HELWAN UNIVERSITY
Faculty of Computers and Artificial Intelligence
Information System Department

Time-To-Travel

A graduation project dissertation by:

Shenouda Farouk Wahba

Lamees Mohamed Faraj

Ahmed Hassanein Mohamed

Marina Saleh Azeiz

Eman Mohamed Shehata

Maryan Fathy Labib

Submitted in partial fulfilment of the requirements for the degree of Bachelor of Science in Computers & Artificial Intelligence, at the Computer Science Department, the Faculty of Computers & Artificial Intelligence, Helwan University

Supervised by:

Dr. Laila Abdel-Hamid

Eng. Amr Essam

July 2020

جامعة حلوان
كلية الحاسبات والذكاء الإصطناعي
قسم علوم الحاسب

Time-To-Travel

رسالة مشروع تخرج مقدمة من:

شنودة فاروق وهبه (20150280)

لميس محمد فراج (20160317)

احمد حسانين محمد (20150029)

مارينا صالح عزيز (20160319)

ايمان نحمد شحاته (20160096)

ماريان فتحي لبيب (20160318)

رسالة مقدمة ضمن متطلبات الحصول على درجة البكالوريوس في الحاسبات والذكاء الإصطناعي، بقسم علوم الحاسب، كلية الحاسبات والذكاء الإصطناعي، جامعة حلوان

تحت إشراف:

(د/ ليلى عبدالحميد)

(م/عمرو عصام)

يوليو 2020

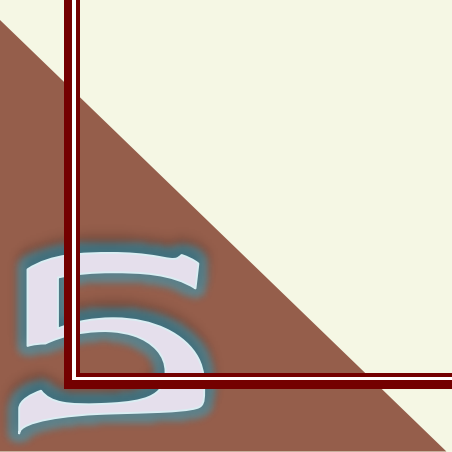
Table Of Contents

Acknowledgements:

- This application has been implemented to facilitate tourists in booking tourist places and see day programs for each place to save time
- This is a 3-tier application that implements the functions for different travel services.
- We have implemented the idea by helping some of the professors in my collage and my friends that work as developer.
- If we have any bugs or errors in this application, we solve them by google it

Goal:

- The goal is to build a web application which enables the user to search for Cars, Hotels and Flights and make a booking. Admin can add Flights, Cars and Hotels and also be able to delete the user and view the statistics.

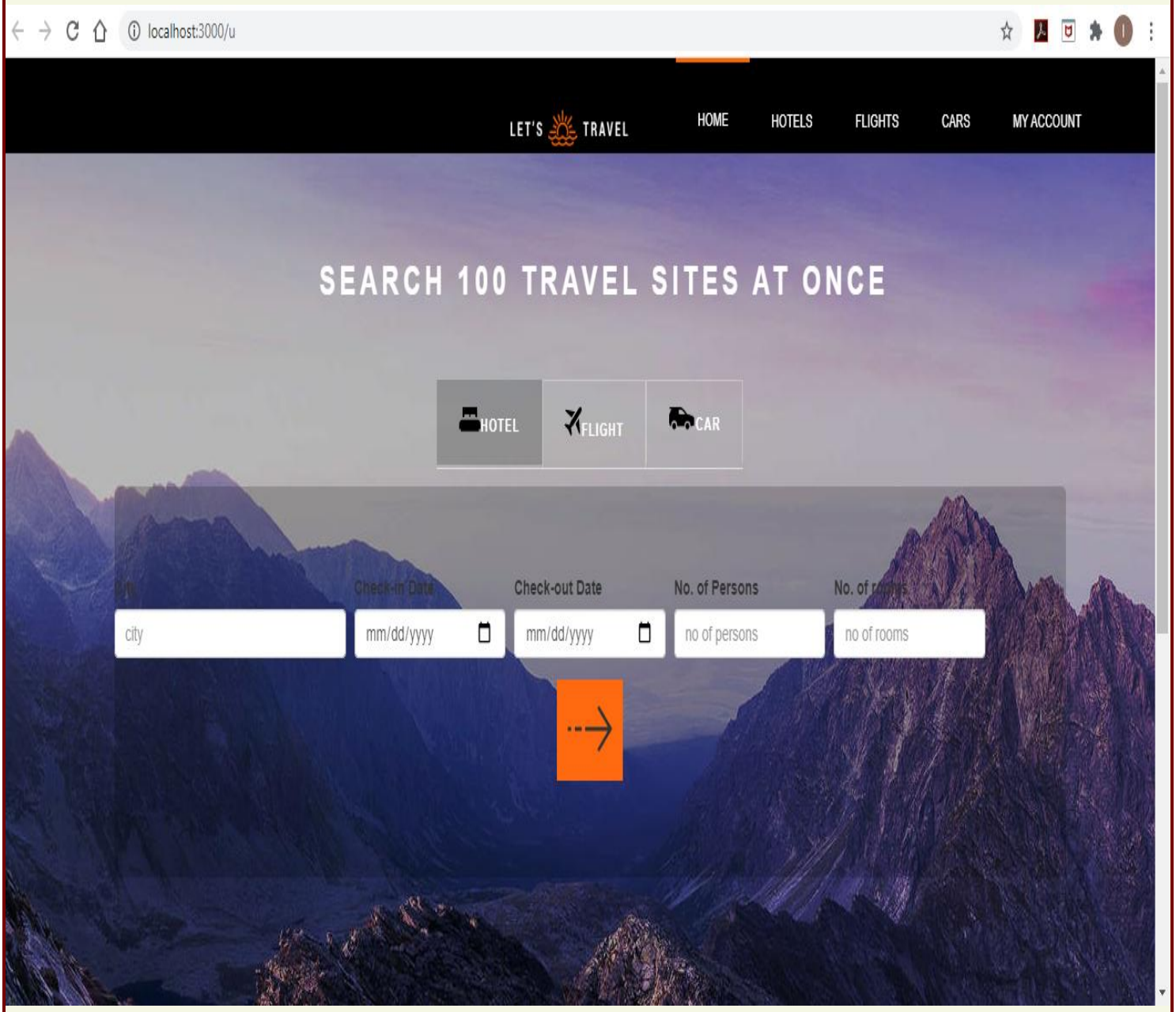


Abstract :

Application idea based on 2 users (manager - tourist)

1. Tasks of the director

- Management account: [Add-Edit-View-Delete]
- Manage transportation: [Add-Edit-View-Delete]
- Hotel management: [Add-Edit-View-Delete]
- Booking management: [transportation-Hotels]
- Manage Ranking with travel company
- Make analytical views



User Tasks:

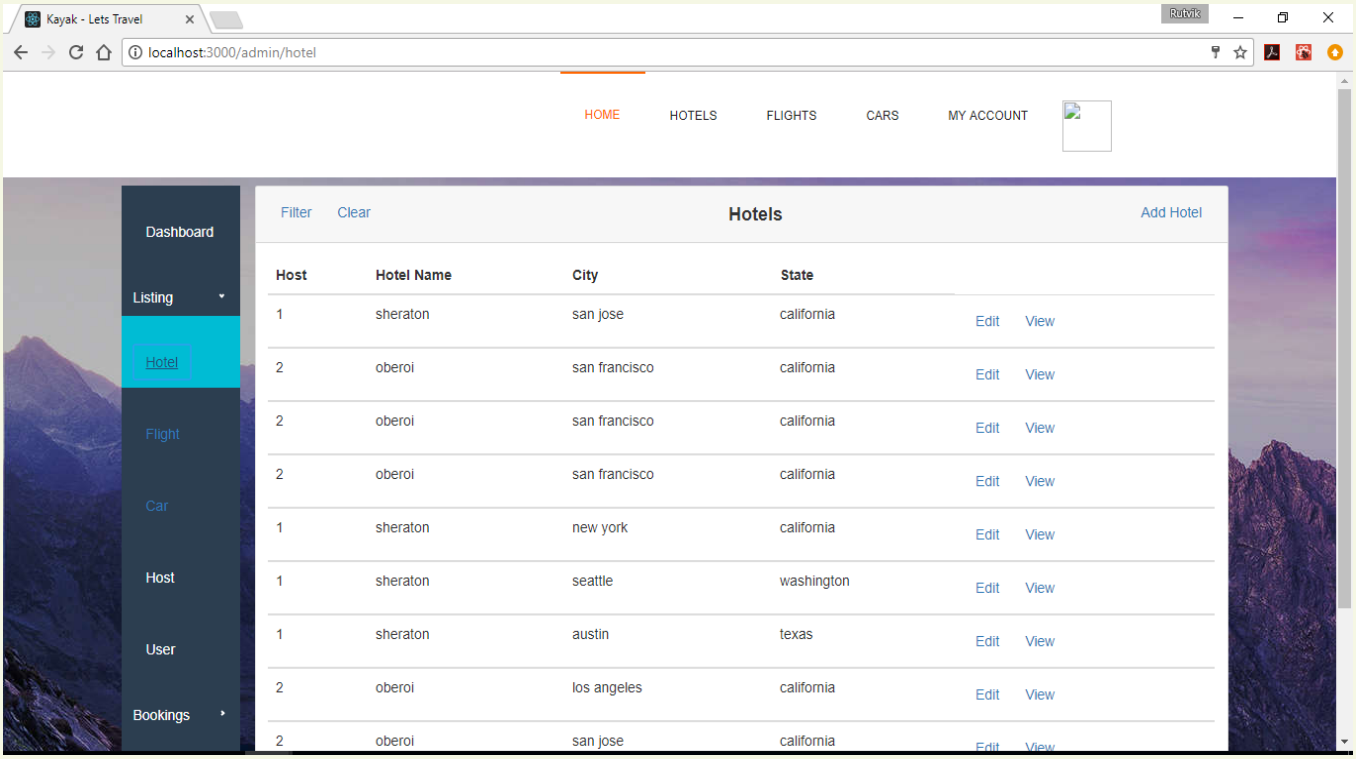
- Users register their data and login

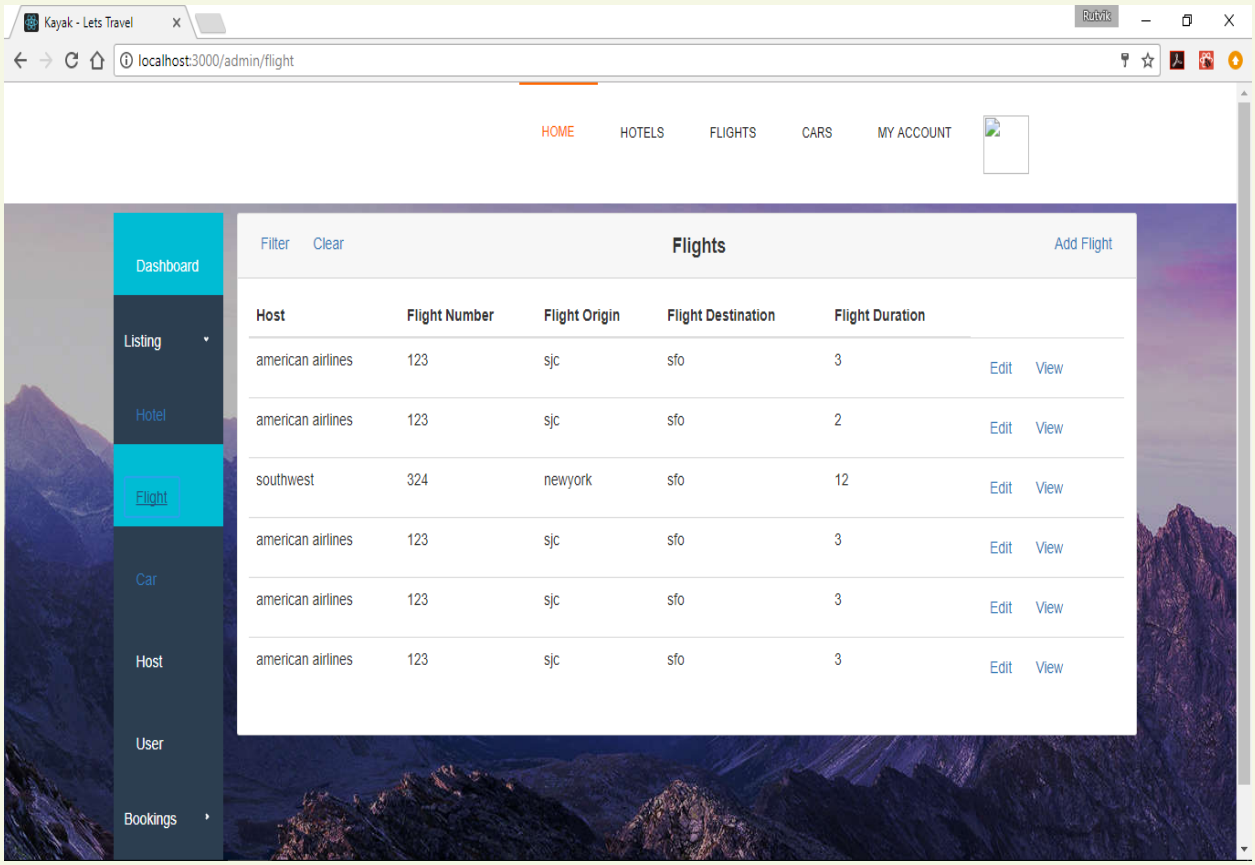
- Reservation of tourist places by seeing the tourist system for every place every tourist system for one place and its price is different

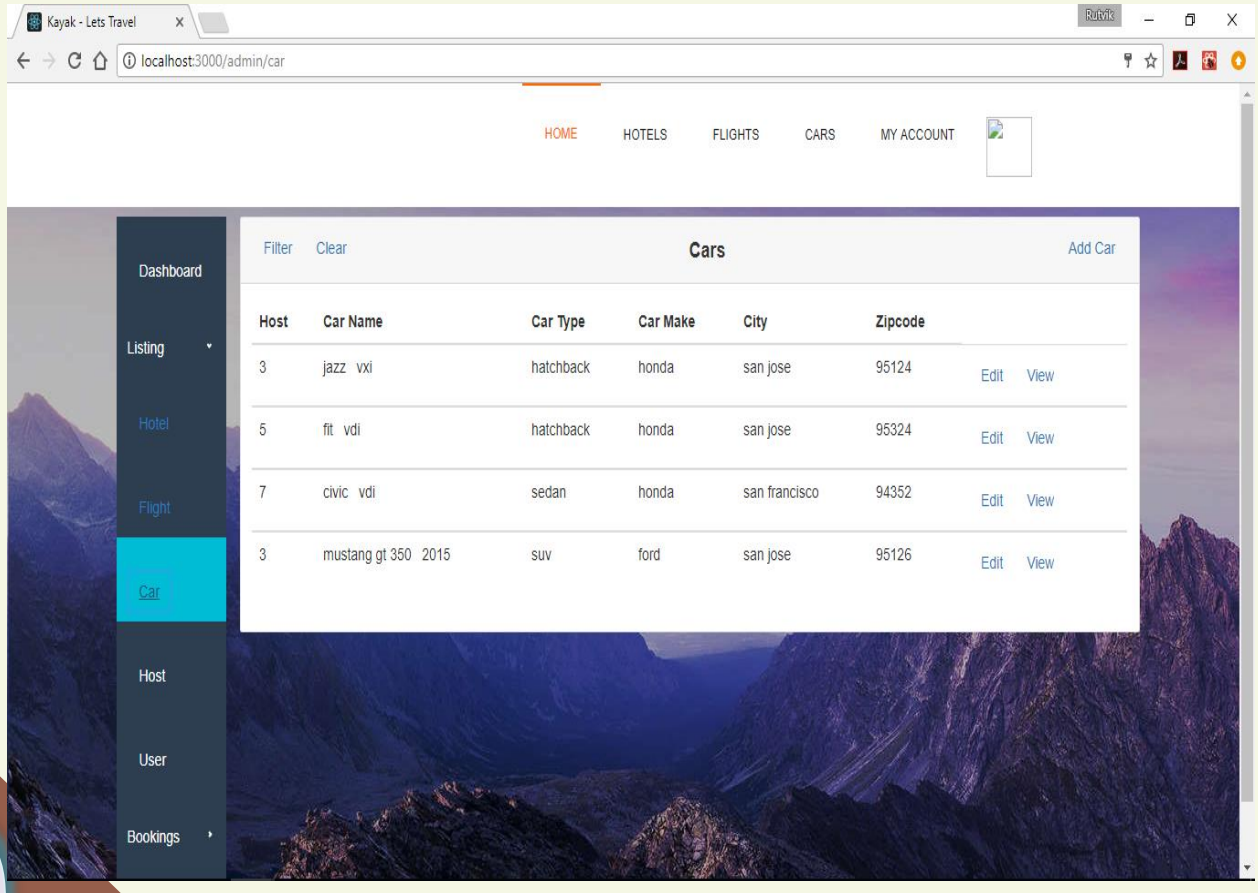
- Book a hotel by choosing the desired hotel and writing the number of people and the number of days

- Book any transportation (car-flight)

- Seeing the requests of hotels and tourist places and can cancel any request them




Kayak - Lets Travel
localhost:3000/admin/flight
HOME
HOTELS
FLIGHTS
CARS
MY ACCOUNT
Dashboard
Listing
Hotel
Flight
Car
Host
User
Bookings
Filter
Clear
Flights
Add Flight
Host
Flight Number
Flight Origin
Flight Destination
Flight Duration
american airlines 123 sjc sfo 3 Edit View
american airlines 123 sjc sfo 2 Edit View
southwest 324 newyork sfo 12 Edit View
american airlines 123 sjc sfo 3 Edit View
american airlines 123 sjc sfo 3 Edit View
american airlines 123 sjc sfo 3 Edit View


Kayak - Lets Travel
localhost:3000/admin/car
HOME
HOTELS
FLIGHTS
CARS
MY ACCOUNT
Dashboard
Listing
Hotel
Flight
Car
Host
User
Bookings
Filter
Clear
Cars
Add Car
Host
Car Name
Car Type
Car Make
City
Zipcode
3 jazz vxi hatchback honda san jose 95124 Edit View
5 fit vdi hatchback honda san jose 95324 Edit View
7 civic vdi sedan honda san francisco 94352 Edit View
3 mustang gt 350 2015 suv ford san jose 95126 Edit View

Chapter 1 : Introduction

1.1 Overview

The objective of the project is to develop ‘time-for-travel’ prototype. ‘time-for-travel’ is a travel metasearch engine, which enables users to search and book hotels, flights, and cars. ‘time-for-travel’s other services include packages, rentals, cruises, guides, trains, flight tracker, routes, and deals. The application compares the prices with other websites and views the best deal.

Users are able to create (sign-up) an account. In order to view/manage their booking they must login and click account preferences under the Account tab. They have the privilege to set up their own profile image which appears at the top-right corner. Users can view and edit all their personal details under the preferences tab. This information include first name, last name, email, gender, address (street, city, state, zipcode) , contact info and date of birth.

One can add and view all the credit/debit cards linked to their account. They may also view all of their past bookings (hotel | flights | cars).

We have included validations on every user input of the application. Server is tested for myriad concurrent users at a time. We have used Jmeter as a load testing tool for analyzing

and measuring performance. Mocha, a JavaScript enabled framework serves the purpose testing the REST APIs.

Refer the requirement section for more info on implementation.

1.2 Objectives

The idea of the project came through the difficulty of the tourist to find information about transportation that he is searching for , so the solution was to find an idea that makes it easier for the tourist to collect the information he wants about transportation and hotels .

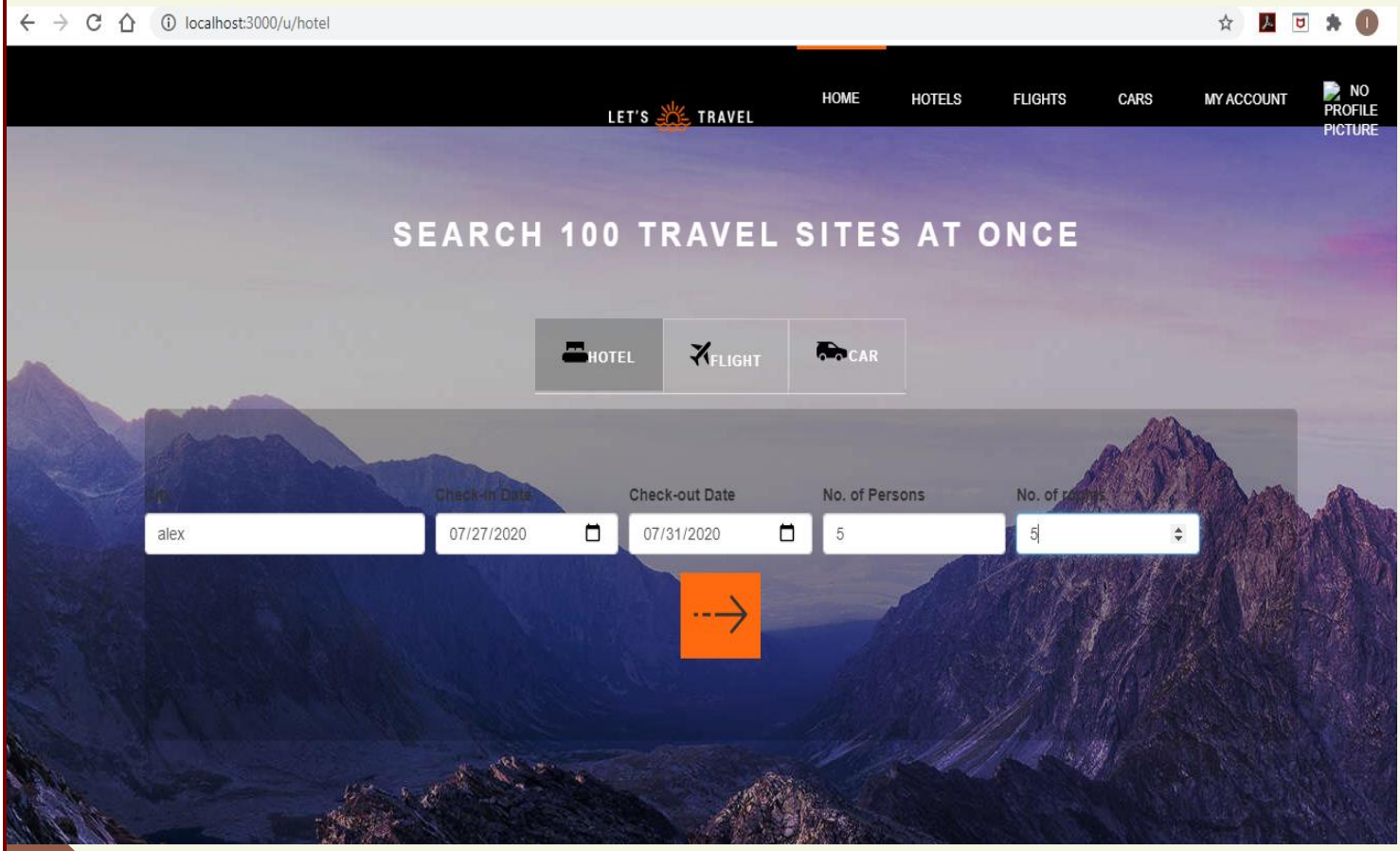
1.3 Purpose of system:

Learn and apply MERN stack to develop an online travel reservation system and use distributed publish-subscribe messaging system like Kafka to develop scalable, durable, reliable, and high-throughput distributed system. Moreover, we have used Redis for cache management.

- The Application is used to search for flights/cars/hotels and make a booking based on the options set by the user. The purpose of each module developed in this application is as follows:

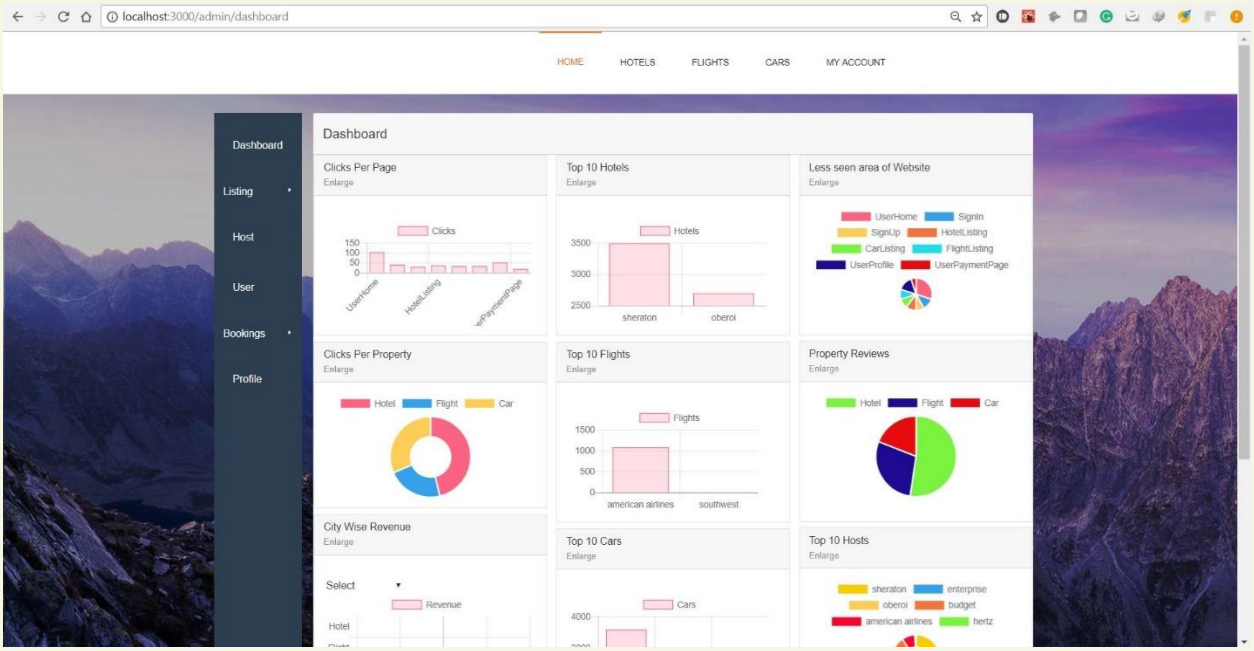
1. <u>User Module:</u>

   Create a new user |Delete the user account |Change and update user information |Display information about the user in profile section |Search listing for different categories |Filter Listings based on user search criteria
   a.Filter Hotels based on stars, price b.Filter Flights based on departure, arrival dates and price c.Filter cars based on car type, price |Book a Hotel/Car or Flight |Make Payment and view purchase summary |View Past/Future Bookings |Upload a profile picture and update profile picture



2. Admin Module:

Only user with Admin credentials can view this module |Stats Page - Consists of Graphs and statistics |Manage Listings: A.Add a Listing a.Car b.Flight c.Hotel B.Update a listing a.Car b.Flight c.Hotel |Manage Users |Search Payment information based on date and months |Mange Admin Profile |Log out



1.4 Scope

In the beginning we made a comprehensive plan for the project and drew the flow chart

Then we did the software diagrams and then we visualized the shape of the project and applied in the form of UI&UX and the we started in the coding phase

1.5 General Constraint

We have encountered many obstacles that have made it difficult for us to implement the project such as :

- Find a free server for the application
- Implementing a project of this magnitude and trying to modify the problem was face.
- Collection of data on tourist transportation and hotels serving the user .

Chapter 2: Project "Planning and analysis"

2.1 Project planning:

- Understand the scope and value of the project
- Conduct extensive research
- Ask the tough questions
- Create your project plan outline
- Talk with your team
- Write your full project plan
- Execute your plan in TeamGantt
- Publish your plan
- Share your plan with the team and make sure they read it!
- Prepare to keep planning

2.1.1 Feasibility Study

- **Travel Right** is performed, which determines whether the solution considered to accomplish the requirements is practical and workable in the software. Information such as resource availability, cost estimation for software development, benefits of the software to the organization after it is developed and cost to be incurred on its maintenance are considered during the feasibility study. The objective of the feasibility study is to establish the reasons for developing the software that is acceptable to users, adaptable to change and conformable to established standards. Various other objectives of feasibility study are listed below.

    - To analyze whether the software will meet organizational requirements.
    - To determine whether the software can be implemented using the current technology and within the specified budget and schedule.
    - To determine whether the software can be integrated with other existing software.

2.1.2 Estimated Cost

- Server Cost: 4000 EGP Per Year
- Domain Cost: 1200 EGP
- Marketing Costs: 12000 EGP Per Year
- Other Cost: 2000 EGP

2.2 Analysis and Limitation of existing system:

- The system is slow when it contains many data on the database the application is not loaded on the server so the download time will be more than that is uploaded to the server.

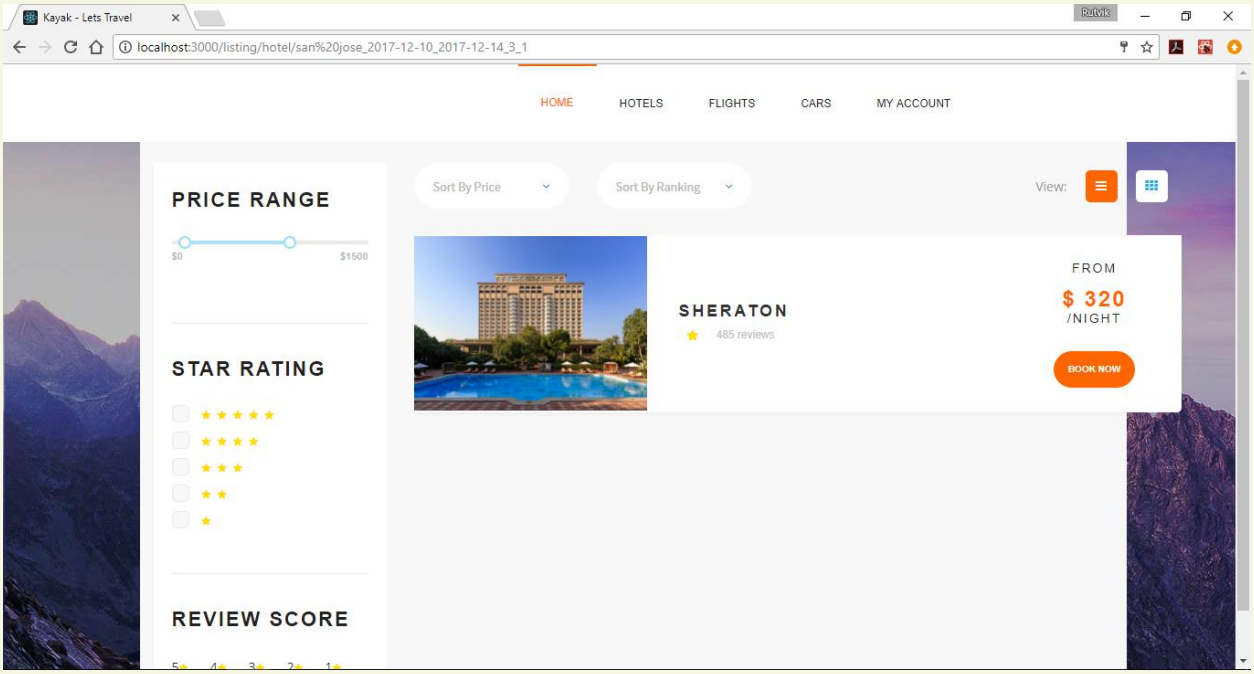
2.3 Need for the new system

- Possible to move from the old system to the new system.
- Modern technology through which the system becomes faster and performance becomes better
- The user does not know how the system works, but what is important is the system performance is faster and provides a good service

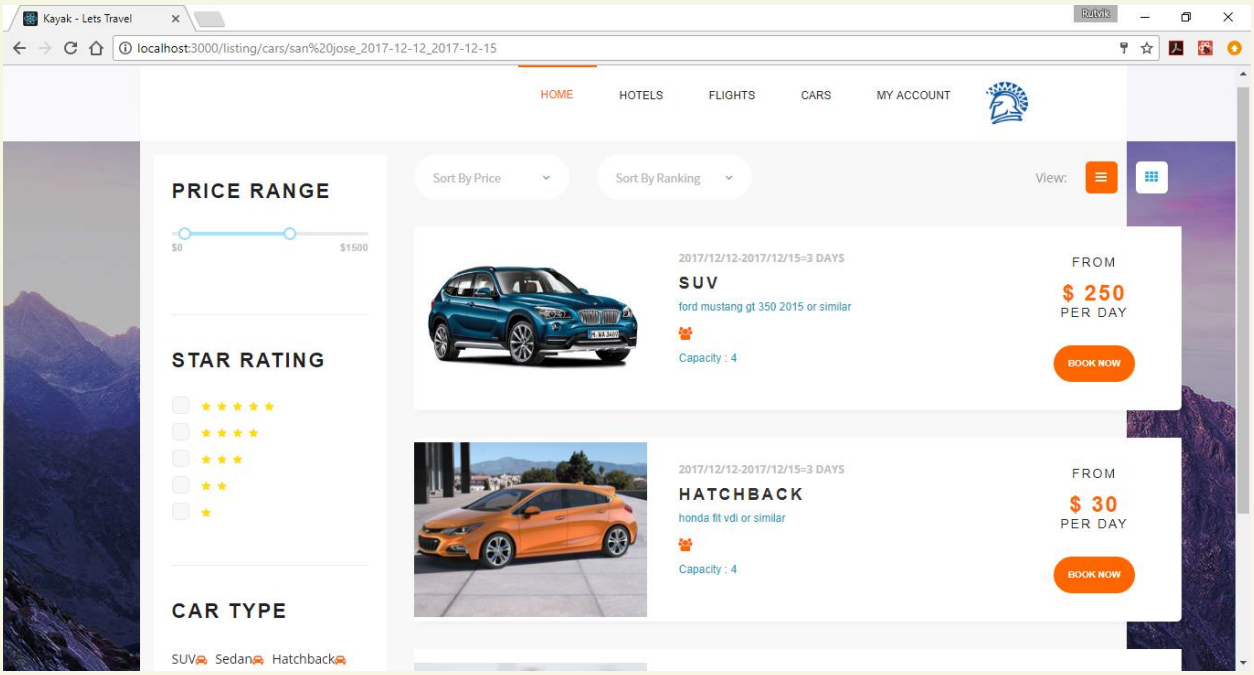
2.4 Analysis of the new system:

2.4.1 user requirement

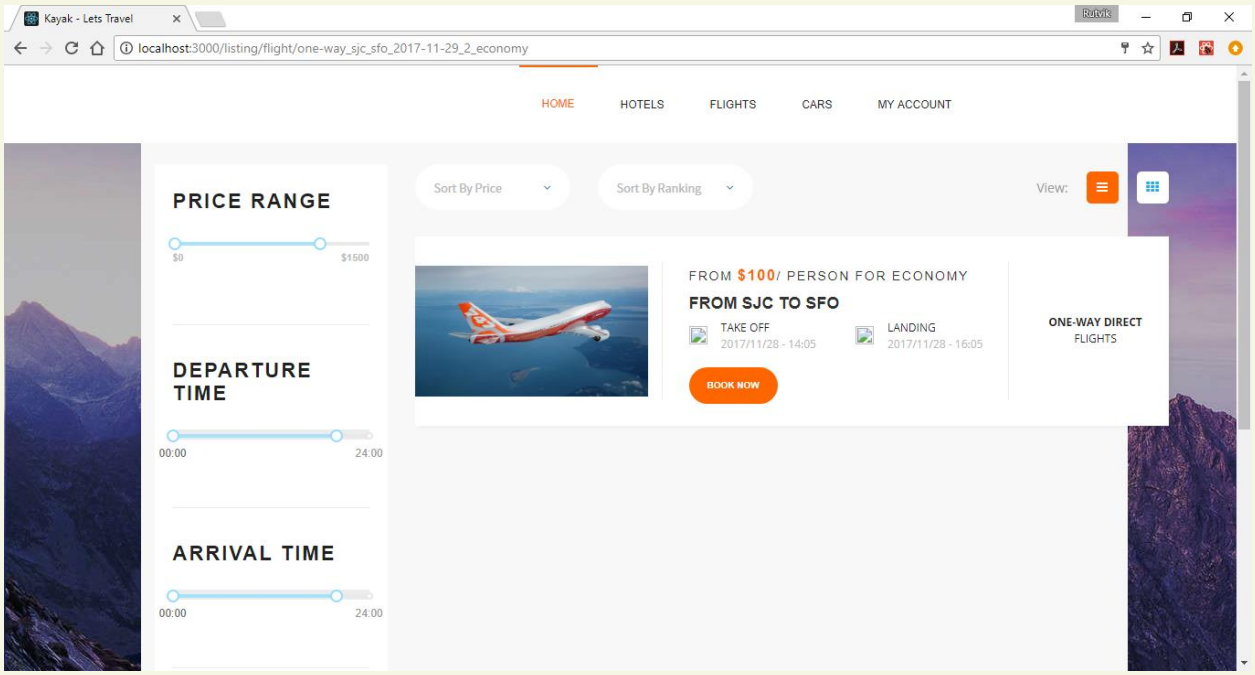
- Manipulate users
- Manipulate hotels



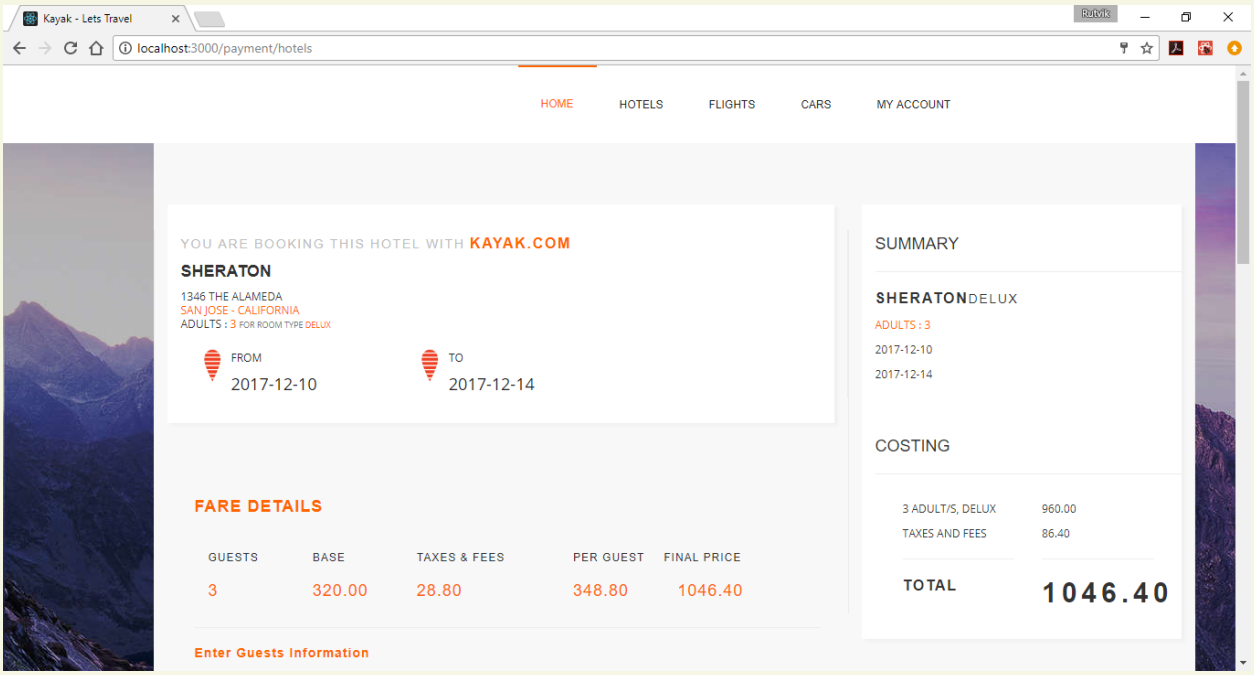
- Manipulate car



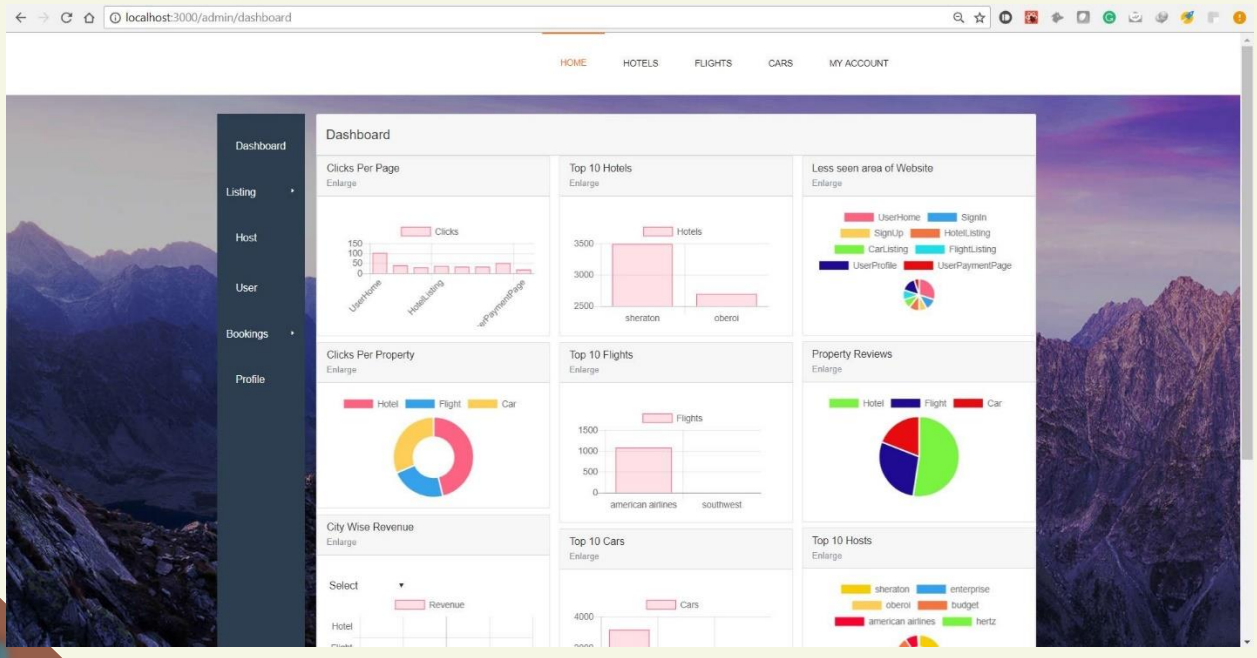
- Manipulate flights



- Manipulate Booking



- Manipulate payment
- View analytical charts



2.4.2 system requirement

1. Accessibility
2. Accuracy
3. Audit, control, and reporting
4. Availability
5. Backup and restore
6. Capacity, current and forecast
7. Certification
8. Compliance
9. Compatibility of software tools, standard, platform, database, and the like.
10. Concurrency
11. Configuration management
12. Dependency on other parties
13. Deployment
14. Documentation

15. Disaster recovery
16. Efficiency (resource consumption for given load)
17. Effectiveness (resulting performance in relation to effort)
18. Environmental protection
19. Emotional factors (like fun or absorbing)
20. Error handling
21. Escrow
22. Exploitability
23. Failure management
24. Interoperability
25. Legal and regulatory
26. Licensing
27. Localizability
28. Maintainability
29. Modifiability

30. Network topology
31. Open source
32. Price
33. Performance/response time
34. Privacy
35. Quality
36. Portability
37. Redundancy
38. Recovery
39. Reliability
40. Reporting
41. Resilience
42. Response time
43. Resource constraints
44. Security

45. Scalability
46. Stability
47. Safety
48. Supportability
49. Testability
50. Throughput
51. Usability

2.4.3 Domain Requirement

- Domain requirements reflect environment in which the system operates so, when we talk about an application domain, we mean environments such as train operation, medical records, e-commerce, etc.

- Domain requirements may be expressed using specialized domain terminology or reference to domain concepts. Because these requirements are specialized, software engineers often find it difficult to understand how they are related to other system requirements

- Domain requirements are important because they often reflect fundamentals of the application domain. if these requirements are not satisfactorily. For example, the requirements for the insulin pump system that delivers insulin on demand include the following domain requirement.

- The system safety shall be assured according to standard IEC 6001-1: Medical Electrical Equipment – Part 1: General Requirements for Basic safety and Essential Performance.

- This requirement means that the developers must be familiar with that standard to ensure that they do not violate it. It constrains both the design of the device and the development process. Other requirements have to be checked against this standard.

- Sometimes, characteristics of the application domain mean that the requirements specification has to include a description of how to carry out some computations. For example, the domain requirement below is included in the requirements specification for an automated train protection system. This system

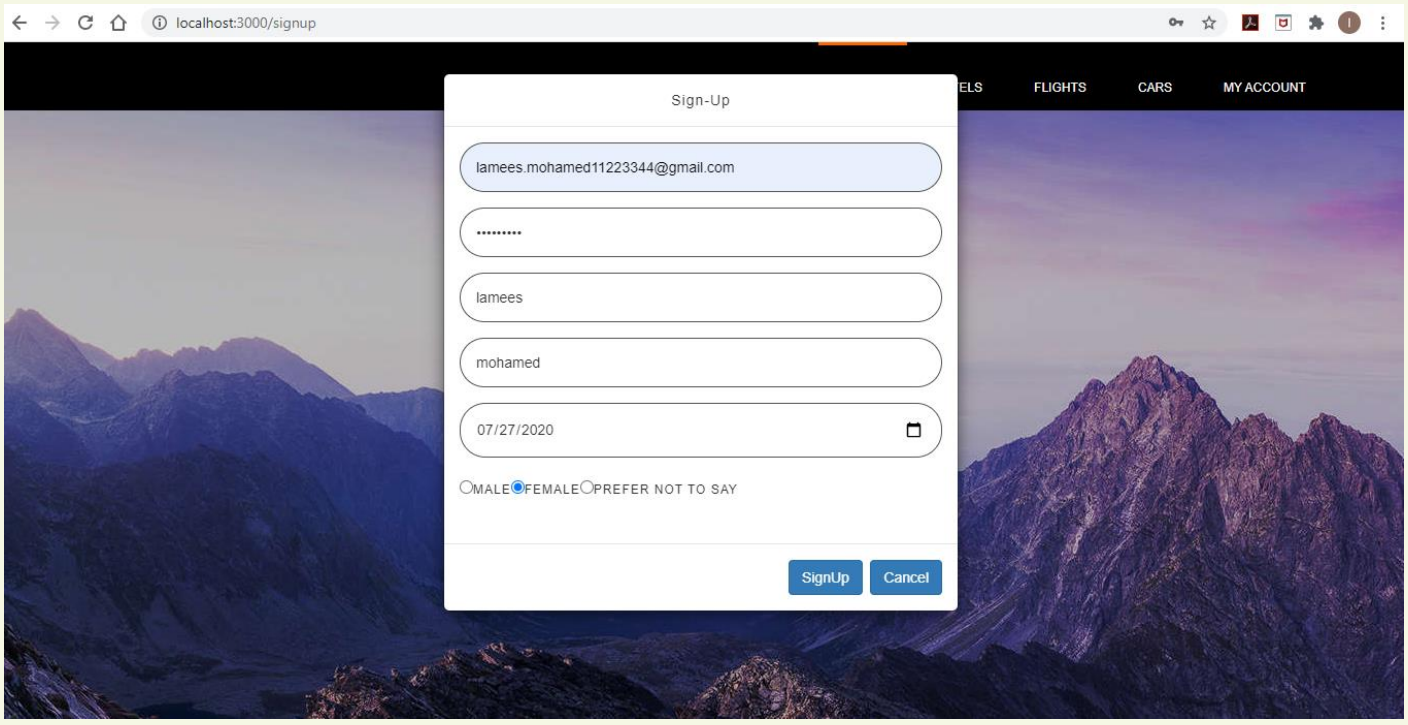
automatically stops a train if it goes through a red signal. This requirement states how the train deceleration is computed by the system. It uses domain-specific terminology. To understand it, you need some understanding of the operation of railway systems and train characteristics

- The requirement for train system illustrates a major problem with domain requirement. they are written in the language of the application domain and it is often difficult for software engineers to understand them simply because it is so obvious to them however, it may not be obvious to the developers of the system and they may therefor implement the requirement in the wrong way.

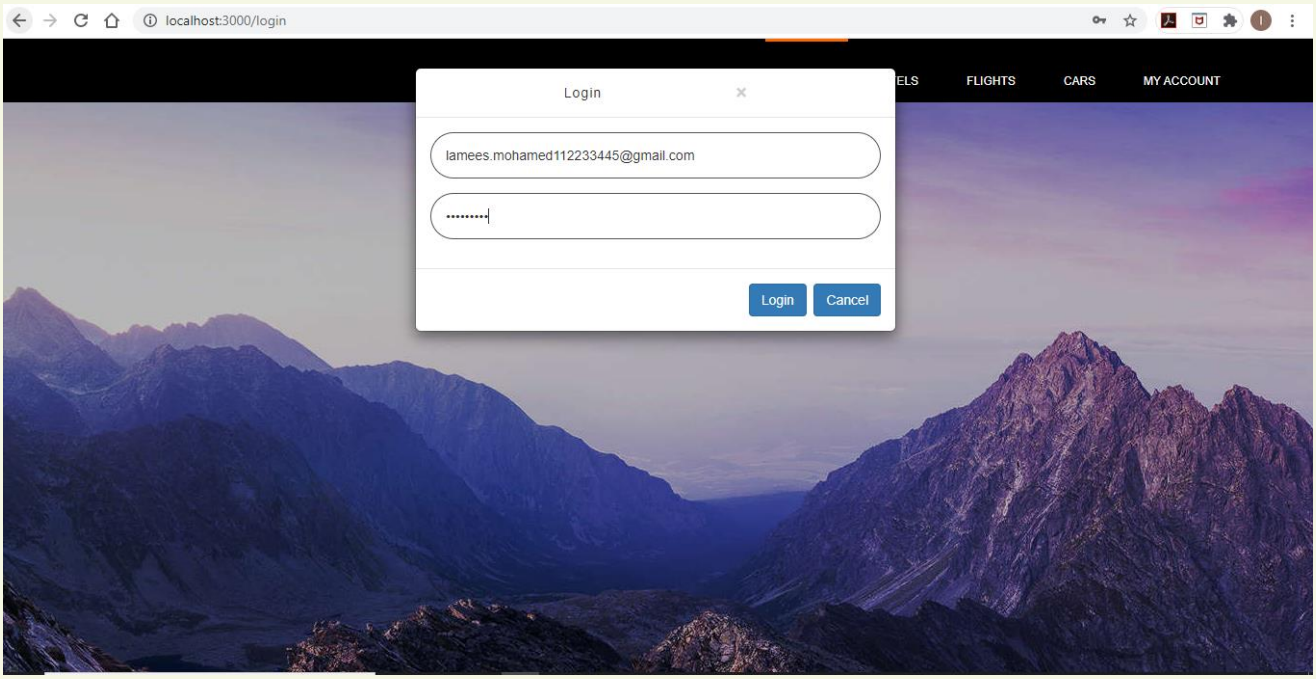
2.4.4 functional requirement

User functionality

Function	Sign up
Parameter	Email – user name –password-date of birth-gender
Description	Make an account to access the system



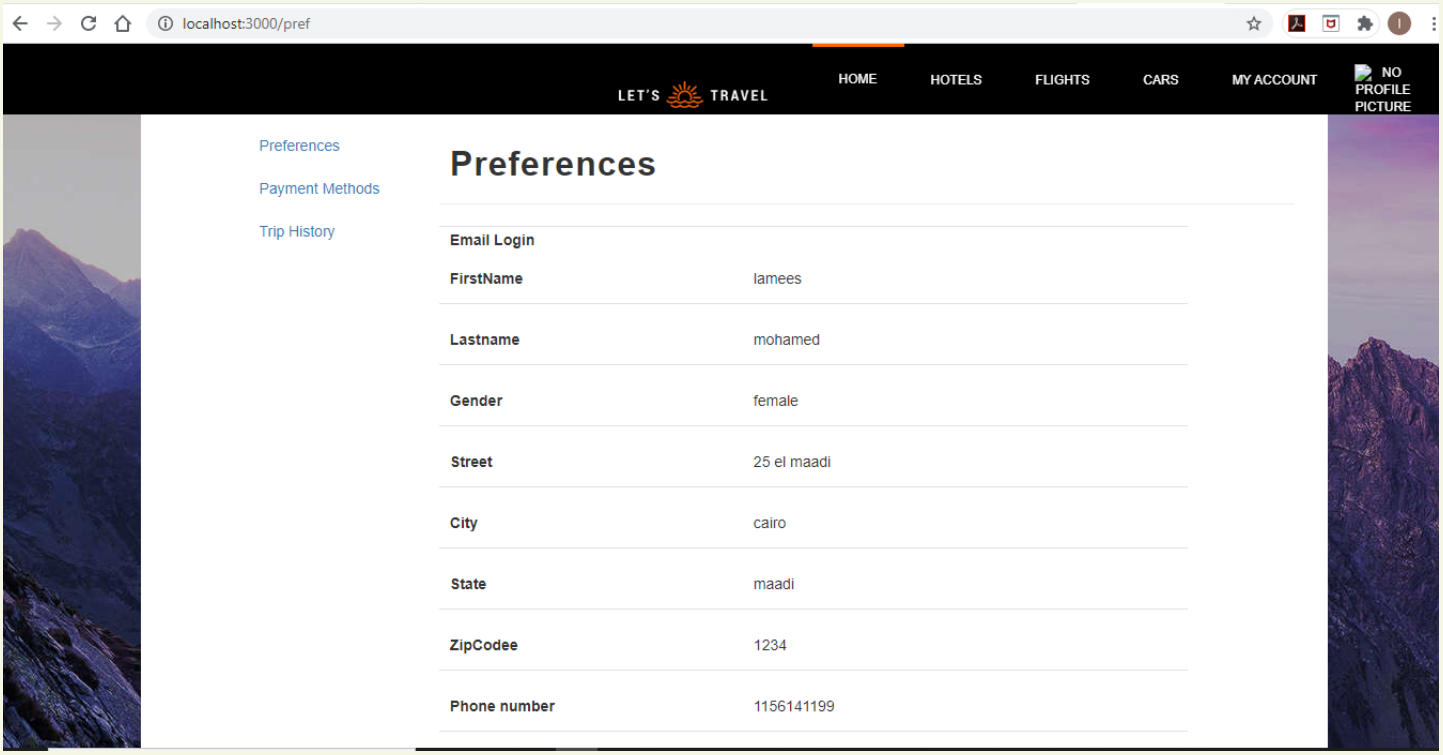
Function	Sign in
Parameter	Email -password
Description	Access the system to make all the available features in the system



Function	Add user
Parameter	Email – user -password
Description	Save user data to the database

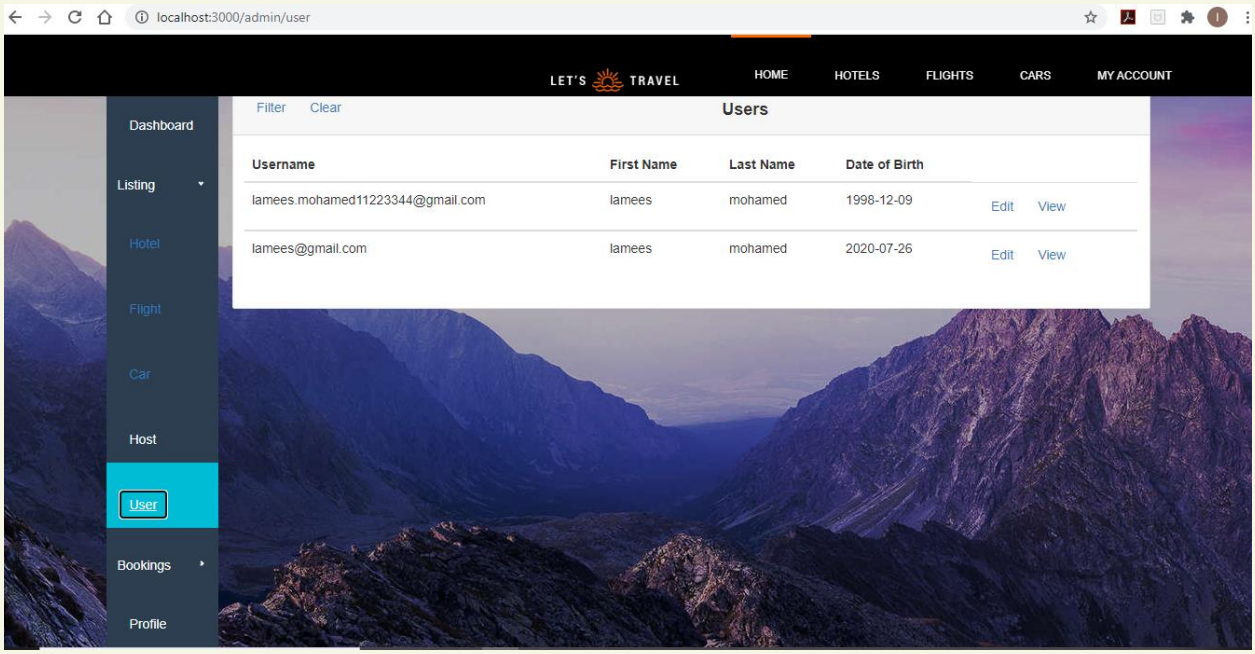
Function	Update user
Parameter	Id –Email –user name -password

Description	Update user data and save to the database

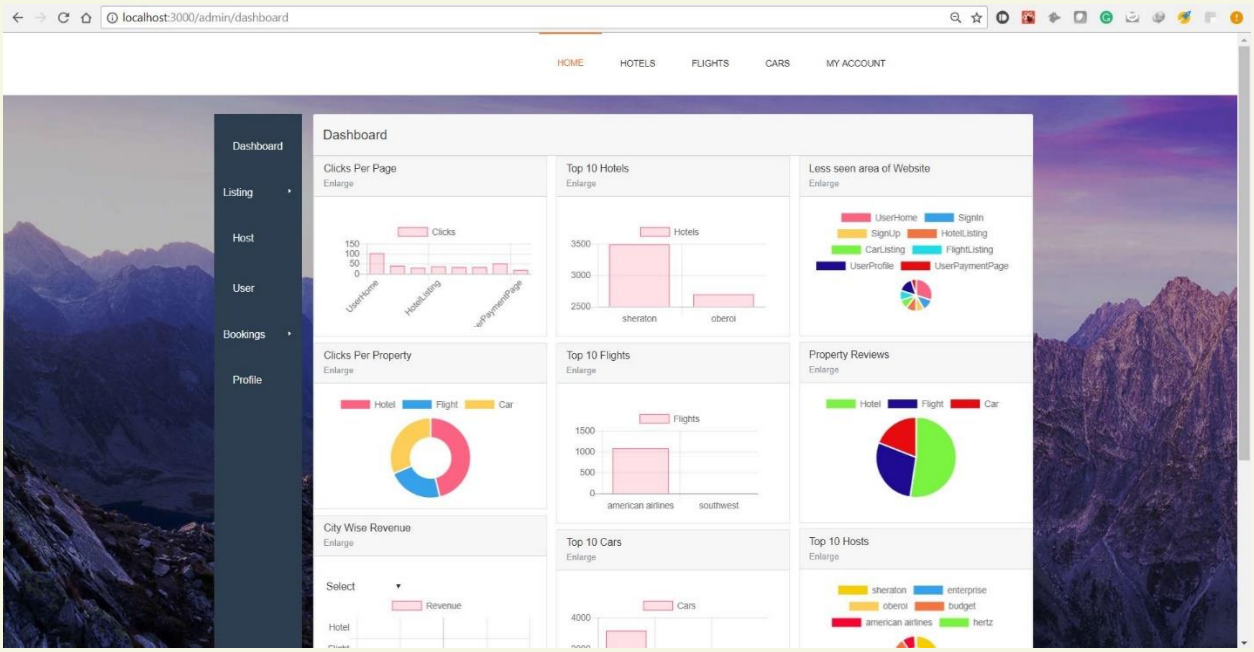


Function	Delete user
Parameter	Id
Description	Delete user data from the database

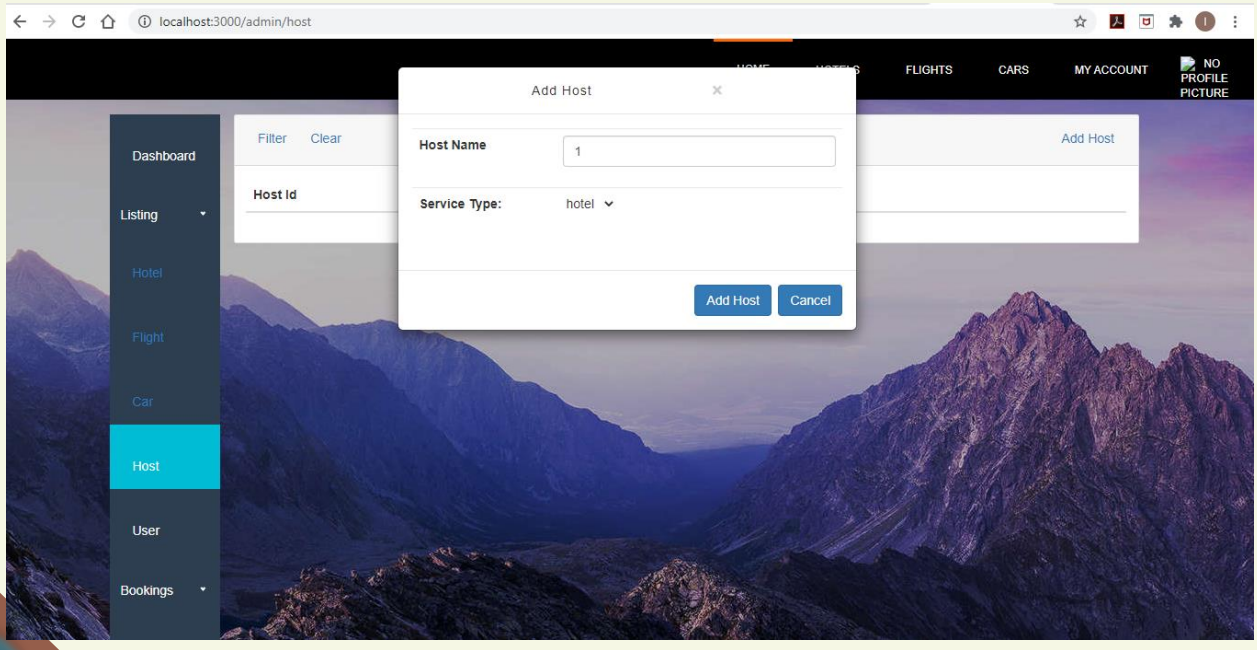
Function	Fetch all users
Parameter	None
Description	Retrieve all users data from database



Function	logAnalyticsData
Parameter	None
Description	Make an analytical charts to view the interaction of the system



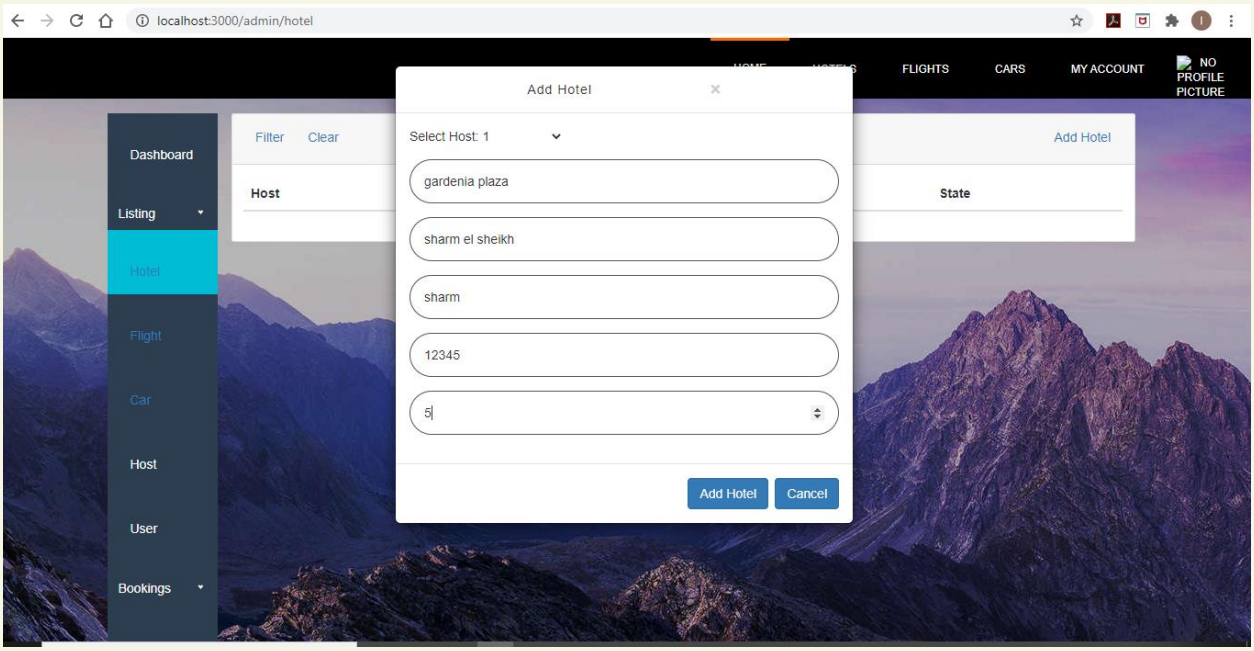
Function	Add host
Parameter	ID – name
Description	Add an host into the database



Function	Modify host data
Parameter	Name
Description	Modify the host and update it in the database

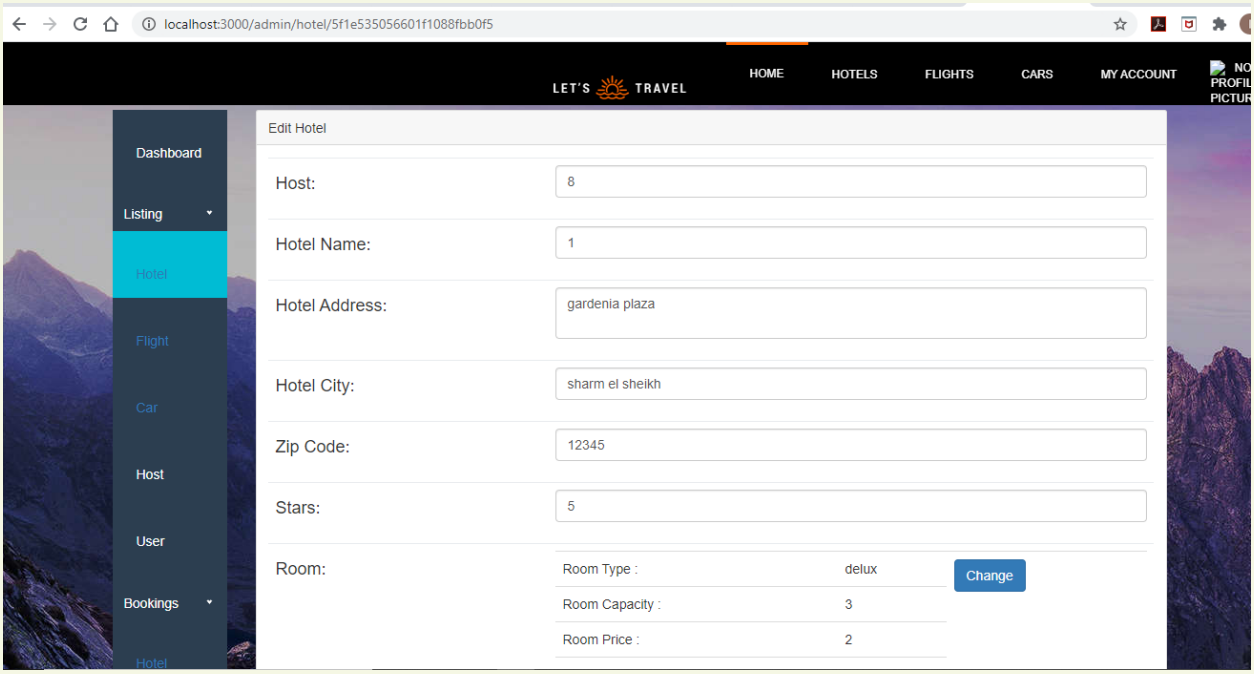
Hotels functionality

Function	Add hotels
Parameter	Name –description –address –image-price-number of rooms
Description	Save hotel data to the database



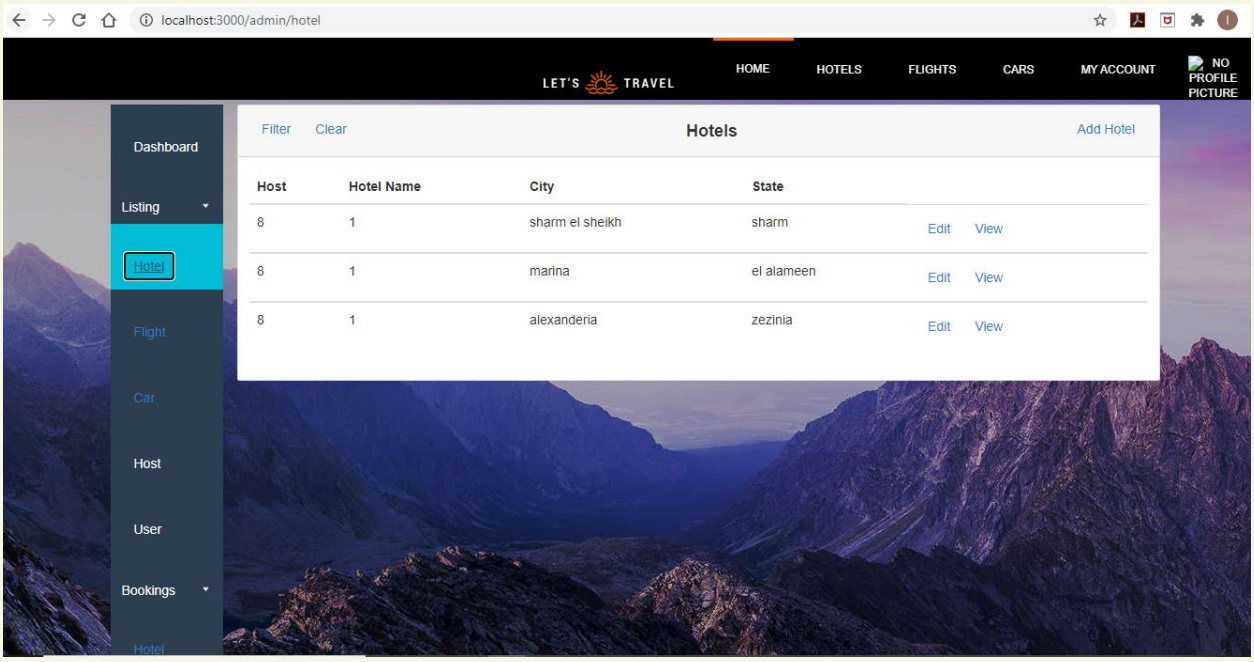
Function	Update hotel

Parameter	Id –name –description-address-image-price –number of rooms
Description	Update hotel data and save to the database



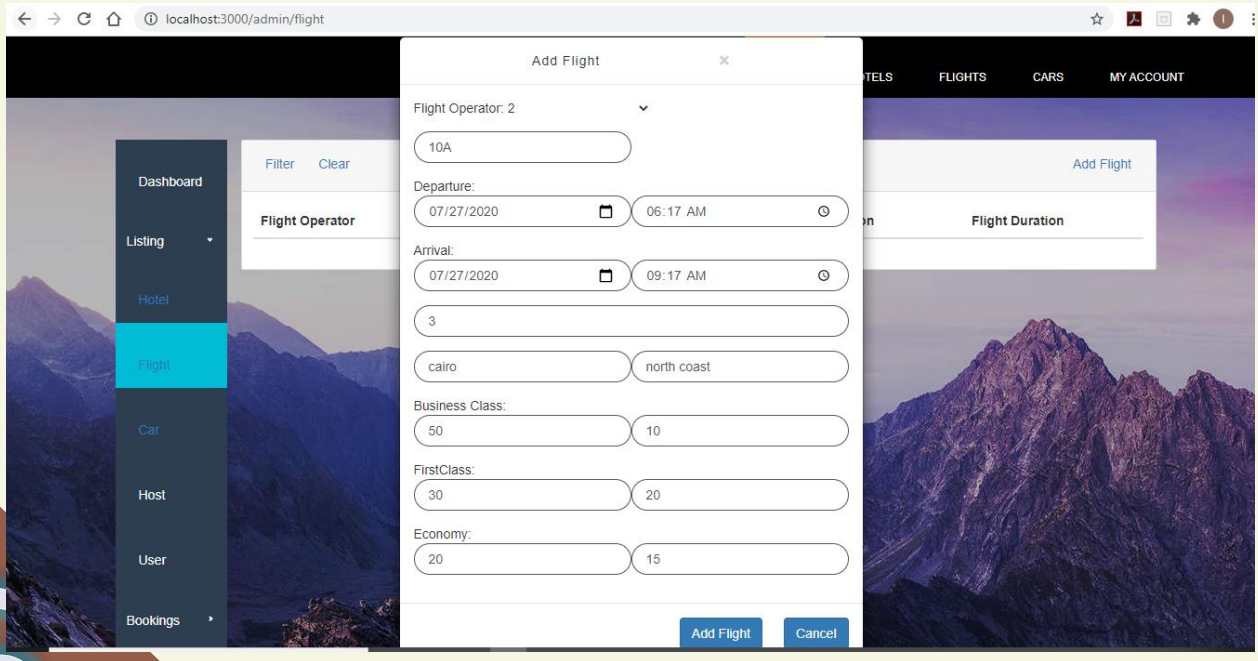
Function	Delete hotel
Parameter	Id
Description	Delete hotel data from the database

Function	View all hotels
Parameter	None
Description	Retrieve all hotels data from database



Flight functionality

Function	Add flight
Parameter	Number of seats
Description	Save flight data to the database



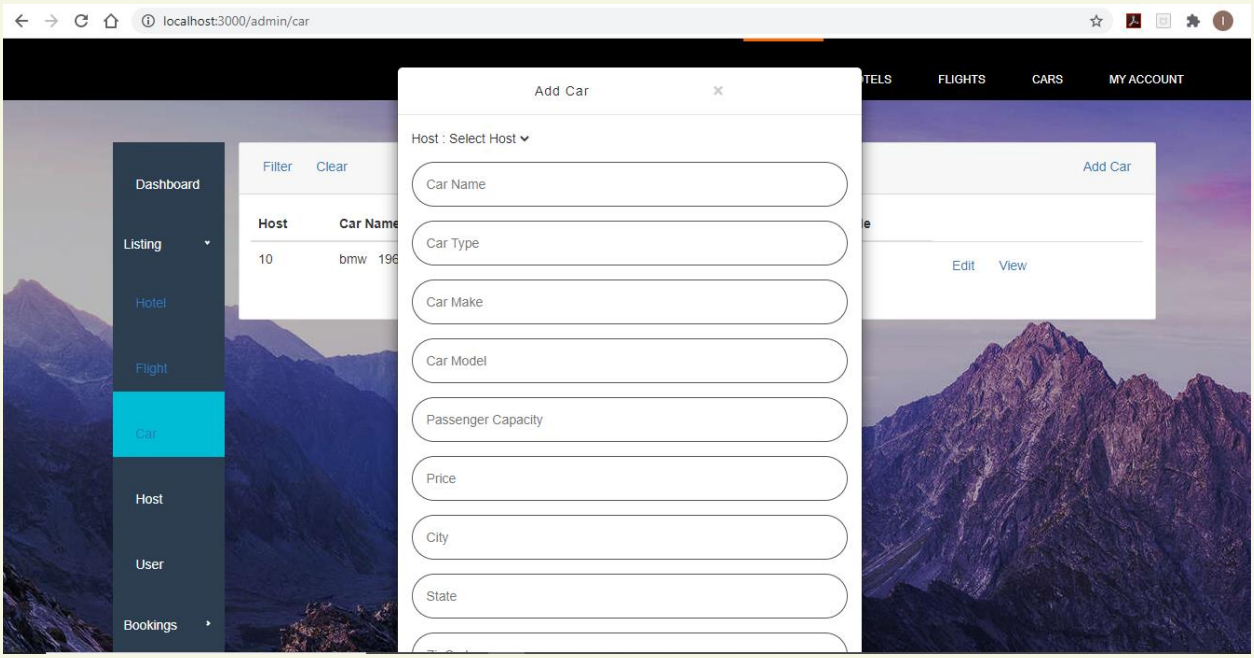
Function	Update flight
Parameter	Id –number of seats
Description	Update flight data and save to the database

Function	Delete flight
Parameter	Id
Description	Delete flight data from the database

Function	View all flight
Parameter	None
Description	Retrieve all flight data from database

Car functionality

Function	Add car
Parameter	Number of seats
Description	Save car data to the database



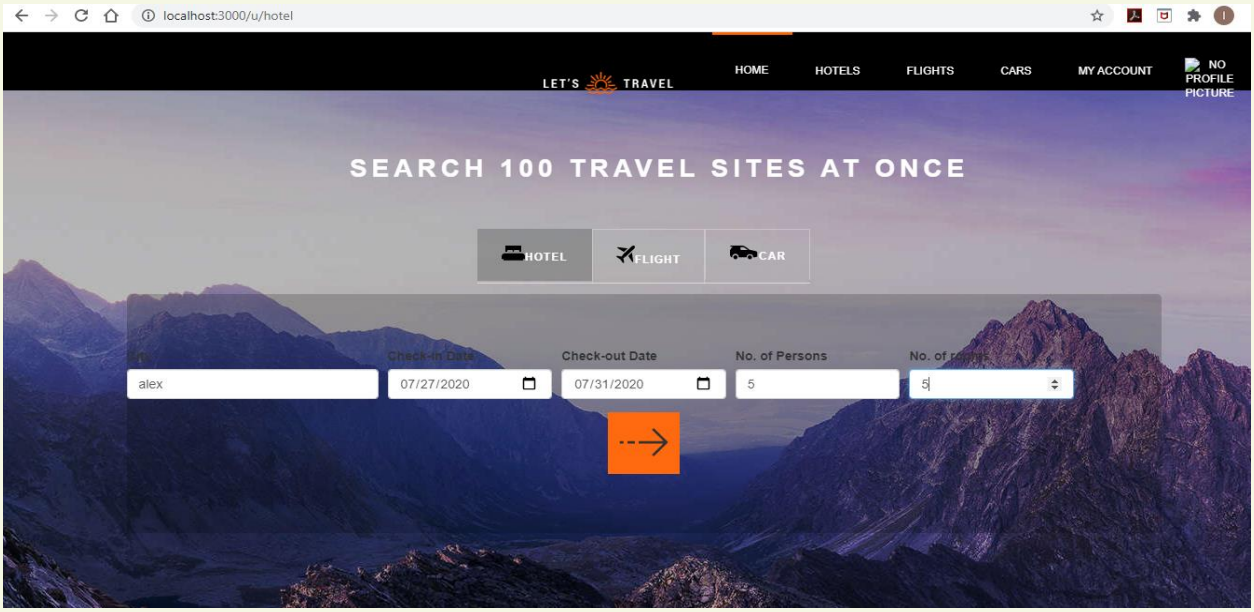
Function	Update car
Parameter	Id –number of seats
Description	Update car data and save to the database

Function	Delete car
Parameter	Id
Description	Delete car data from the database

Function	View all car
Parameter	None
Description	Retrieve all car data from database

Hotel order functionality

Function	Add hotel order
Parameter	Hotel-user- number of rooms –number of days –price
Description	Save hotel order data to the database



Function	Delete hotel order
Parameter	Id
Description	Delete hotel order data from the database

Function	View all hotel order
Parameter	None
Description	Retrieve all hotel order data from database

2.4.5 Nonfunctional requirement

- Basically, Nonfunctional requirement describe how the system work, while functional requirement describe what the system should do

- This does not mean the latter are more important, but most requirement gathering techniques focus on functional requirements, so large gaps in non-functional requirements are common.

- So, what exactly are we looking for here? Well, here are four examples of non- functional requirement group; Usability, Reliability, performance, and supportability, as well as a few top tips on each one.

Usability

- Prioritize the important functions of the system based on usage patterns.

- Frequently used functions should be tested for usability, as should complex and critical functions. be sure to create a requirement for this.

Reliability

- User have to trust the system, even after using it for a long time.
- Your goal should be along MTBF (mean time between failures).
- Create a requirement that data created in the system will be retained for a number of years without the data being changed by the system.
- It’s a good idea to also include requirements that make it easier to monitor system performance.

Performance

- What should system response time be, as measured from any point, under what circumstances?
- Are there specific peak times when the load on the system will be unusually right?
- Think of stress periods, for example, at the end of the month or in conjunction with payroll disbursement.

Supportability

- The system needs to be cost-effective to maintain
- Maintainability requirements may cover diverse levels of documentation, such as system documentation, as well as test documentation, which test cases and test plans will accompany the system.

2.5 Advantages of the new system

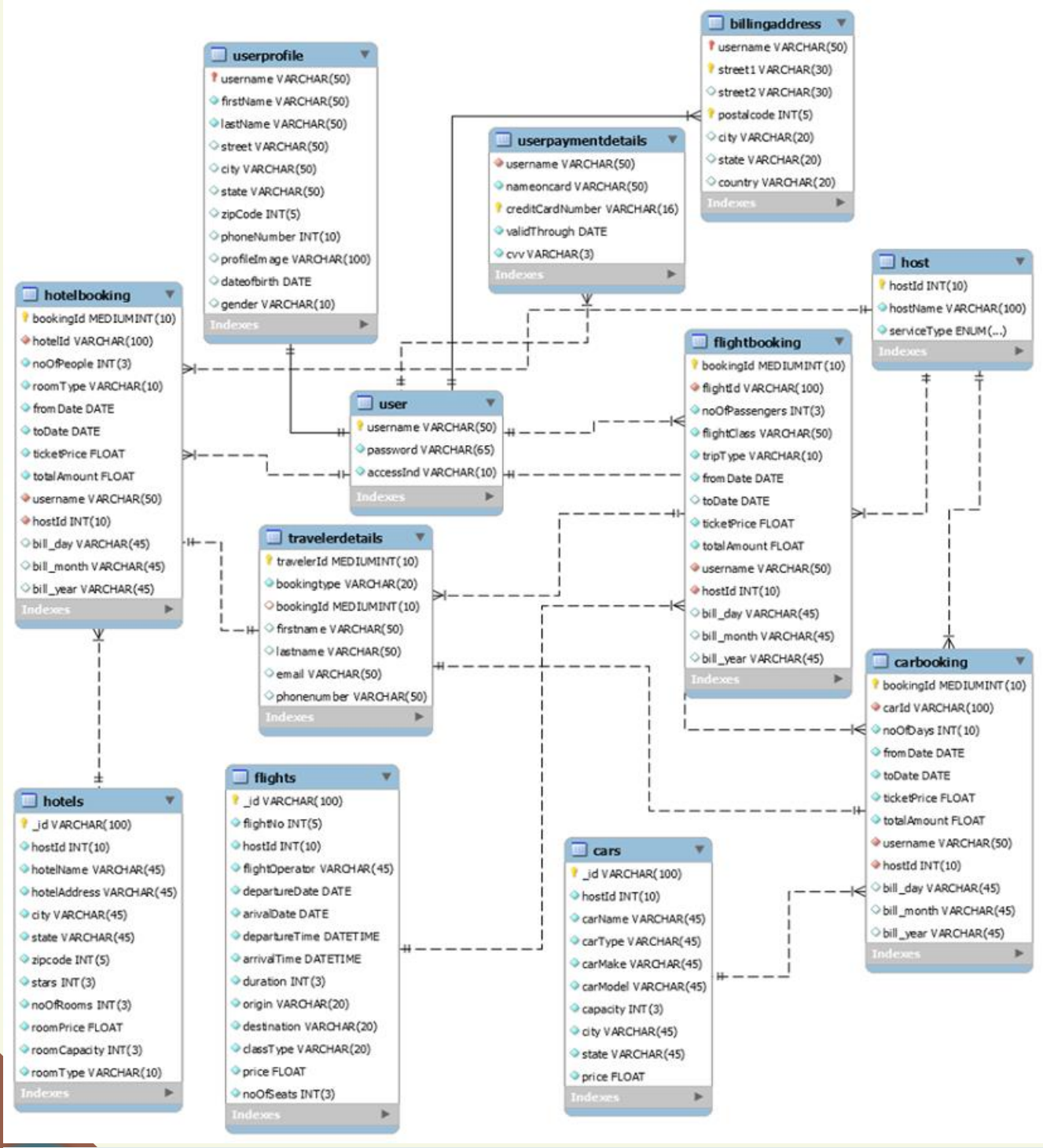
- System will be more fast
- System will use latest technology
- Good performance
- Redesign UI and UX
- System will be easier to use

2.6 Risk and Risk Management

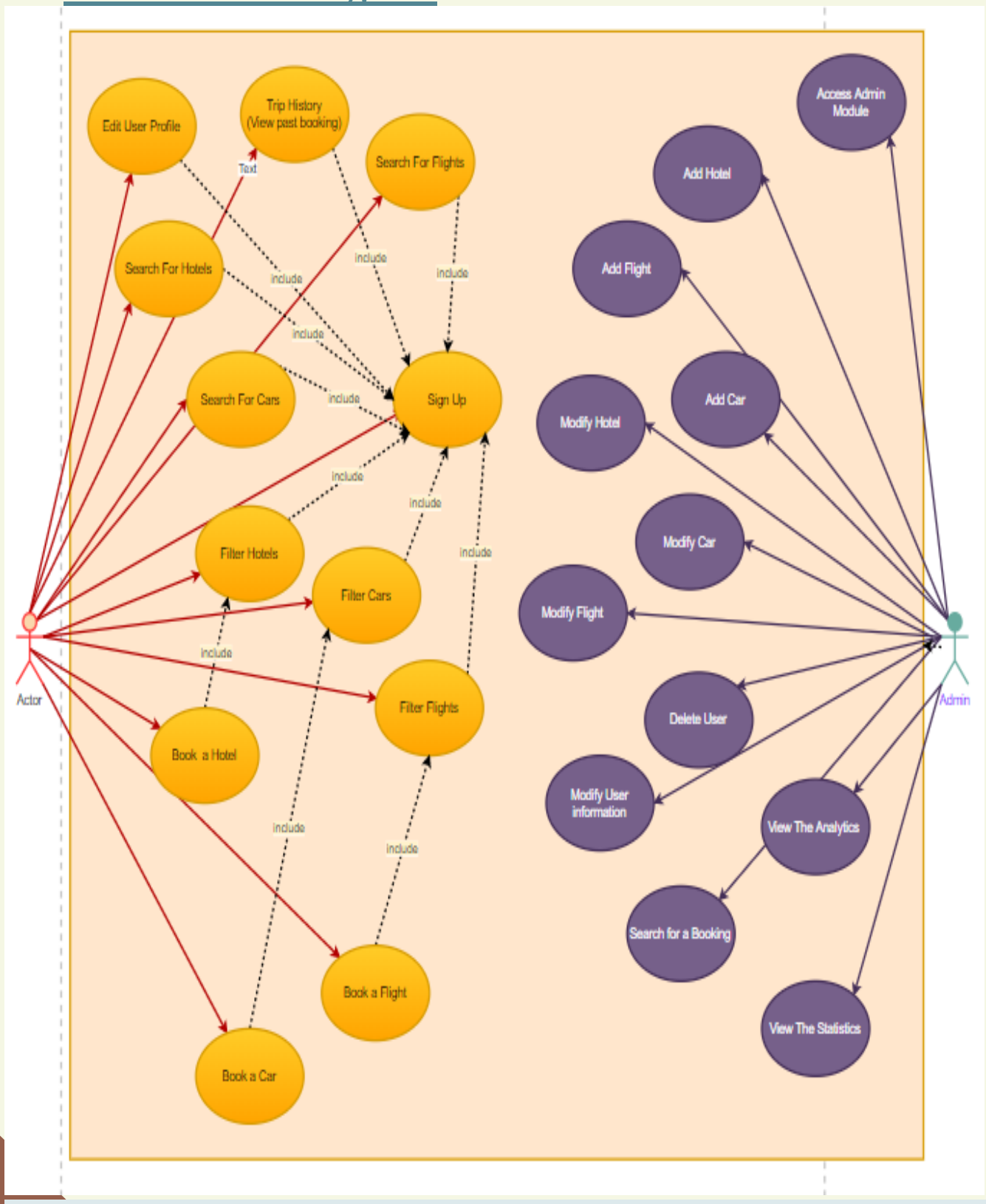
- Inventory- Price Risk: As the raw material for jewelry .i.e.. gold is imported and there are
- Restrictions imposed by RBI on its imports, therefore, the company has to maintain a high level of
- Inventory requiring high working capital. However, maintaining this inventory becomes difficult
- For the company during the slack season, as it carries inventory price risk. Also, with the volatility
- In gold prices and currency movement, hedging assumes a very important role to safeguard the
- Working of the company with respect to the gold inventory
- Competition Risk: With the increase in disposable income and the change in standard of living
- The demand for luxury goods such as consumer electronics, mobile phones, leather goods
- Automobiles, gadgets etc. are also increasing. The jewelry sector may experience company
- Mechanism. Through this approach, the company strives to identify opportunities that enhance
- Organizational values while managing or mitigating risks that can affect its future performance

3 Chapter 3: Software Design

3.1 Design of database (ERD or Class) Diagram

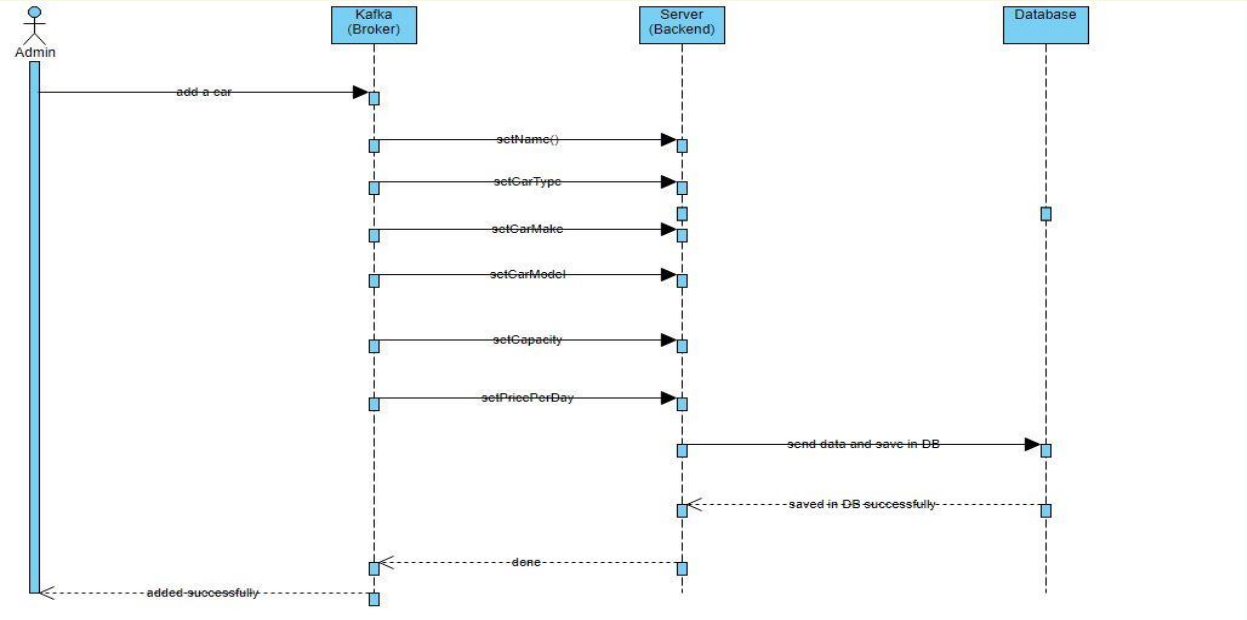


3.2 Use Case diagram

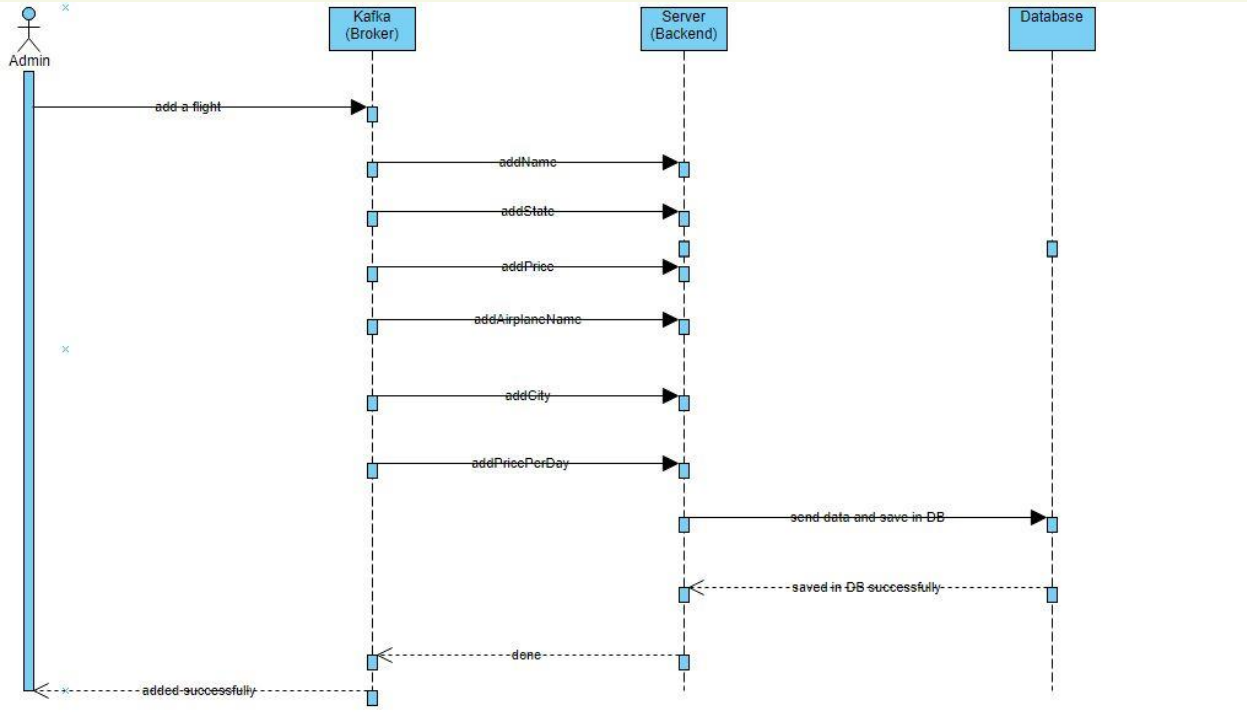


3.3 Sequence Diagram:

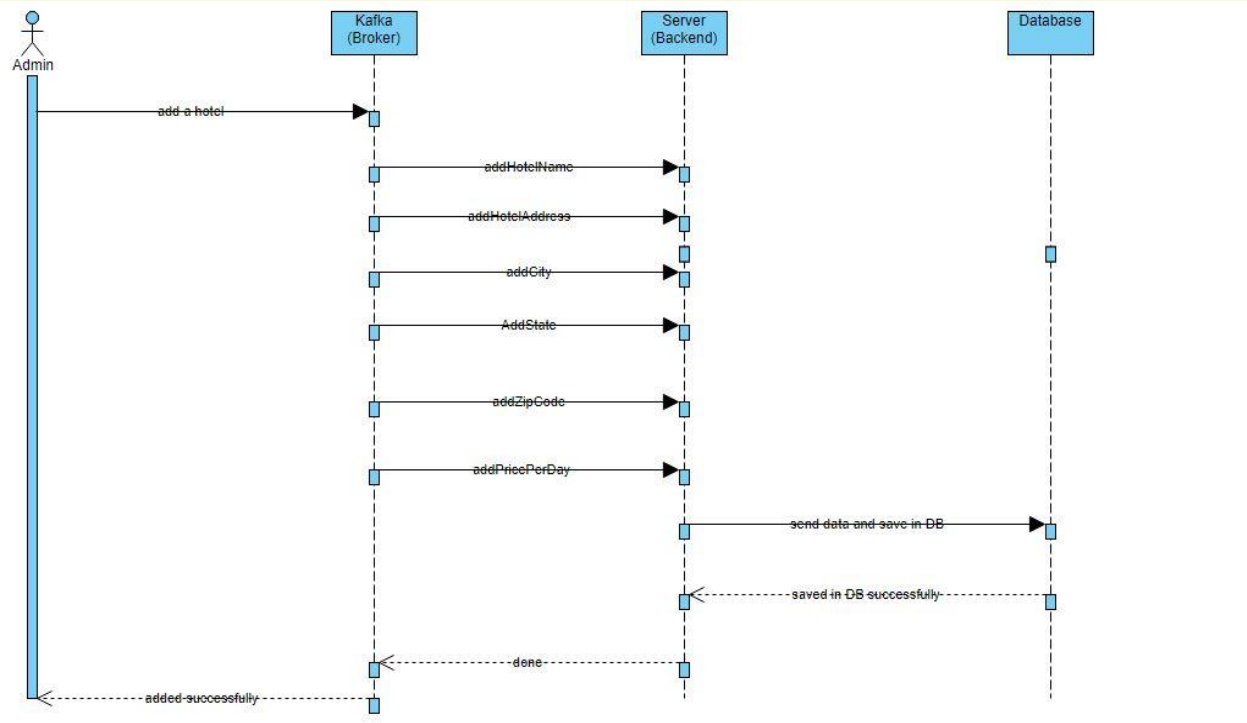
- Admin:
  1- Add a car:



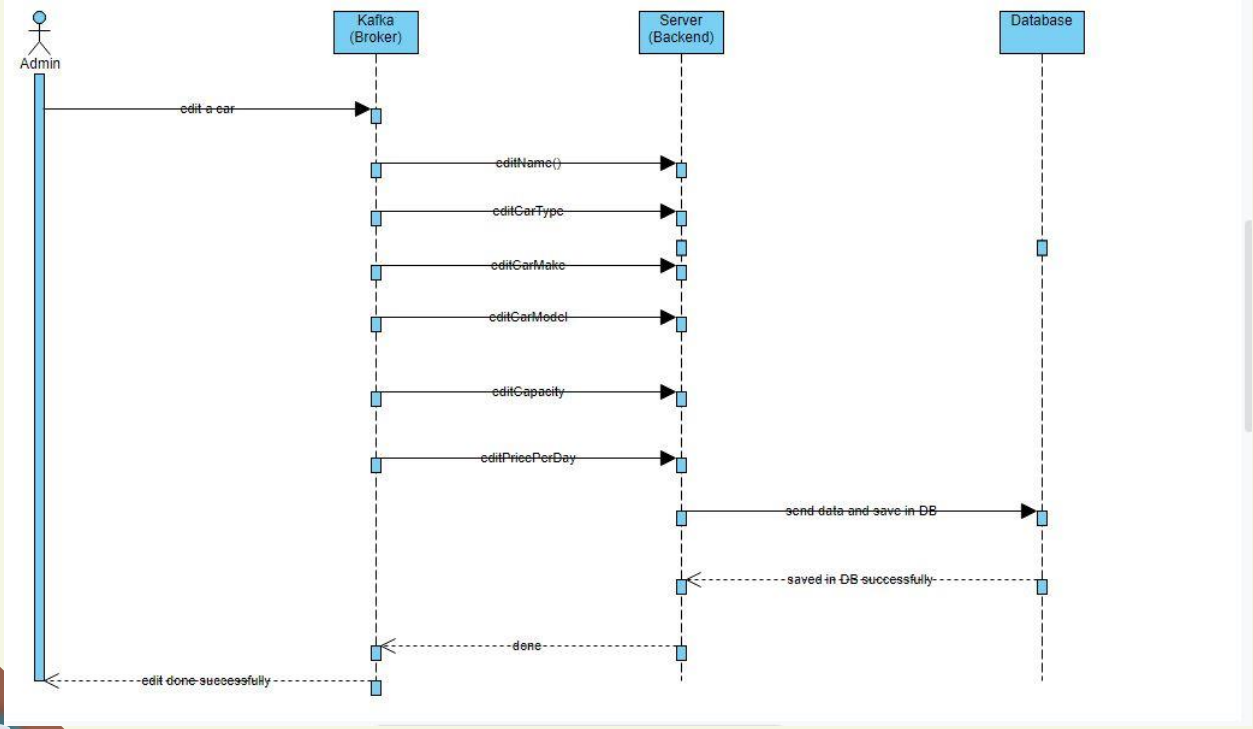
2- Add a flight:



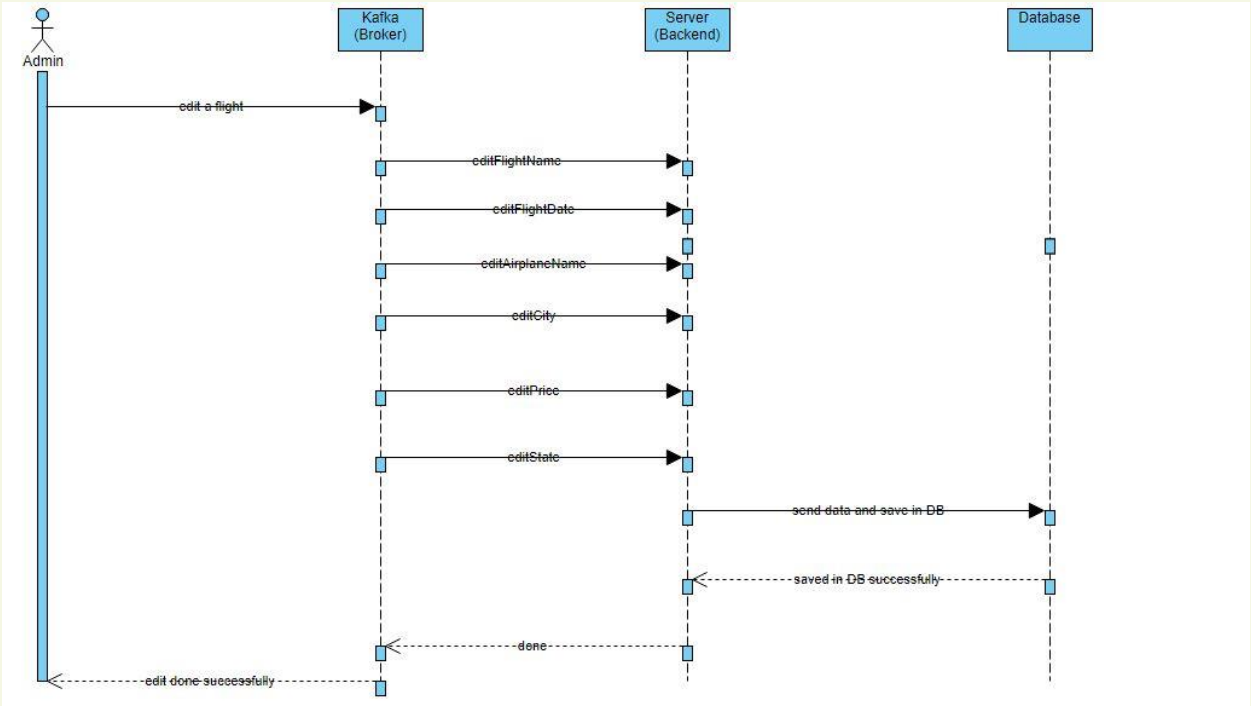
3- Add a Hotel:



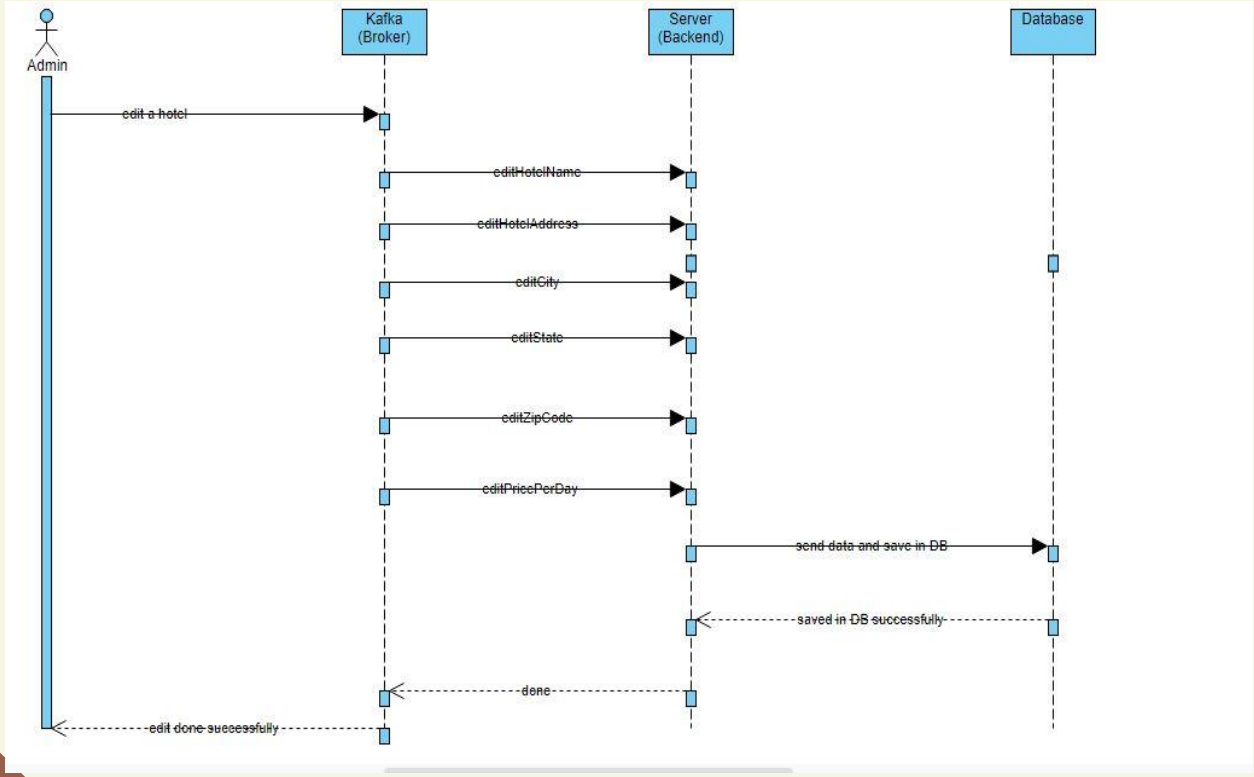
4- Edit a car:



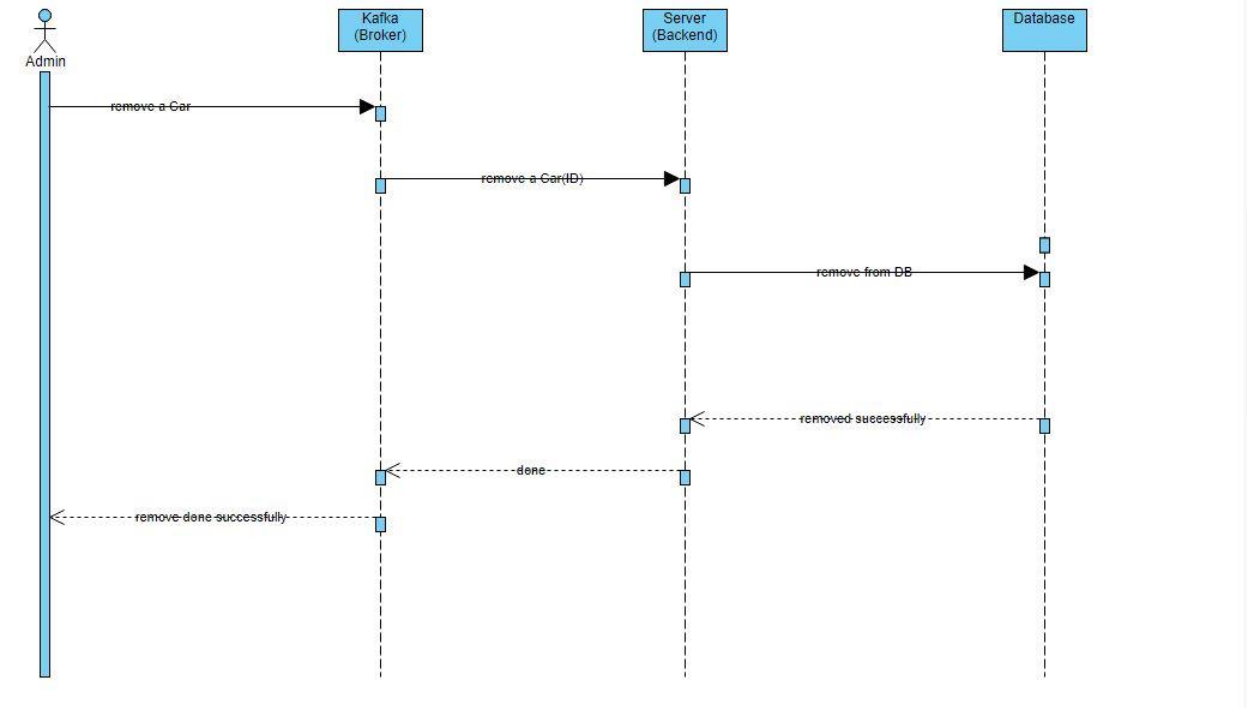
5- Edit a flight:



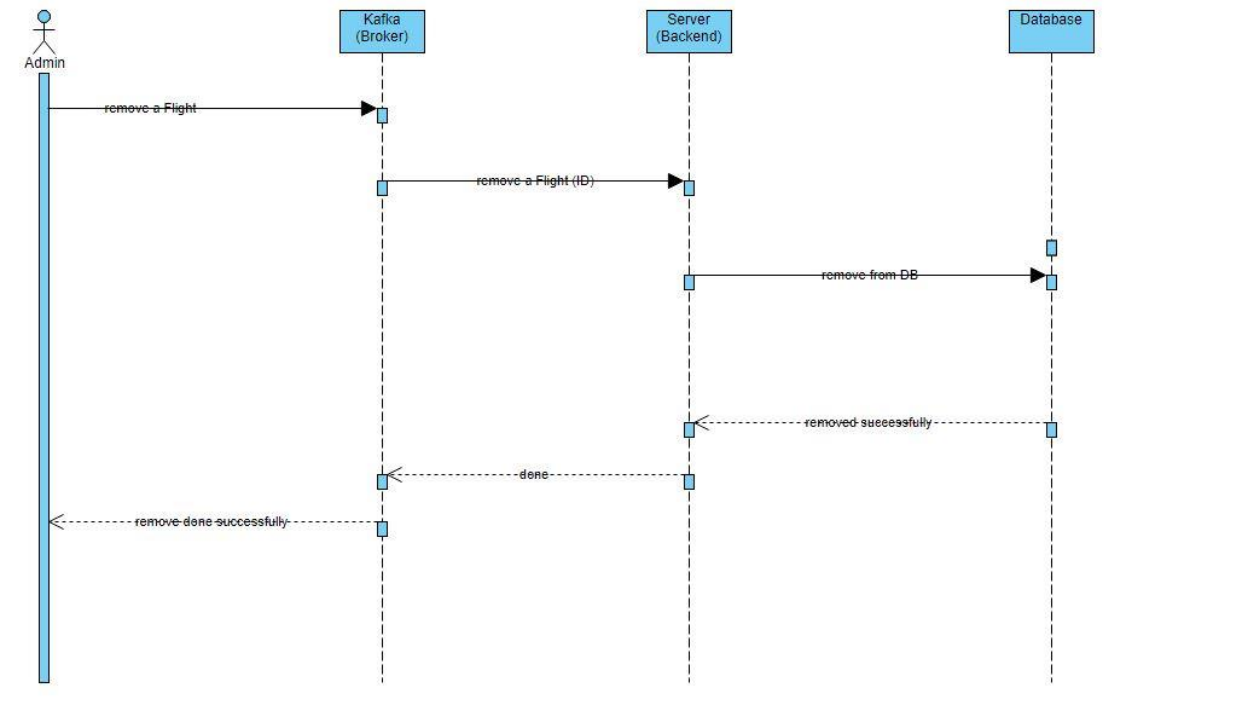
6- Edit a hotel:



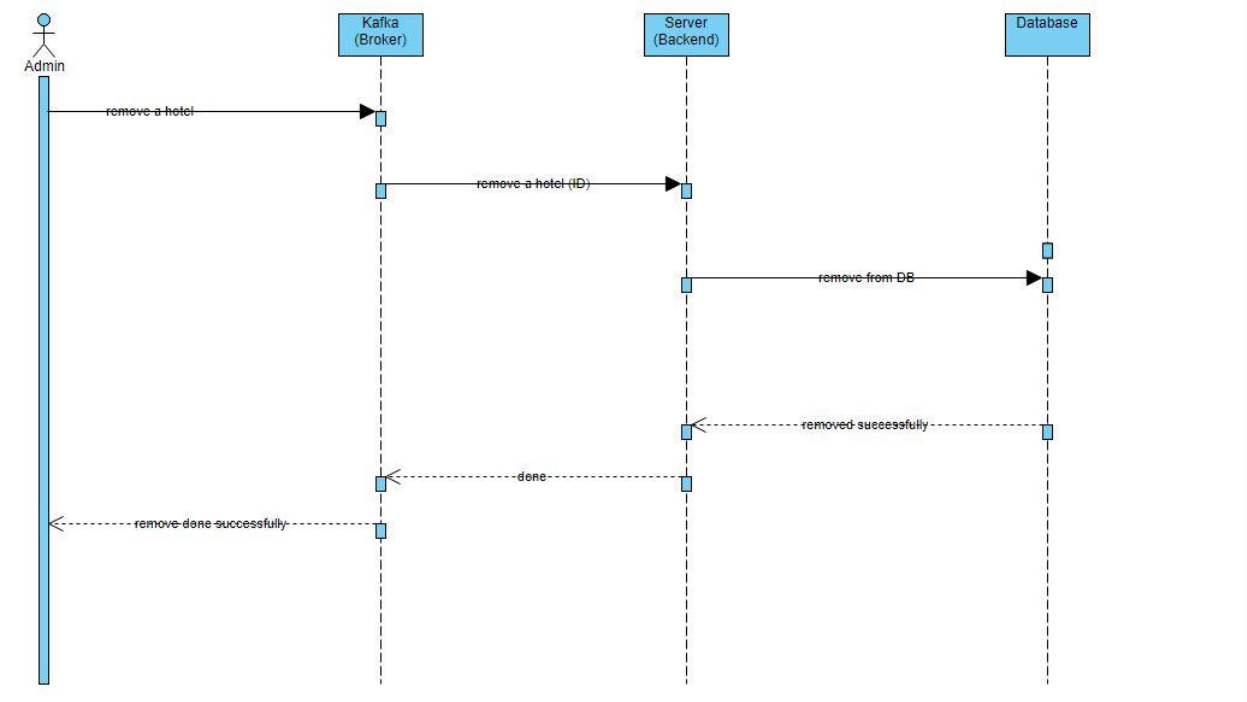
7- Remove a car:



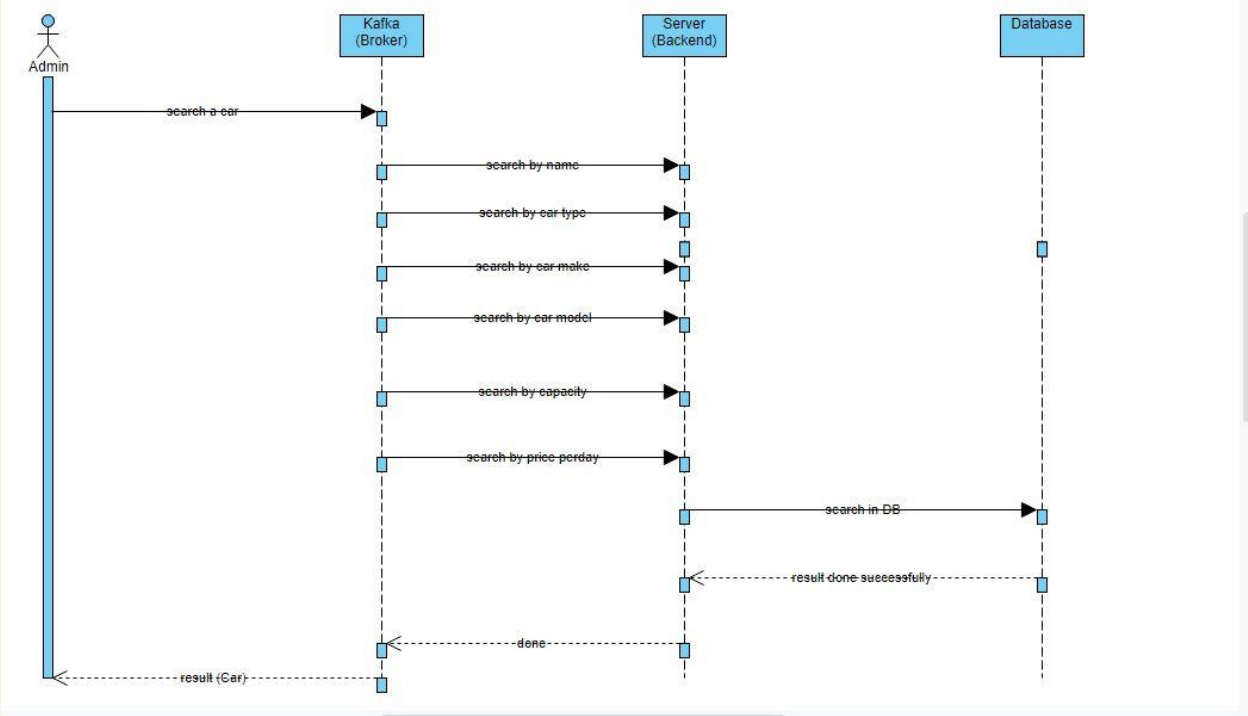
8- Remove a flight:



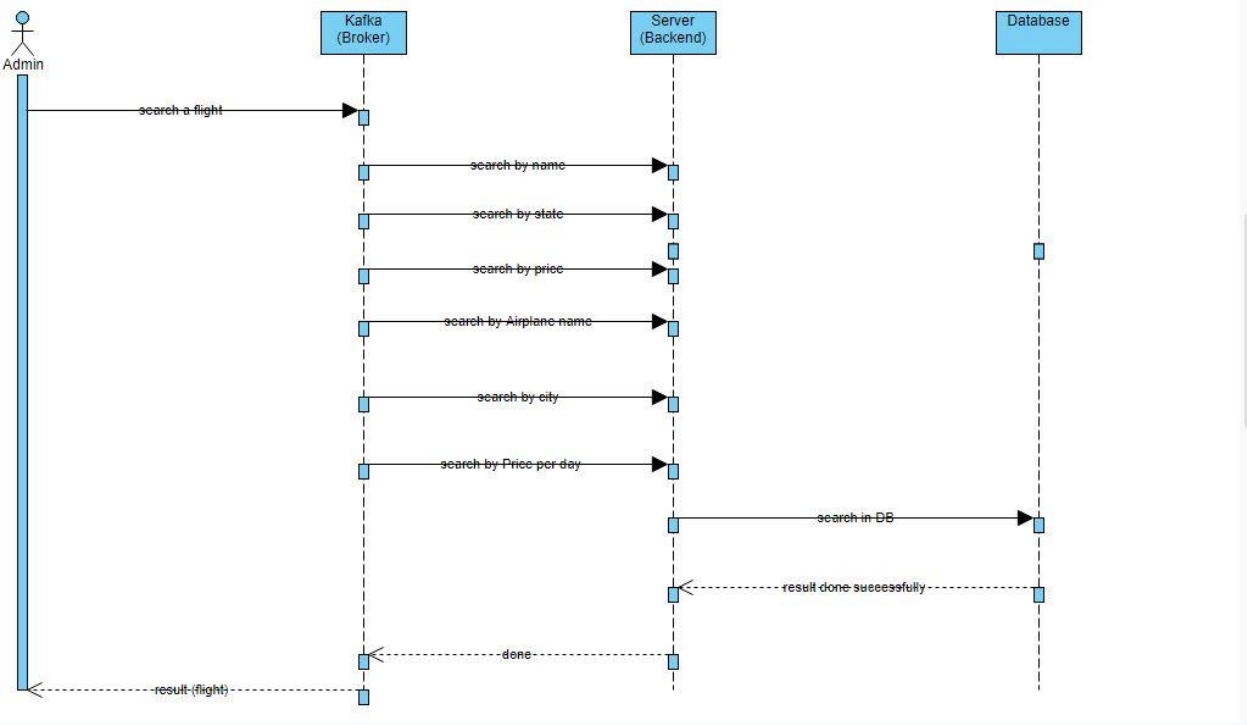
9- Remove a hotel:



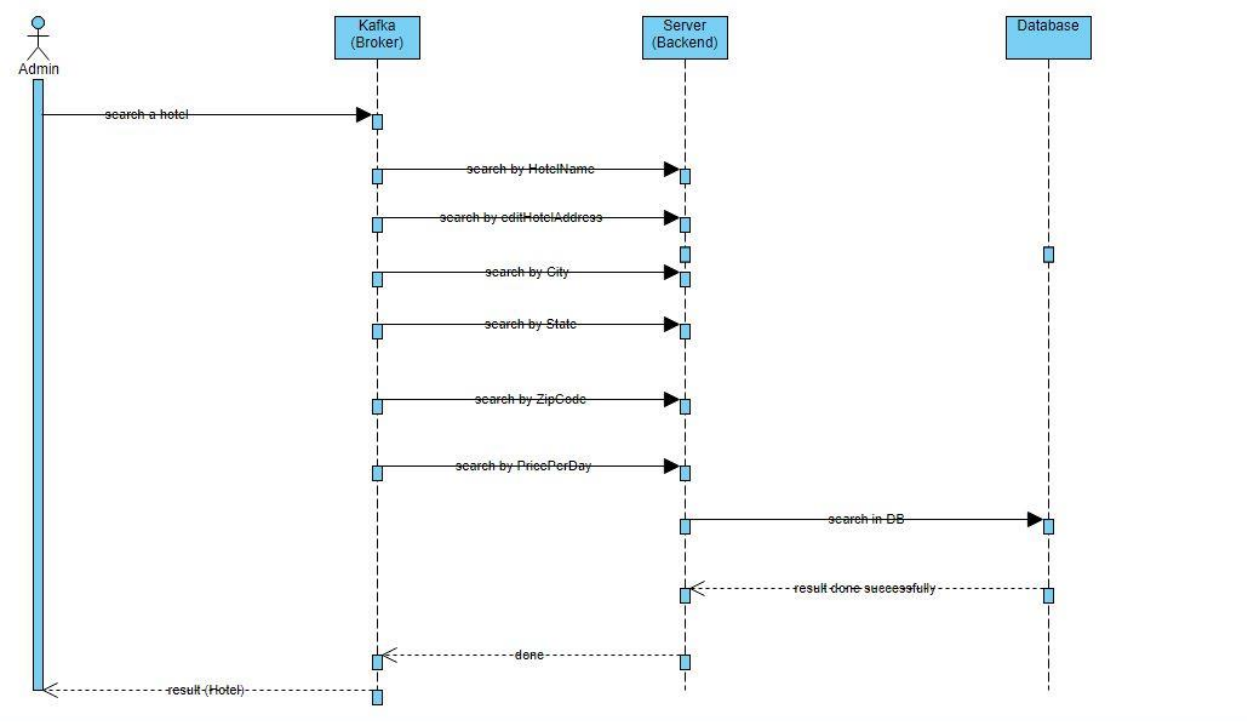
10- Search a car:



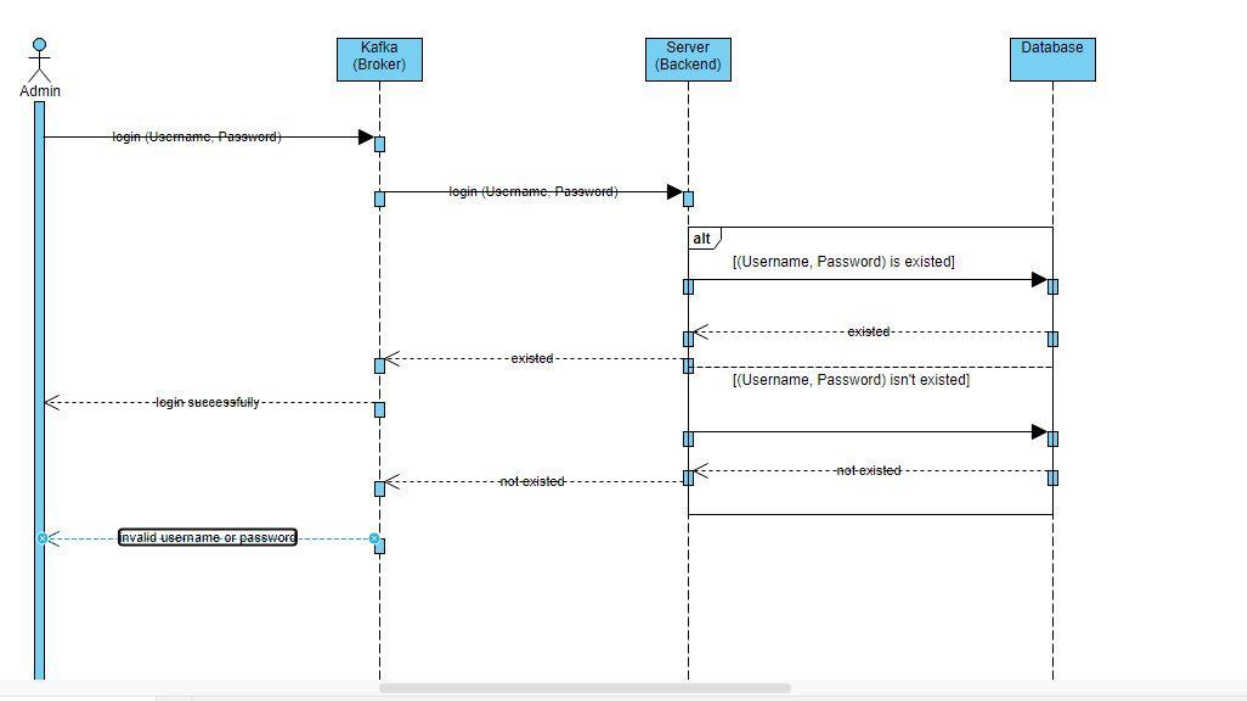
11-search a flight:



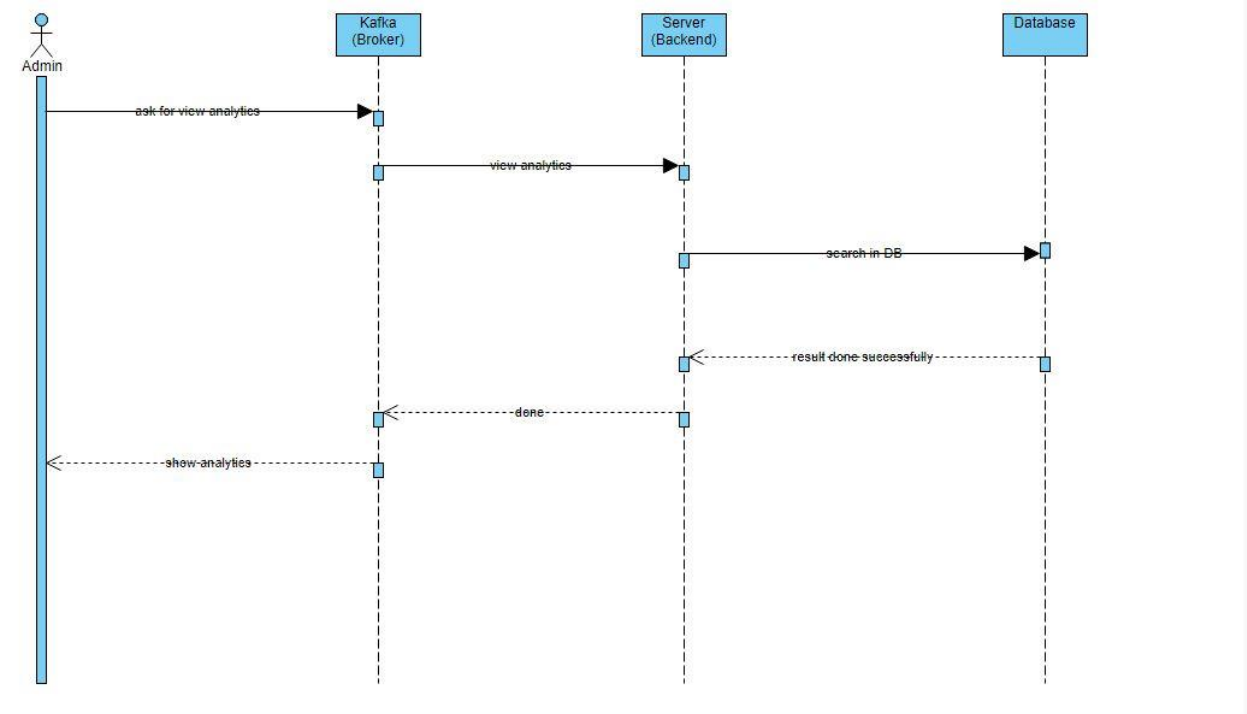
12- search a hotel:



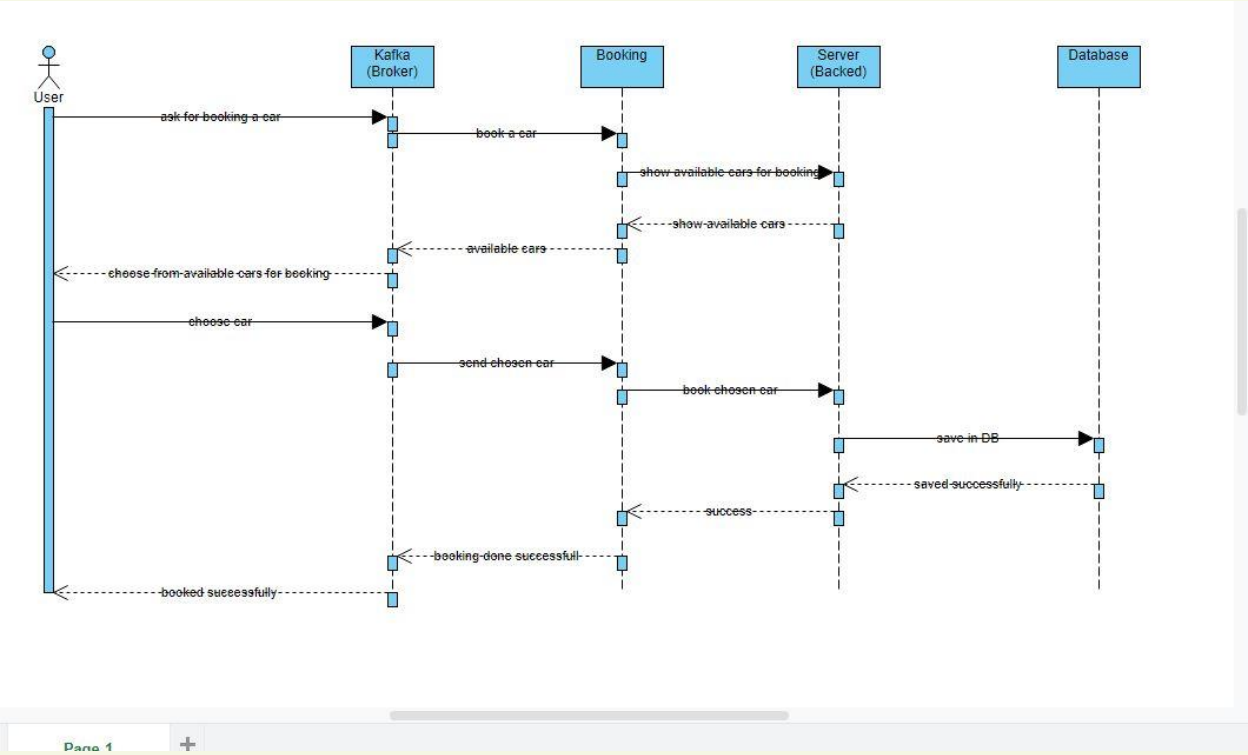
13- login:



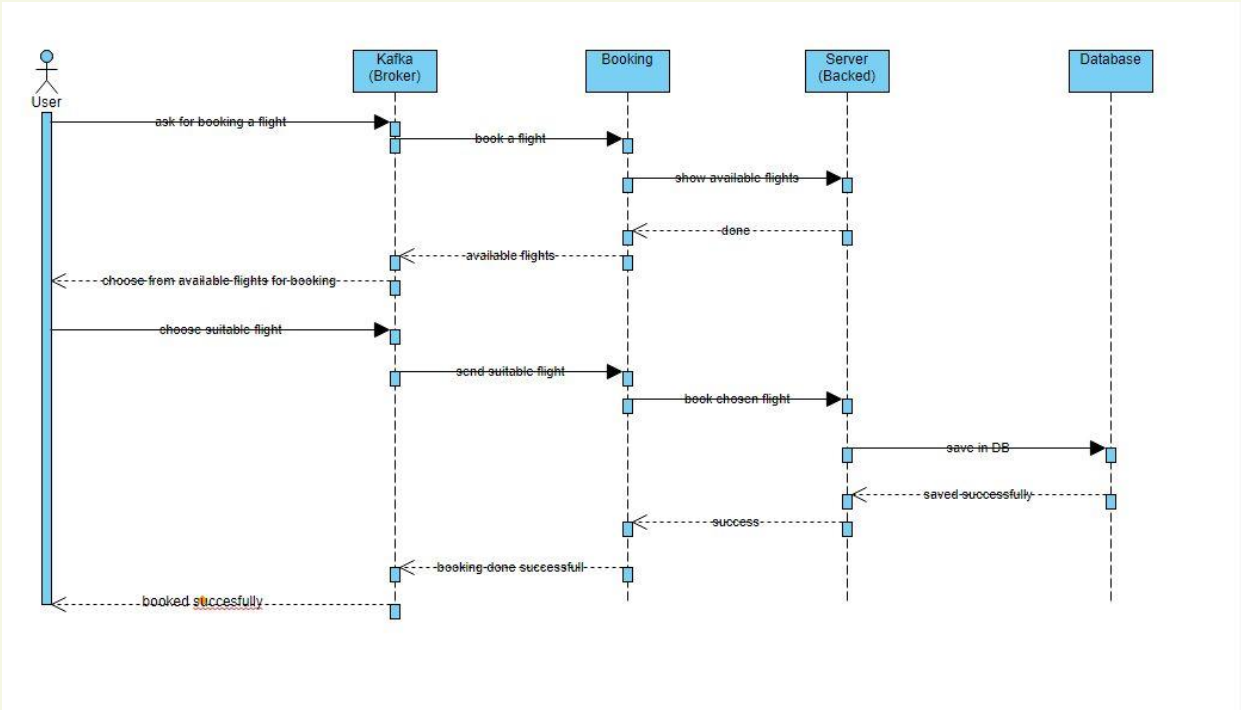
14- view analytics:



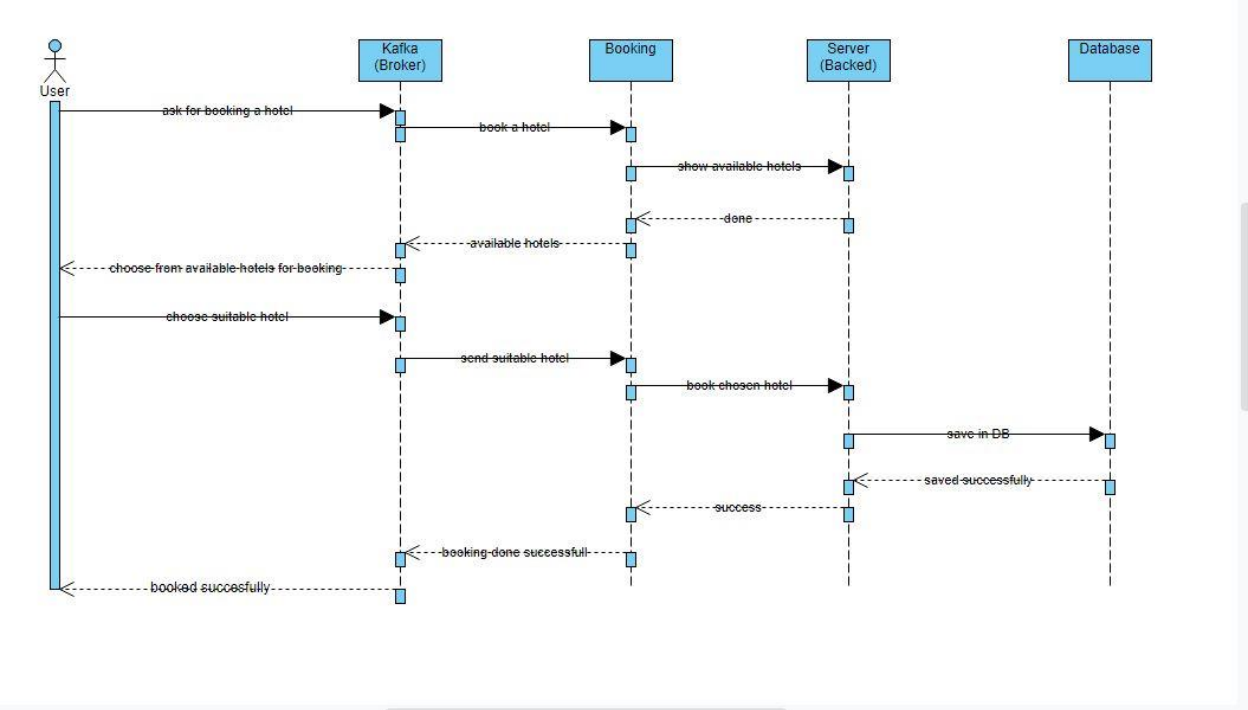
- User:
  1- Book a car:



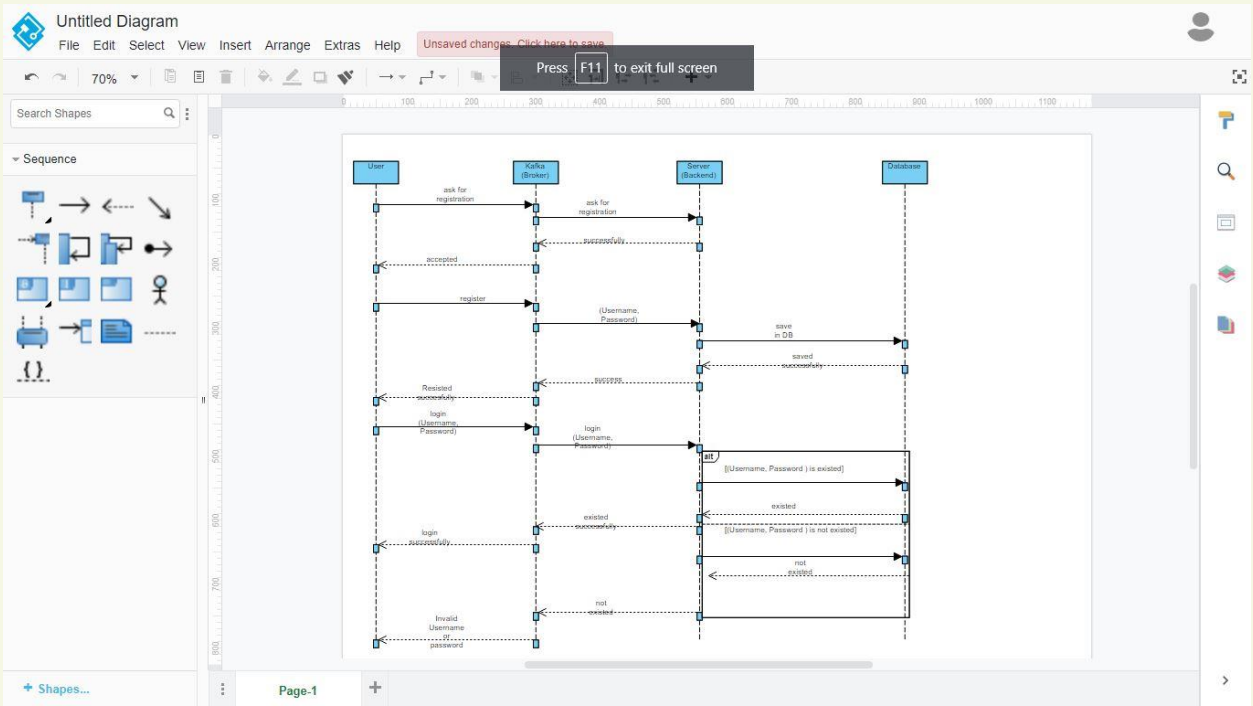
  2- Book a flight:



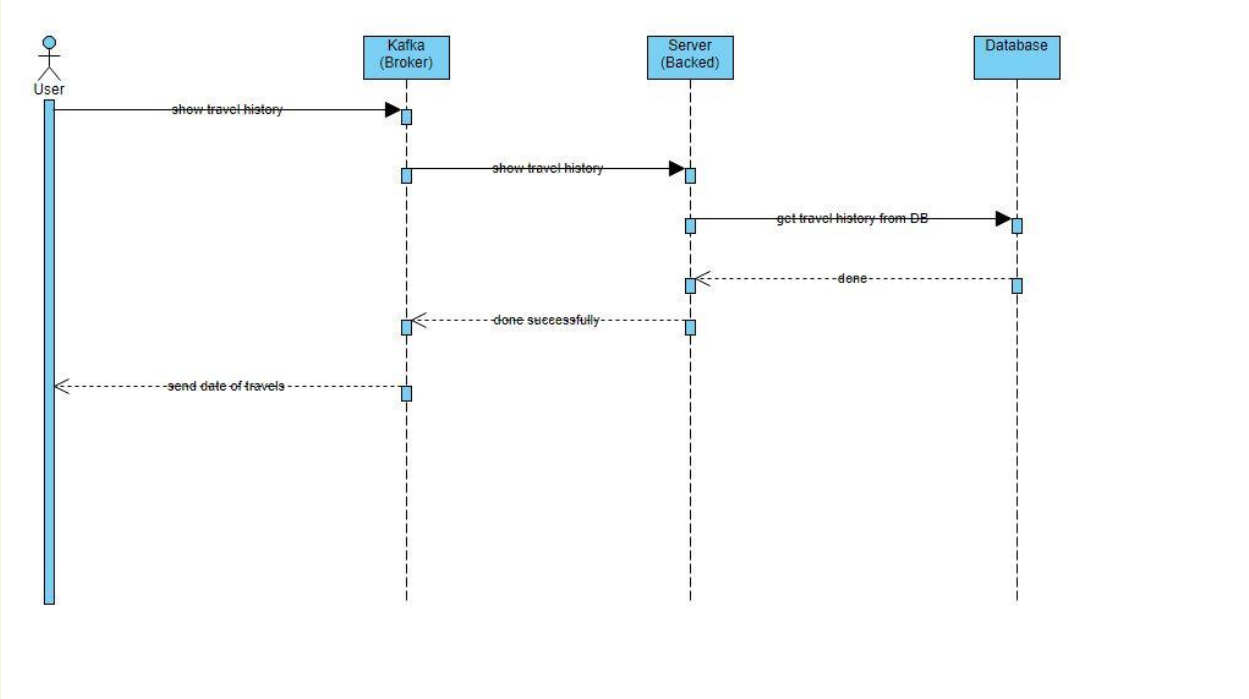
3- Book a hotel:



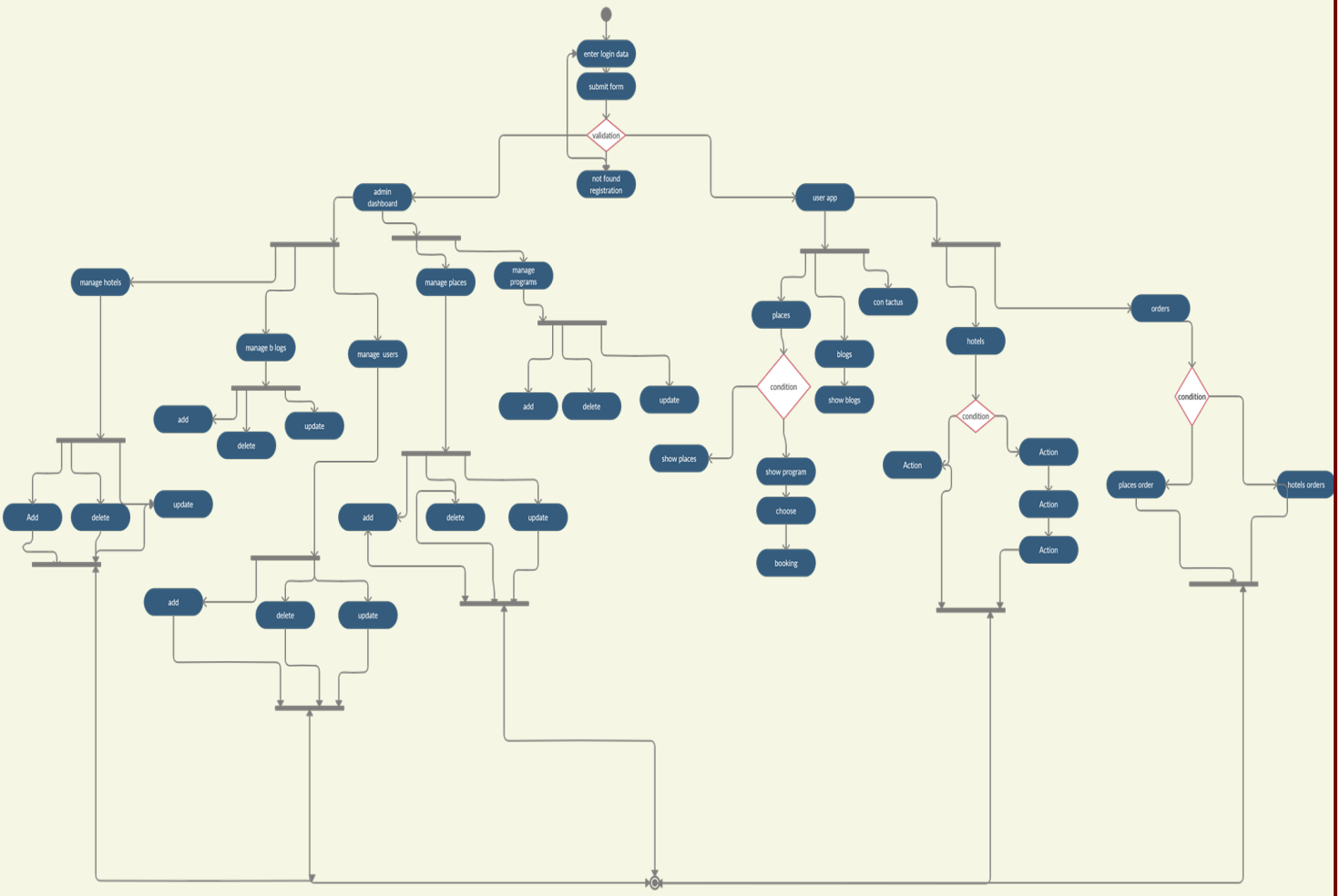
4- Login and register:



5- Show travel history:



3.4 activity diagram :

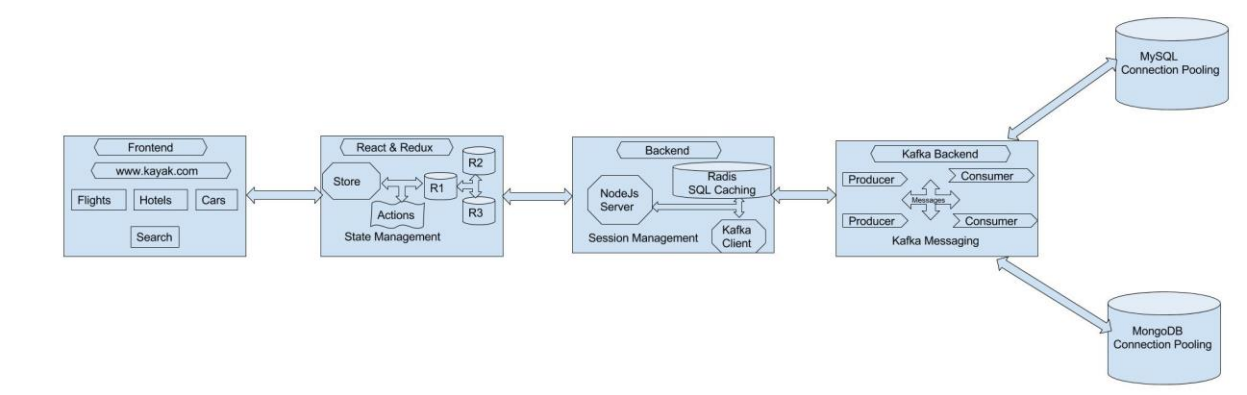


Chapter 4: Implementation:

Domain	Technology
Front End	React JS
Back End	Node JS
UI/UX	Twitter Bootstrap & Material UI
Database	MongoDB & MySQL
Authentication	Passport JS
Middleware	Kafka
SQL Caching	Redis
Visualization Graphs	Recharts
Load Testing	JMeter
API Testing	Mocha

- User send request to server (Register)
- Server check mail of user is existing or not exists
- If exists server send response to user that mail is exists
- If not, server send response to server you can login
- User send request to server(login)
- Server check if email and password are existing
- Send response to users that can access system
- User send request to view hotel, cars, flights
- System send response to user (retrieve all of this)
- User send request to book a hotel, car, flight
- System and response to user that booking is successfully or not

➢System Design



Chapter 5: Testing:

5.1 Unit Testing

- Unit testing is the level of software testing where individual units/components of software are tested. The purpose is to validate that each unit of the software performs as designed. A unit is smallest testable part of any software. It usually has one or a few inputs and usually a single output. In procedure programming a unit may be an individual program, function, procedure, etc. In object- oriented programming, the smallest unit is a method, which may belong to base/super class, abstract or derived/child class. (Some treat a module of an application as a unit. this is to be discouraged as there will probably be many individual units within that module). unit testing frameworks, drivers, stubs, and mock/fake objects are used to assist in unit testing.

- Unit testing increases confidence in changing/ maintaining code. If they are run every time any code is changed, we will be able to promptly catch any defects introduced due to the change. Also, if codes are already made less interdependent to make unit testing possible, the unintended impact of changes to any codes is less
- Codes are more reusable. In order to make unit testing possible, codes need to be modular. This means that codes are easier to reuse.

- Development is faster .How? if you do not have unit testing in place ,you write your code and perform that fuzzy' developer test'(you set some breakpoints, fire up the GUI ,provide a few inputs that hopefully hit your code and hope that you are all set) But ,if you have unit testing in place ,you write the test ,write the code and run the test. Writing tests takes time is compensated by the less amount of time it takes to run the tests; you need not fire up the GUI and provide all those inputs. And, of course, units' tests are more reliable than 'developer tests. Development is faster in the long run too. How? The effort required to find and fix defects found during unit testing is very less in comparison to the effort required to fix defects
- During system testing or acceptance testing.
- The cost of fixing a defect found during unit testing is lesser in comparison to that of defects detected at higher levels. Compare the cost (time, effort, destruction, humiliation) of a defect detected during acceptance testing or when the
- software is live.

Unit Testing Benefits

- Debugging is easy. When a test fails, only the latest changes need to be debugged. With testing at higher levels, changes made over the span of several days/weeks/months need to be scanned.

Unit Testing Tips

- Find a tool/framework for your language.

- Do not create test cases for everything. Instead, focus on the tests that impact the behavior of the system.
- Isolate the development environment from the test environment.
- Use test data that is close to that of production.
- Before fixing a defect, write a test that exposes the defect. Why? First, you will later be able to catch the defect if you do not fix it properly. Second, your test suite is now more comprehensive. Third, you will most probably be too lazy to write the test after you have already fixed the defect.
- Write test cases that are independent of each other. For example, if a class depends on a database, do not write a case that interacts with the database to test the class. Instead, create an abstract interface around that database connection and implement that interface with a mock object.
- Aim at covering all paths through the unit. Pay particular attention to loop conditions.
- Make sure you are using a version control system to keep track of your test scripts.
- In addition to writing cases to verify the behavior, write cases to ensure the performance of the code.
- Perform unit tests continuously and frequently.

Unit Testing Techniques

Code coverage techniques used in united testing are listed below:

- Statement Coverage
- Decision Coverage
- Branch Coverage
- Condition Coverage
- Finite State Machine Coverage

Unit Testing is of two types

- Manual
- Automated

Unit testing is commonly automated but may still be performed manually. Software Engineering does not favor one over the other but automation is preferred. A manual approach to unit testing may employ a step-by-step instructional document. Under the automated approach

- A developer writes a section of code in the application just to test the function. They would later comment out and finally remove the test code when the application is deployed.
- A developer could also isolate the function to test it more rigorously. This is a more thorough unit testing practice that involves copy and paste of code to its own testing environment than its natural environment. Isolating the code helps in revealing unnecessary dependencies between

the code being tested and other units or data spaces in the product. These dependencies can then be eliminated.

- A coder generally uses a UnitTest Framework to develop automated test cases. Using an automation framework, the developer codes criteria into the test to verify the correctness of the code. During execution of the test cases, the framework logs failing test cases. Many frameworks will also automatically flag and report, in summary, these failed test cases. Depending on the severity of a failure, the framework may halt subsequent testing.
- The workflow of Unit Testing is 1) Create Test Cases 2) Review/Rework 3) Baseline 4) Execute Test Cases

5.2 INTEGRATION TESTING:

is a level of software testing where individual units are combined and tested as a group? The purpose of this level of testing is to expose faults in the interaction between integrated units. Test drivers and test stubs are used to assist in Integration Testing

- **integration testing**: Testing performed to expose defects in the interfaces and in the
  interactions between integrated components or systems.
  See also component integration
  testing, system integration testing.
- **component integration testing**: Testing performed to expose defects in the interfaces and
  interaction between integrated components.
- **system integration testing**: Testing the integration of systems and packages; testing

interfaces to external organizations (e.g. Electronic Data Interchange, Internet).

Analog: During the process of manufacturing a ballpoint pen, the cap, the body, the tail and clip, the ink cartridge and the ballpoint are produced separately and unit tested separately. When two or more units are ready, they are assembled and Integration Testing is performed. For example, whether the cap fits into the body or not.
Method: Any of Black Box Testing, White Box Testing and Gray Box Testing methods can be used. Normally, the method depends on your definition of 'unit'.

- **Tasks:**
- **Integration Test Plan**
    - Prepare
    - Review
    - Rework
    - Baseline
- **Integration Test Cases/Scripts**
    - Prepare
    - Review
    - Rework
    - Baseline
    - Integration Test
    - Perform

Bottom-up Integration

In the bottom-up strategy, each module at lower levels is tested with higher modules until all modules are tested. It takes help of Drivers for testing

Advantages:

- Fault localization is easier.
- No time is wasted waiting for all modules to be developed unlike Big-bang approach

Disadvantages:

- Critical modules (at the top level of software architecture) which control the flow of application are tested last and may be prone to defects.
- An early prototype is not possible

Top-down Integration:

In Top to down approach, testing takes place from top to down following the control flow of the software system.
Takes help of stubs for testing

Advantages:

- Fault Localization is easier.
- Possibility to obtain an early prototype.
- Critical Modules are tested on priority; major design flaws could be found and fixed first.

Disadvantages:

- Needs many Stubs.
- Modules at a lower level are tested inadequately.

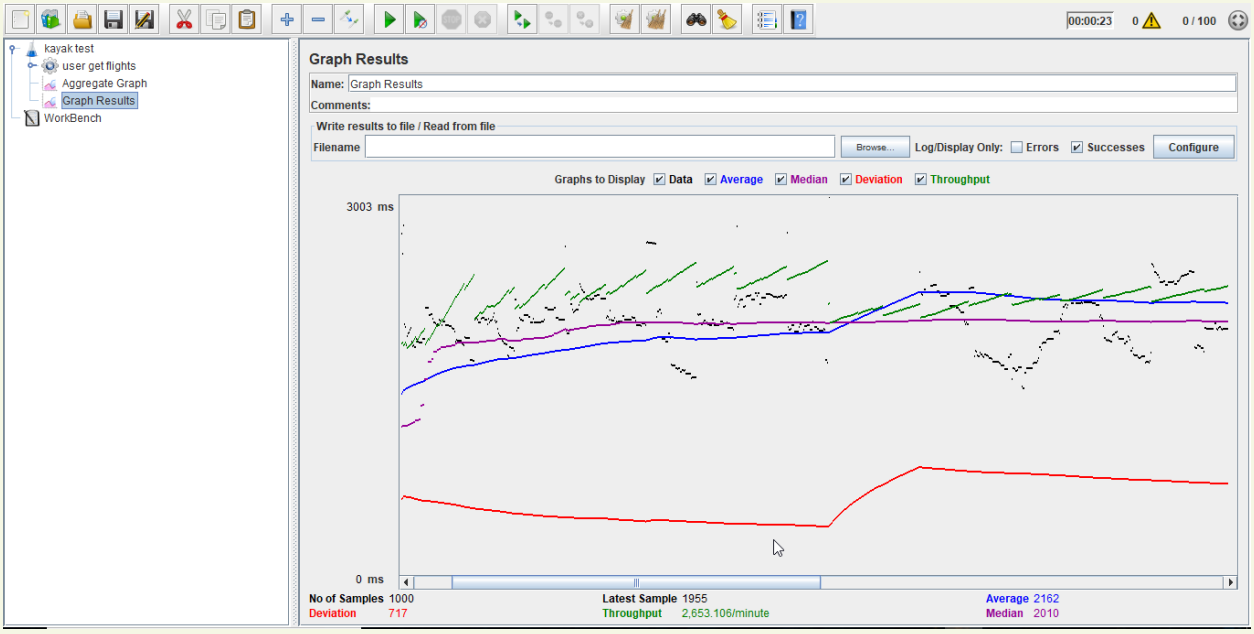
5.3 Additional testing

- Additional testing is any type of software testing that seeks to verify the interfaces between components against a software design .Software components may be additional in an iterative way or all together ("big bang").Normally the former is considered a better practice since it allows interface issues to be located more quickely and fixed

- Additional testing works to expose defects in the interfaces and interaction between additional (module).progressively larger groups of tested software components corresponding to elements of the architectural design and tested until the software works as a system

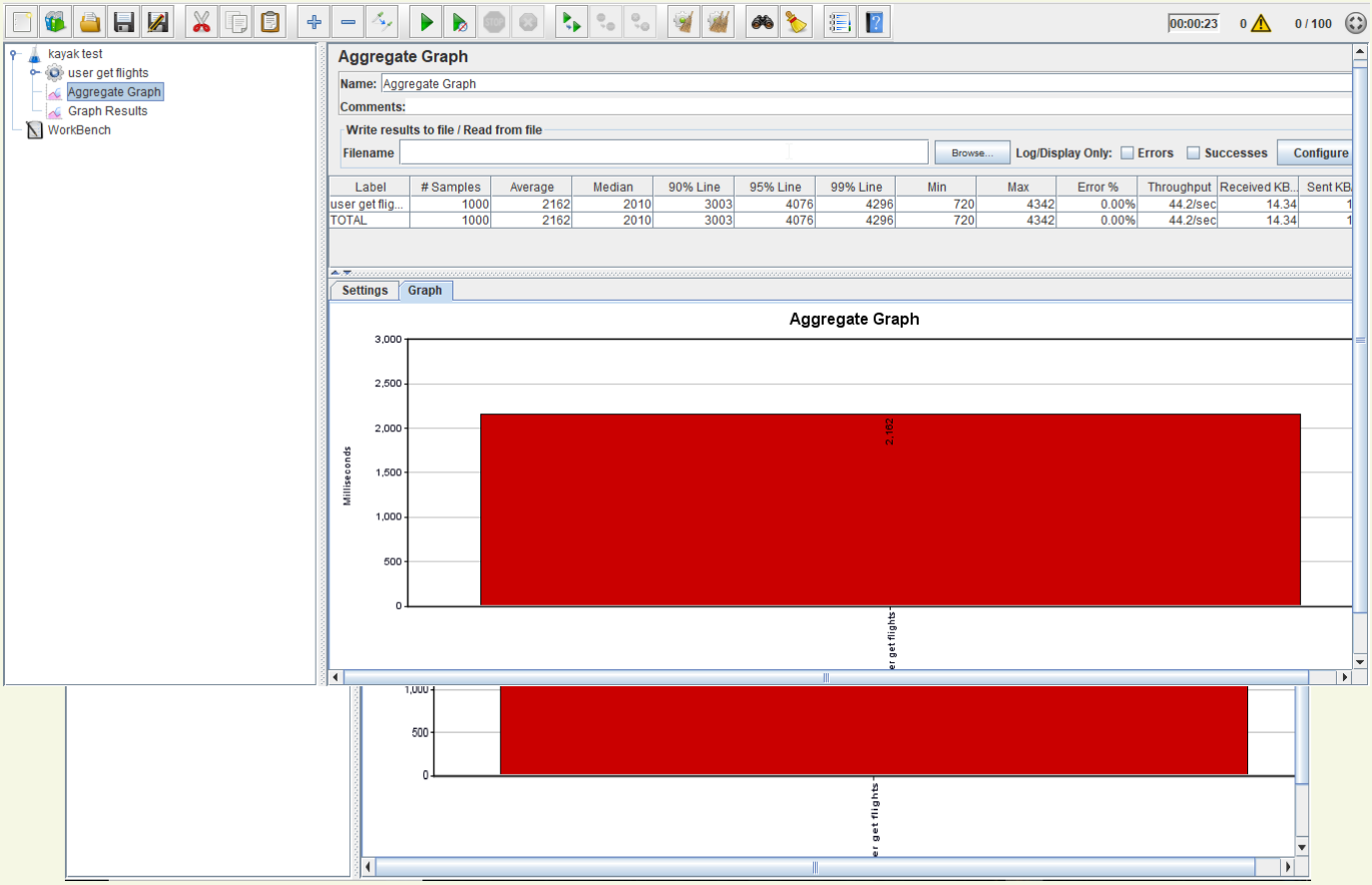
5.4 JMeter testing:

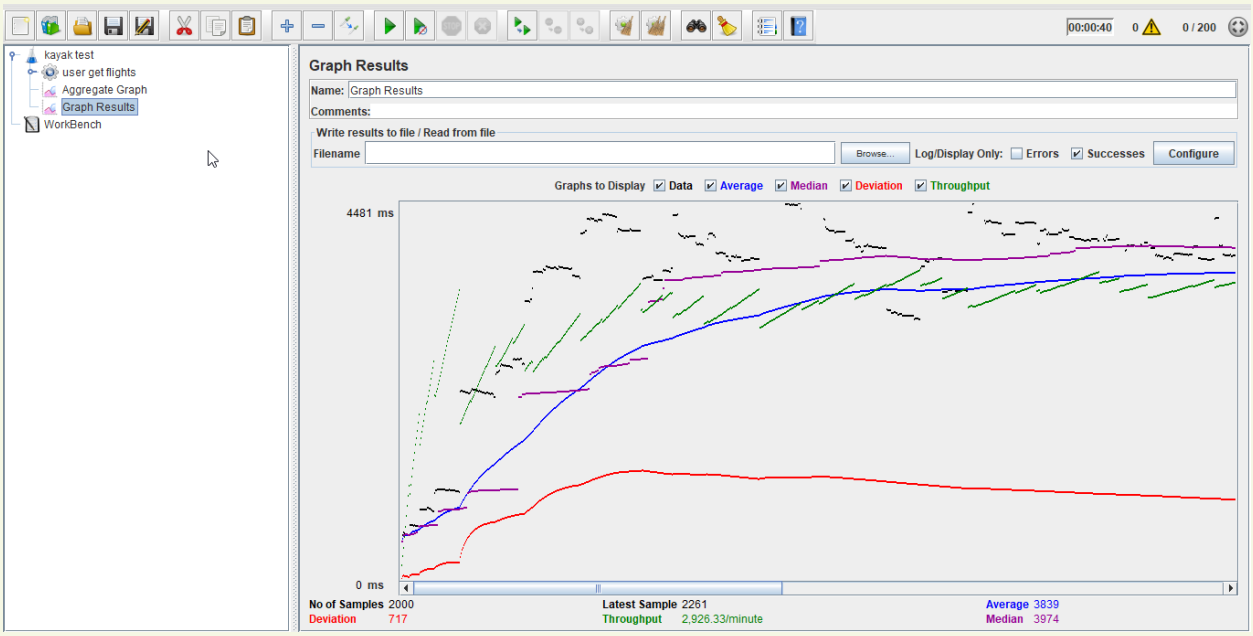
- No connection pooling, no Redis:



1- 100 concurrent users:

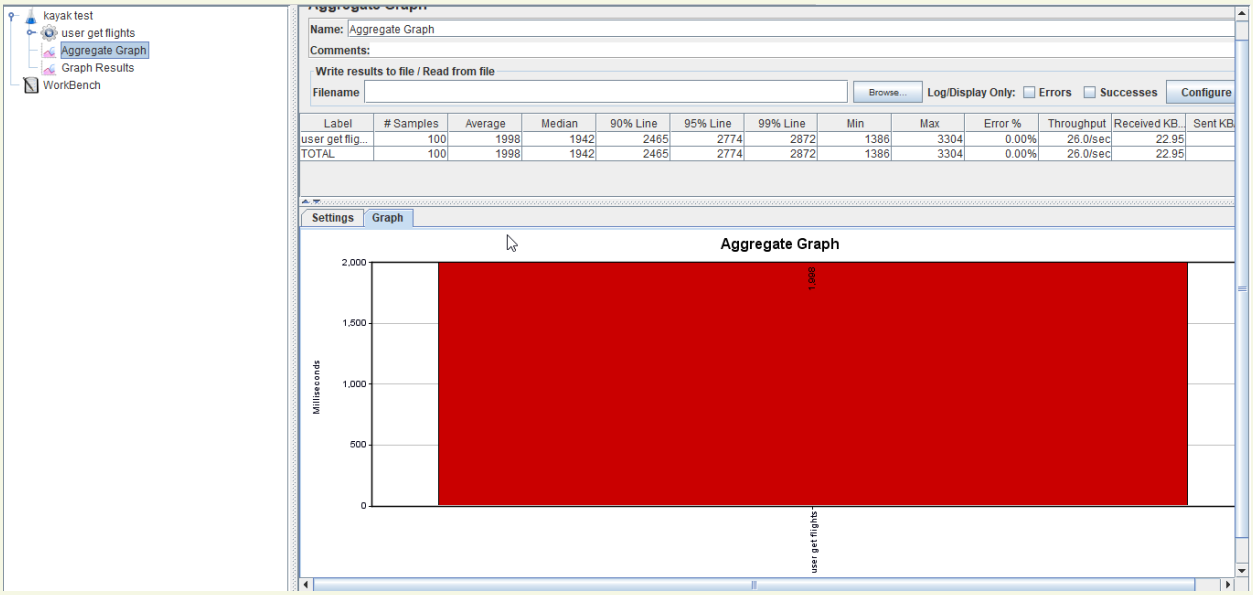
2- 200 users:

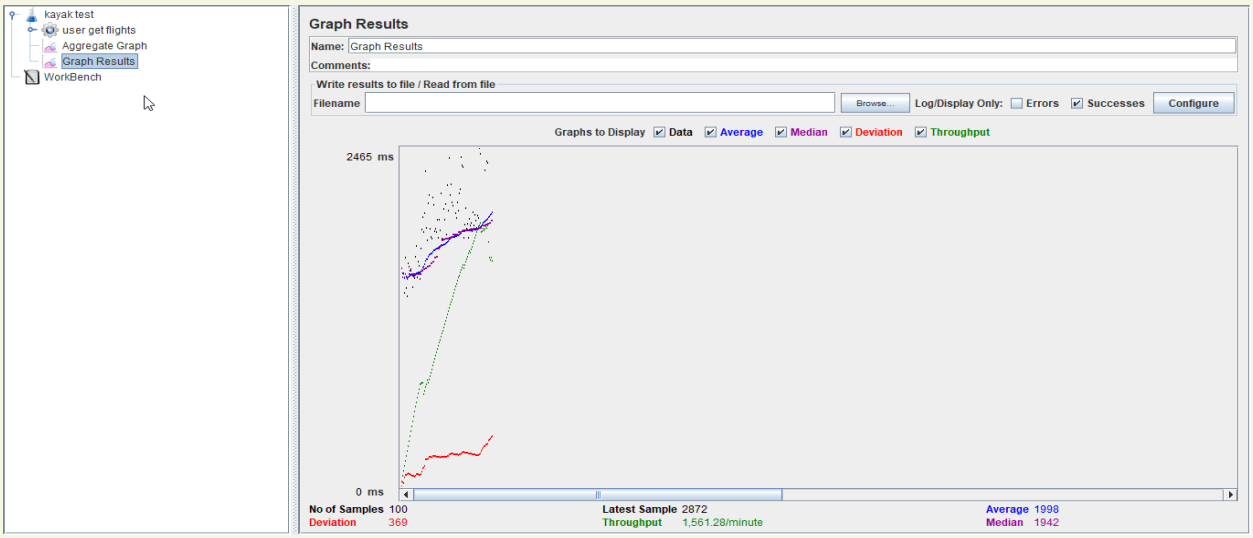




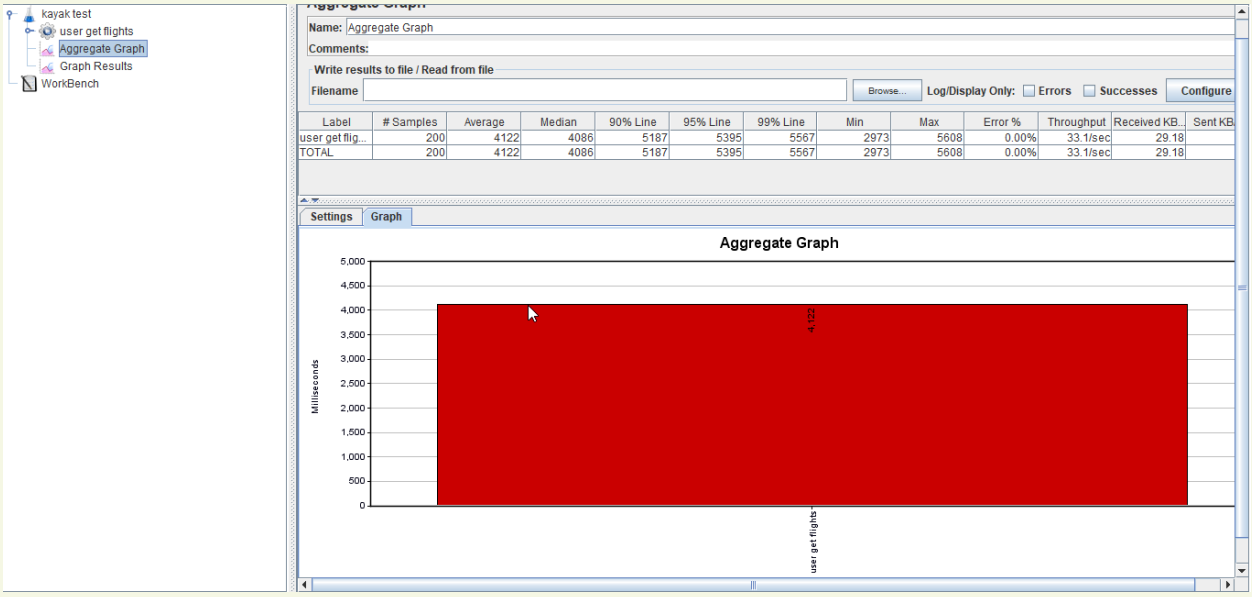
- With connection pooling, no Redis:

1- 100 concurrent users

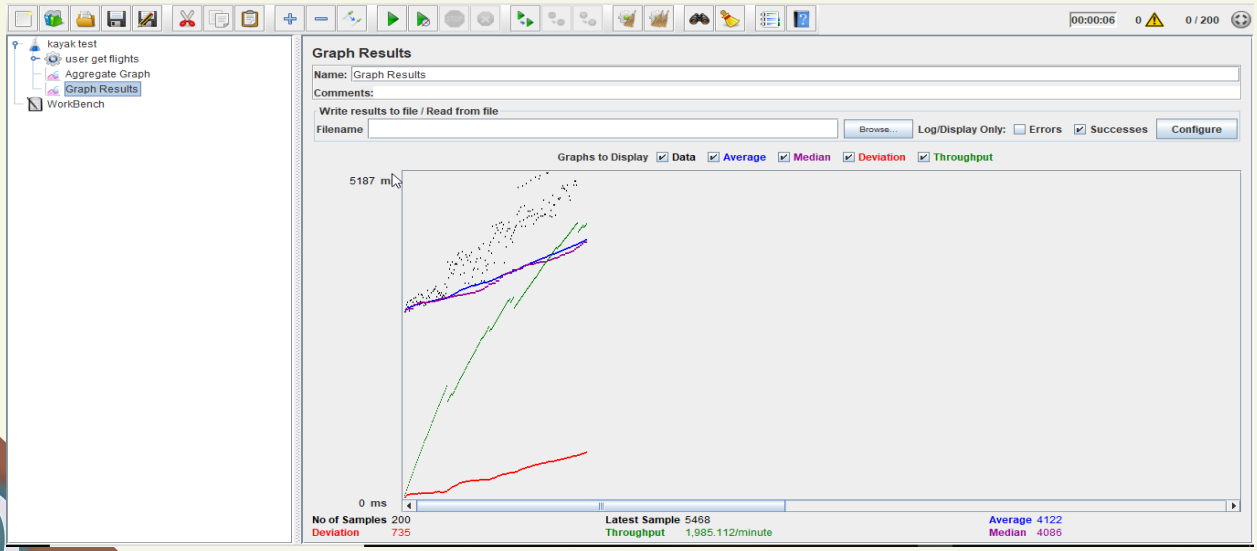




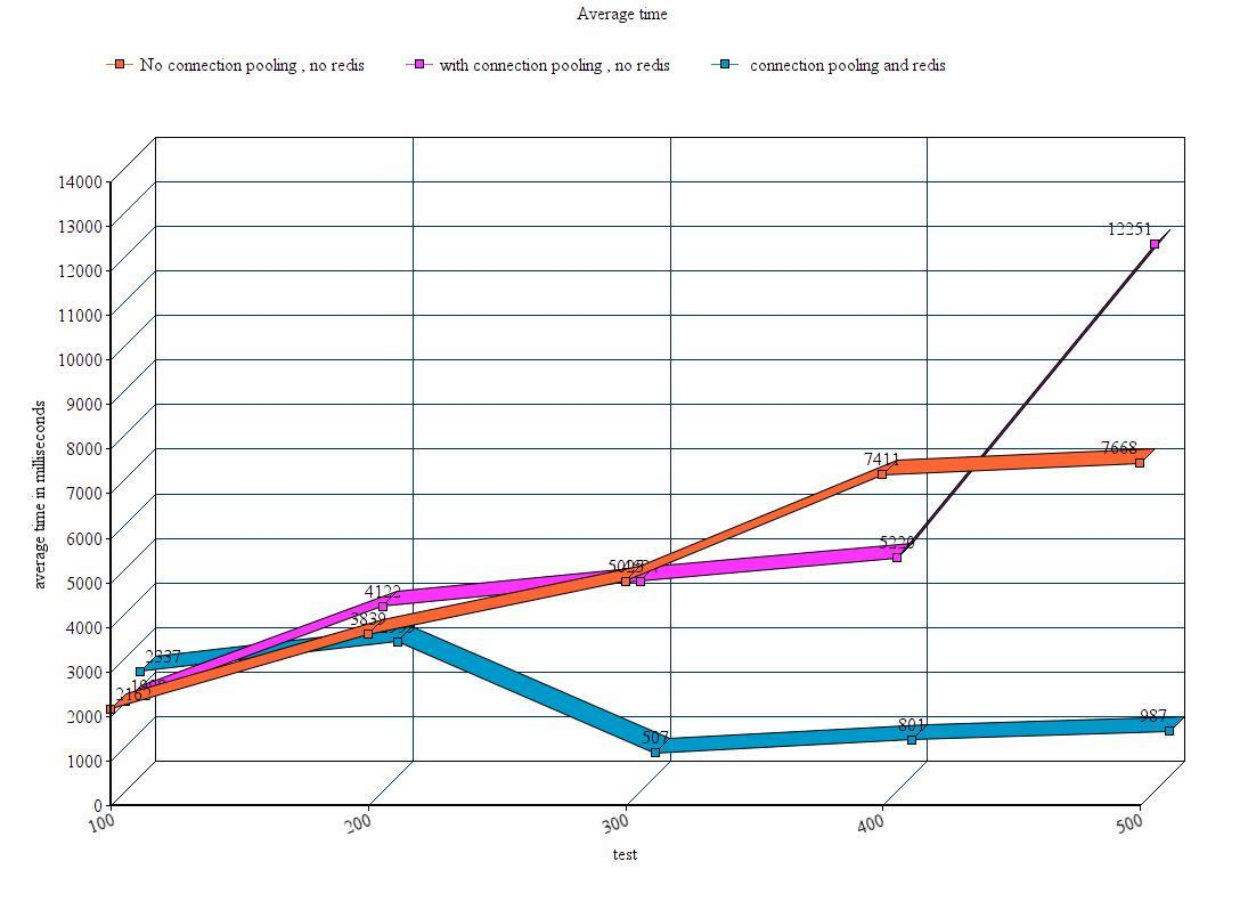
2- 200 users:



PERFORMANCE: -3



PERFORMANCE:



Chapter 6: Result and Discussion

6.1.1 Expected result

- Of the results of the application it must be fast in dealing with the function inside and the time of response when user request a service from it should take a short time.

- Satisfy the customer from the direction of cost so that he does not pay any price to use the application.

- Run time of the application is quick.

- Application configuration is easy and simple.

- Free of defects and errors.

-Serves the needs of user and all available services provided by the application.

-The user interface is easy and very simple for any user who uses this application.

-Application maintenance is very easy to clear and organize application.

6.1.2 Actual result

But actual results are only half the story for effective software testing. What makes the execution a test, rather than production, is that we get the actual results so we can determine whether the software is working correctly. to tell, we compare the actual results to expected software testing results, which are our definition of software testing correctness.

*User's ease of use

*Serves the needs of the user and all available services provided by the application

*The user interface is easy and very simple for any user who uses this application.

7 chapter 7: Conclusion

- Finally, we discuss the conclusions of the work carried out in the document. We present a brief overview of the subject and the work that was carried out. Then, we enumerate the contributions of our work

7.1 Summary:

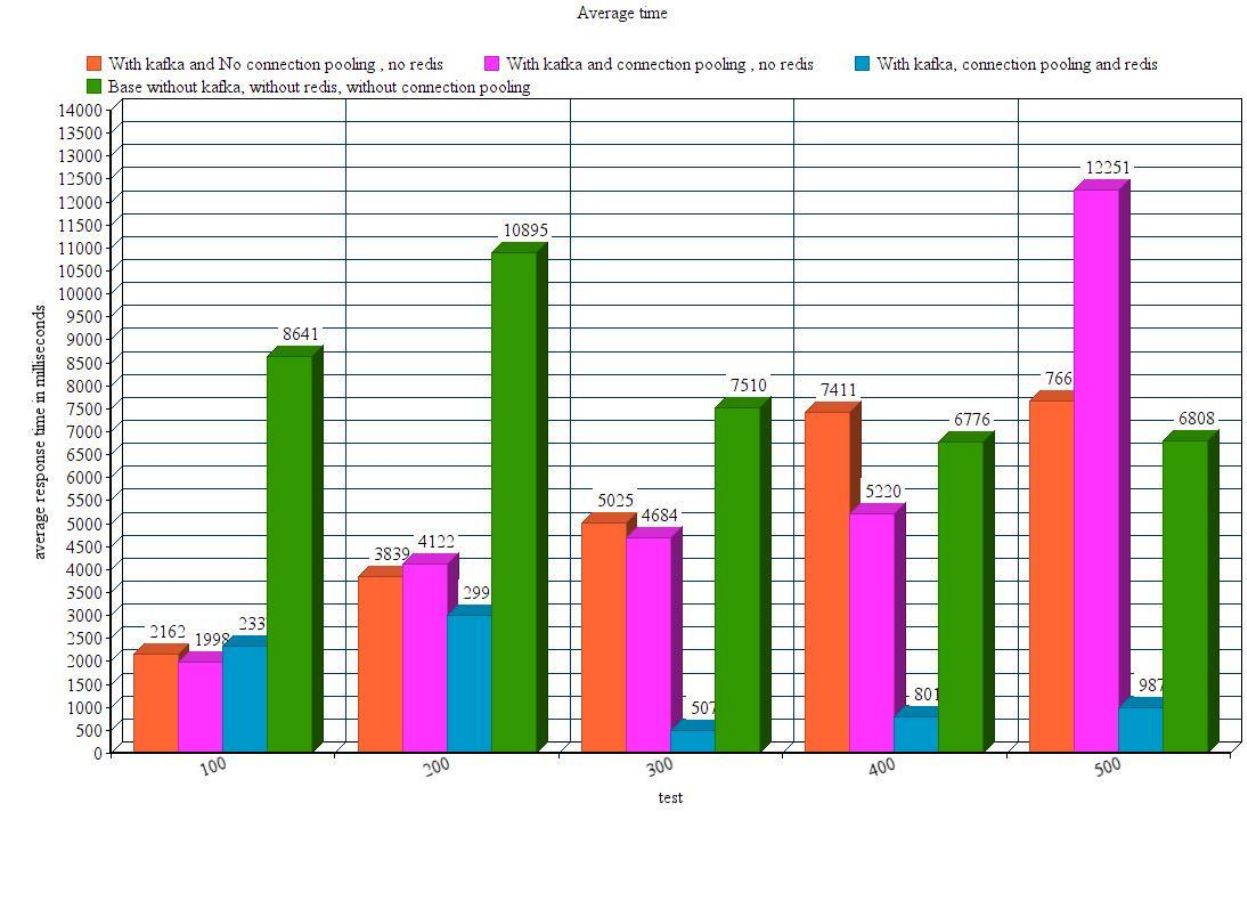
- Travel right is an application that help tourist to send good time thought booking a trip by guiding him to his requirements in a smart and easy way. It also helps him recognize all trips and hotels by showing all information he needs. It also will help him know his total price by calculating the total price of trip including the hotel and the internal movement

7.2 Project Enhancement

- This subject area is so rich that there are many idea that can be suggested, and that still leave room for improvement, here we introduce some new features and enhancements to be applied in next version of

this project, like if we have the right resources we will enhance our project and don't hesitate to use payed tools with advanced features that will improve App functionalities.

Chapter 8: performance Future Work:



8.1.1 Performance comparison:

This Performance comparison graph for four criterias

- Without Kafka, connection pooling or Redis
- With Kafka but no connection pooling and Redis.
- With Kafka and connection pooling but no Redis.
- With Kafka, connection pooling and Redis.

We believe that this graph is a very realistic representation of how the application would perform when deployed on a Enterprise level because as explained earlier we deployed our db on cloud, so this graph also includes the network latency.
We tested the performance of API called list cars which serves the functionality of getting all cars from our mongodb for the specific city and for each of these cars makes a search in mysql database to verify from booking table that the car is available on those dates.

With Kafka we can see the response time is very high.
In our Kafka implementation strategy we tried to minimize the bottleneck for performance, so we had two topics for each api , one for request and other for response so there where multiple consumers listening to these topics simultaneously for request and response instead of single consumer with single topic serving all the requests serially eventually resulting into dropping of messages.

Our approach to Redis implementation was simple and effective. We used it at places which were database and read intensive like flight search car search. So, when a user searches for a listing we create a search string with all criterias for search and use it for listing which is very similar to how kayak does it. we use this search string as key to store entries in Redis. So, when a user

searches for a listing with a search string we see if the search exists in Redis, if it does then use it else send a message over Kafka, get data from db, set it in the cache and show it to user. Here the user will get the data from cache on subsequent search. On our exploration for Redis features we found that we can invalidate the cache after a fixed timeout, so we included that feature and set a time out of 3 sec. Also, we invalidate cache entries when user adds a listing so that latest data is available for user. This boosted the application performance so much that i have even seen response times of 1ms. So, in a hypothetical situation, when 1000 users search for a car in the same city for same from to date within a frame of 3 sec, they will get the response in 1ms. Average response time falls to 980ms from 11000 ms. Also, we used hset instead of set which increases performance of Redis by indexing.

8.1.2 future work:

- Finally, we give a clear insight of future projects, and future updates to be added to this project.
- Add additional experiences like Airbnb, trip advisor, food options, etc.
- Weather API
- the Users Will choose already planned travel through their interests.