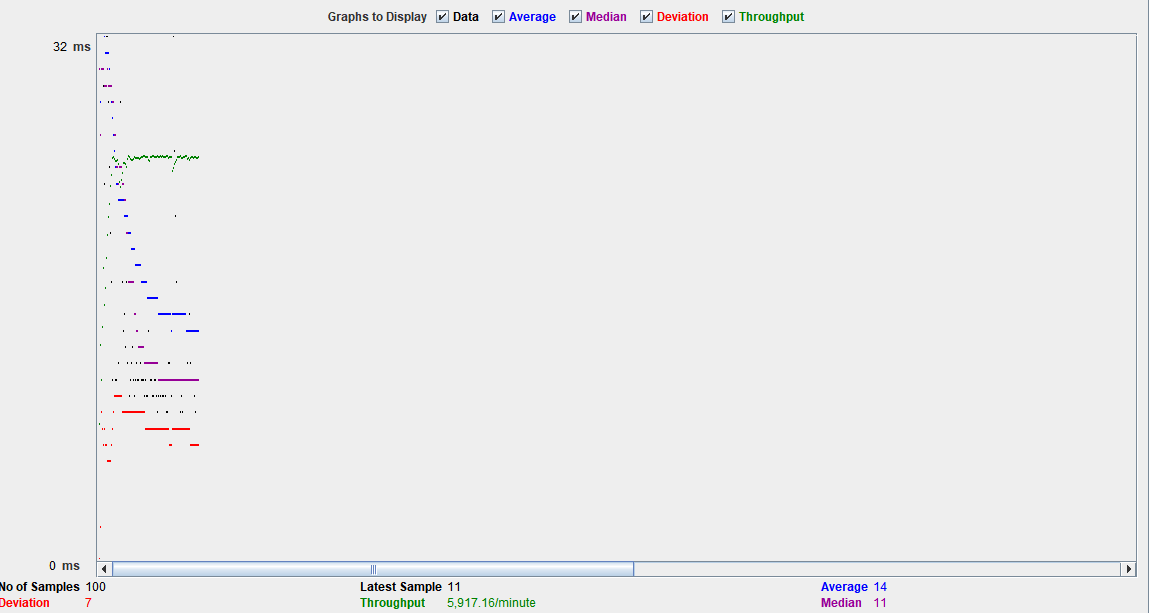


10,000 Users with average time of 5040 ms

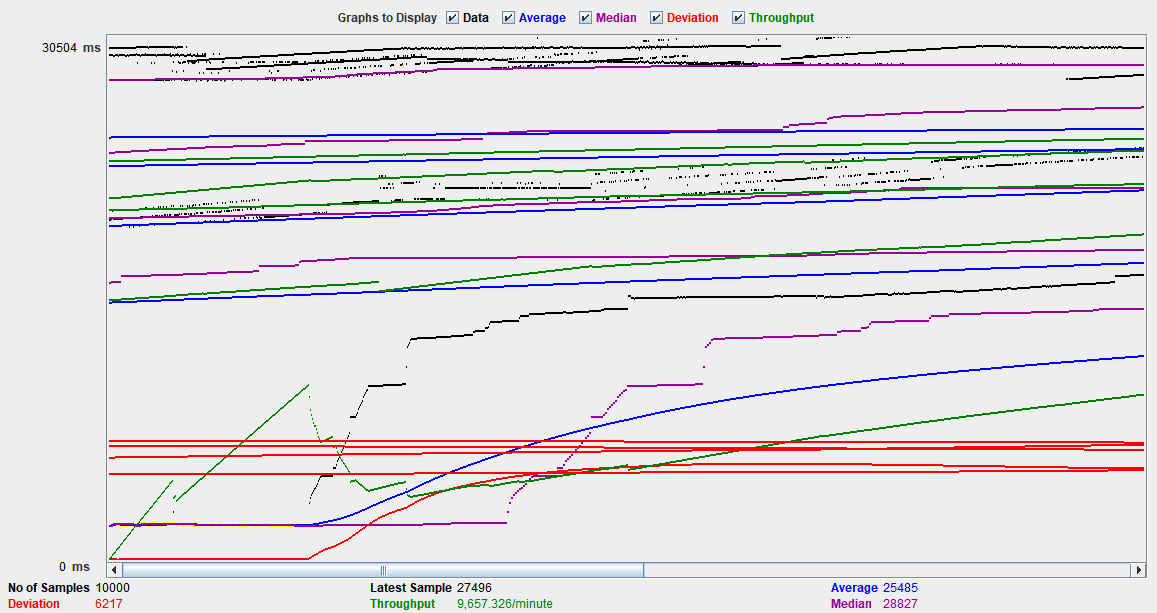


## Bookings

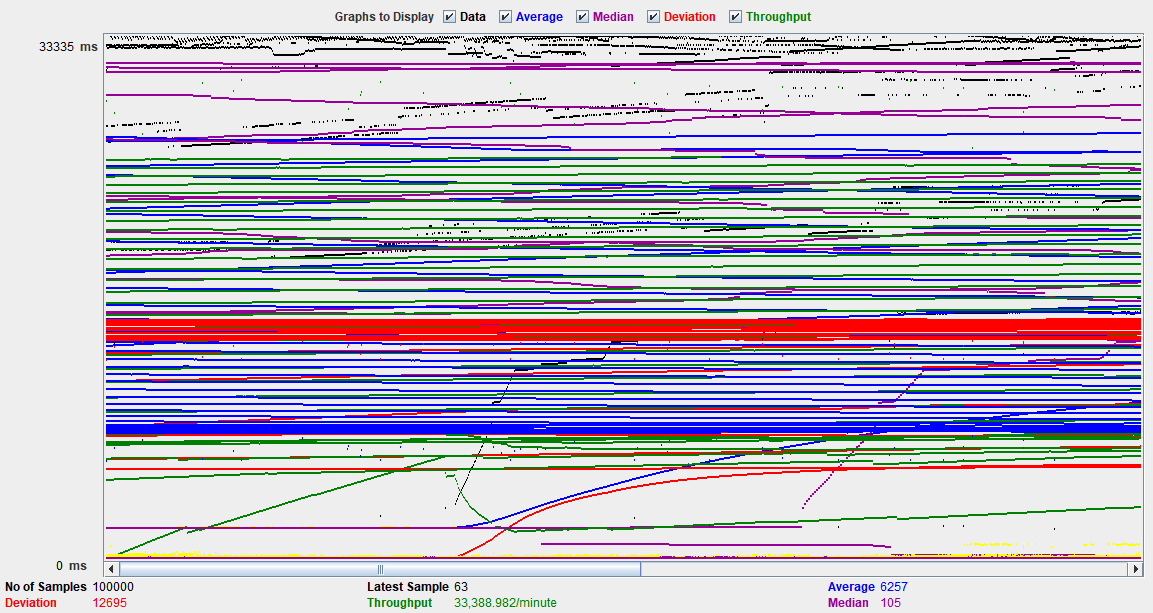
100 Users with average time of 14 ms



10,00 Users with average time of 25485 ms

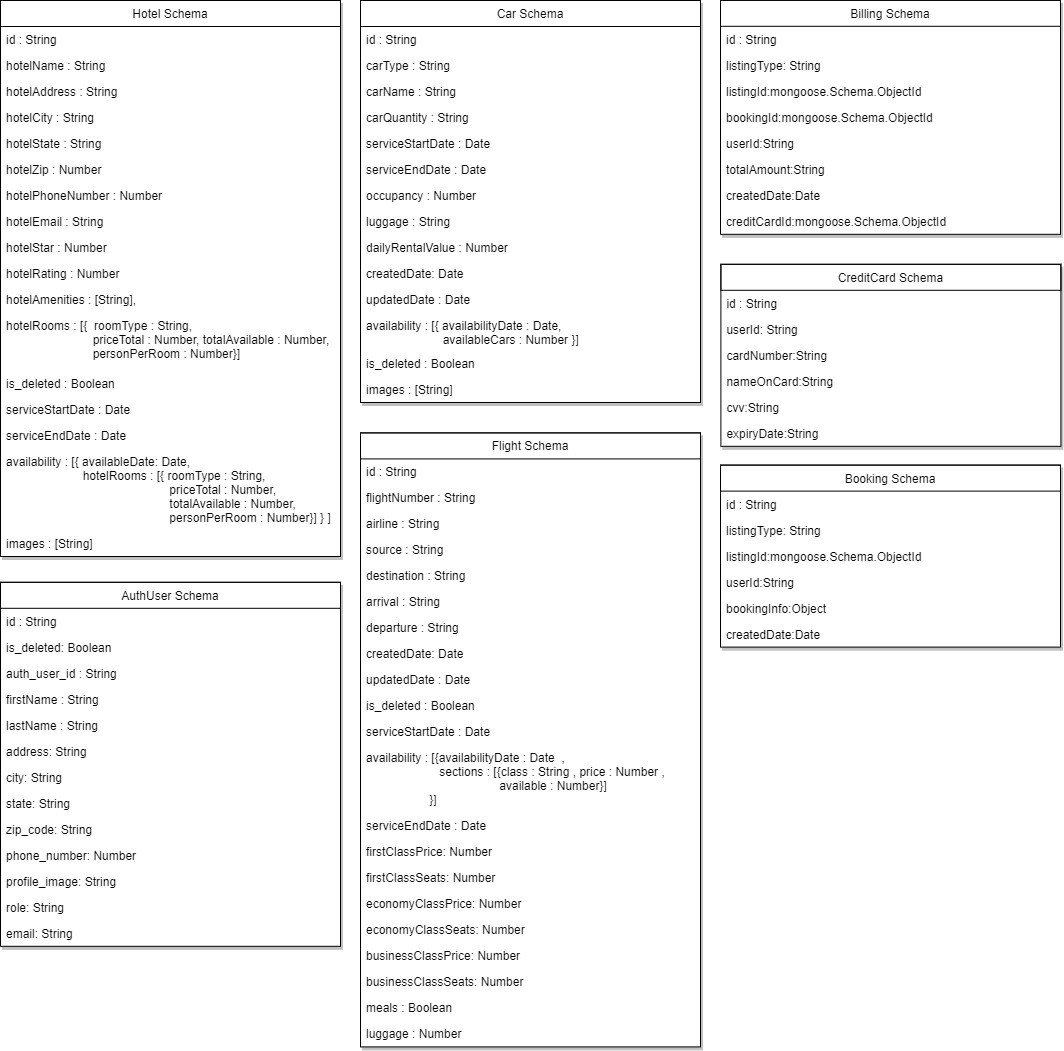


100,00 Users with average time of 6257 ms



# MOCHA TESTING

# DATABASE SCHEMA



# 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 Mangoose 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.

## Lesson 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.