*Cairo University  
Faculty of Computers and Information  
Computer Science Department*

***Fs7ny***

Supervised By:

**Dr Cherry Ahmed**

TA. Ashraf Mohey

Implemented By:

|  |  |
| --- | --- |
| **Name** | **ID** |
| Abdel-aziz Abdel-naser | 20140168 |
| Mohamed Fawzy | 20140234 |
| Israa Abdel-naby | 20140055 |

Academic Year 2017-2018

Midyear Documentation of Graduation Project

**ABSTRACT**

Egypt is great country with a lot of sites and place which is wonderful and worth visiting. Although a lot of people straggle when they decide to take a vacation. They face many problem while the process of preparing for that vacation. Starts by the question of where to go? Passing by searching for a traveling company that provide a trip for the desired destination. But where does the search start form?

On the other hand there’s a lot of traveling companies that provides trips and offers. They advertise for their offers using street posters or social media posts and events.

You may be lucky and pass by an advertisement poster for a trip for the destination you want. But the offer may not suit you perhaps it’s the price or the duration of the trip may be the hotel or the transport mean.

So a system is required to be the platform where the customer and the companies can meet. This platform must provide some important features for the customer for example, the ability to search for a trip by price, destination or period, the ability of reordering the search results by company rate, price its period or the departure location, the ability of booking the trip on-line and the ability to rate a company. The customer can also share a trip he/she likes to social media or send its link to his/her friends.

The platform must also provide some important features for the company like the ability to add new trip, modify existing one and add new offer.

To implement this platform our team will use 3 tier web-based architecture with MVC design pattern. First tier is the user-interface tier in which we will use react and react native framework to build a cross-platform application. The second tier is the serve-side tier which will be Lamp stack and the third and last tier is the database tier.

Contents

[1 Introduction 1](#_Toc506765008)

[1.1 Project Idea 1](#_Toc506765009)

[1.2 Problem Significance 1](#_Toc506765011)

[2 System analysis and Design 3](#_Toc506765012)

[2.1 system architecture 3](#_Toc506765013)

[2.2 Stakeholders 4](#_Toc506765014)

[2.3 Functional Requirements 4](#_Toc506765015)

[2.4 Non-functional Requirements 13](#_Toc506765016)

[2.5 Use-case Diagram 16](#_Toc506765017)

[2.6 Sample Use-cases 17](#_Toc506765018)

[2.7 Class Diagram 28](#_Toc506765019)

[2.8 Sequence Diagram 29](#_Toc506765020)

[2.8.1 Add Trip 29](#_Toc506765021)

[2.8.2 Book Trip 30](#_Toc506765022)

[2.9 Entity relationship diagram 31](#_Toc506765023)

[2.10 proto type 32](#_Toc506765024)

[2.10.1 Login page 32](#_Toc506765025)

[2.10.2 Home page 33](#_Toc506765026)

[2.10.3 View Trip 34](#_Toc506765027)

[2.10.4 Book Trip page 35](#_Toc506765028)

[2.11 Time plan 36](#_Toc506765029)

[3 Conclusion 37](#_Toc506765030)

Chapter One

# Introduction

## Project Idea

### Nowadays the youth especially in our great country Egypt faces a problem when they decide to take a vacation. The problem of preparing for such trips is managed by many companies that offers trips and voyages but they don’t have a platform to advertise their offers. So a system is required to be the platform where the customer can view and surf offers made by deferent companies and select offer that suits him and be able to book tickets and pay for them on the platform.

## Problem Significance

A platform for the traveling companies to advertise their offers is missing

• Deliver such a platform will be win-win situation for all.

• Creating a cross-platform System

• Previous solutions:

◦ Street-posters

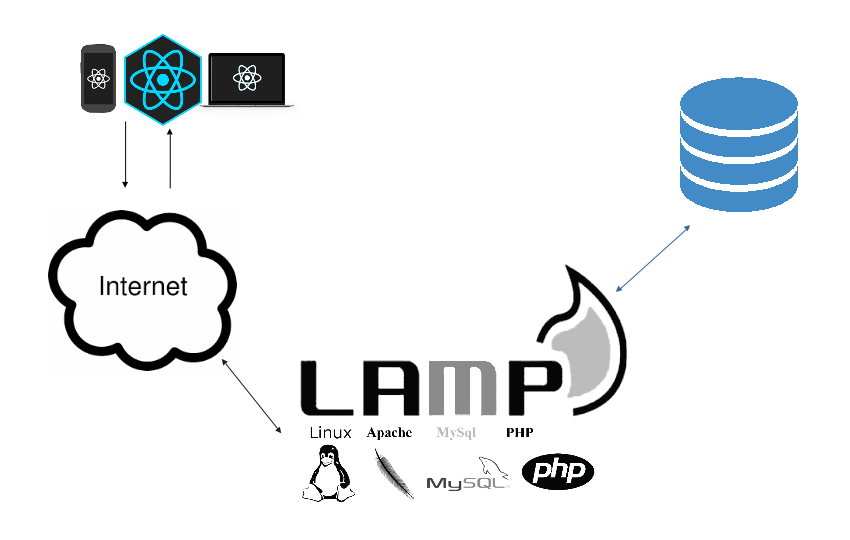
◦ Flayers

◦ Social Media Events

Chapter Two

# System analysis and Design

## system architecture

 System architecture is web-based 3-tier. Where the first tier is the user interface tier in which we will use React-JS and React Native frameworks to support cross platform system. The second tier is server-side is the LAMP stack -which consists of Linux Apache My-SQL and PHP to be the logic layer and The third tier is the database which is responsible of storing system’s data.

## Stakeholders

This project is a graduation project at the faculty of computer science and information technology, Cairo University. Stakeholders for this project splits into two groups within FCI-CU and in market.

In FCI-CU:

- Doctors in CS department

- TAs in CS department

In market:

- Travels Companies

-Customers

## Functional Requirements

2.3.1 **RateCompany**

2.3.1.1 Why: It is important for customers to rate the company that they deal with to let the other customers to know more about this company and the quality of its services.

2.3.1.2 Expected Input: Company name, Rate.

2.3.1.3 Expected Output: Rate value specified for the selected company.

2.3.1.4 Interacts To: Customer

2.3.1.5 Process: The customer asks the system to rate a company and enters the company name and the rate that he decides after finishing the trip and then the system save that rate to the company to be visible for all customers

2.3.2 **SendMessage**

2.3.2.1 Why: The users should be able to interact with each other by sending message and responding to it.

2.3.2.2 Expected Input: Written message text.

2.3.2.3 Expected Output: A hint to the users that the message is sent.

2.3.2.4 Interacts To: User.

2.3.2.5 Process: The user asks the system to send a new message and then enter the message and choose the other user to be sent to.

2.3.3 **ShareTrip**

2.3.3.1 Why: The customer may need to share a trip with other customers “Friends” to attend this trip with him/her.

2.3.3.2 Expected Input: Trip.

2.3.3.3 Expected Output: A shared trip between customers that related to the sharer customer.

2.3.3.4 Interacts To: Customer.

2.3.3.5 Process: The customer asks the system to share a trip and choses it then the system shares it between the customers related to that sharer customer.

2.3.4 **ReserveTrip**

2.3.4.1 Why: The user must be able to reserve a trip as he wants to attend.

2.3.4.2 Expected Input: Trip that the customer specifies.

2.3.4.3 Expected Output: Notification with a successful reservation for the trip.

2.3.4.4 Interacts To: Customer.

2.3.4.5 Process: The customer asks the system to reserve a trip after viewing its details then the system make the customer able to pay the fees and reserve a seat for the customer on the trip.

2.3.5 **PayFees**

2.3.5.1 Why: The customer must pay the fees for the trips that he/she reserves.

2.3.5.2 Expected Input: Trip that the user reserved.

2.3.5.3 Expected Output: Notification with a successful pay and notification from the bank.

2.3.5.4 Interacts To: Customer.

2.3.5.5 Process: After the customer chooses a trip to attend he/she asks the system to pay the fees for this trip and complete the reservation, Then the system deals with the bank to withdraw the price of the trip from the user’s account.

2.3.6 **ReceiveMessage**

2.3.6.1 Why: The user should be able to receive the messages that other users sent to him/her/it to v be able to view and respond to it.

2.3.6.2 Expected Input: User’s id.

2.3.6.3 Expected Output: A list with all messages that had sent to this user.

2.3.6.4 Interacts To: User.

2.3.6.5 Process: The system receive the messages sent to a specific user and make him/her/it able to view and respond to it.

2.3.7 **SearchForTrip**

2.3.7.1 Why: The customer must be able to search for a trip and view all trips related to it and also be able to sort these trips with a specific type such as price, period, location, date.

2.3.7.2 Expected Input: Search key, Sort type.

2.3.7.3 Expected Output: A list with all trips related to the trip that the user has searched for.

2.3.7.4 Interacts To: Customer.

2.3.7.5 Process: The user asks the system to search for a trip by entering a search key and sort type then the system lists all trips that match with this key and sorted by the type that the user specifies.

2.3.8 **SignUp**

2.3.8.1 Why: The user must be able to sign up to the system to be able to use its functions

2.3.8.2 Expected Input: User name, email, password, phone number, ...

2.3.8.3 Expected Output: Notification with a successful Sign up and ability to log in and use the system.

2.3.8.4 Interacts To: User.

2.3.8.5 Process: The user asks the system to sign up and then enters information about him/her/itself the system checks these information and then send a notification to the user with the successful sign up and log the user in to the system.

2.3.9 **SignIn**

2.3.9.1 Why: Necessary feature enables the user to login to the system

2.3.9.2 Expected Input: userName, password

2.3.9.3 Expected Output: Confirmation Message

2.3.9.4 Interacts to: Customer or Company

2.3.9.5 Process: Receives the parameters from The View working with the model to search the database for the userName and returns true if found and the password matches else returns false

2.3.10 **ViewTrips**

2.3.10.1 Why: enables the user to view all trips on the system

2.3.10.2 Expected Input: None

2.3.10.3 Expected Output: List of all trips

2.3.10.4 Interacts to: Customer or Company

2.3.10.5 Process: Calls the TripModel to get all Trips from the database sort and return them to the view

2.3.11 **ExpandTrip**

2.3.11.1 Why: enables the user of view trip’s details

2.3.11.2 Expected Input: tripId

2.3.11.3 Expected Output: List of Details

2.3.11.4 Interacts to: Customer or Company

2.3.11.5 Process: Calls the TripModel to get all details about the trip prepare the and return them to the view

2.3.12 **AddTrip**

2.3.12.1 Why: enables a Company of adding new trip

2.3.12.2 Expected Input: trip details

2.3.12.3 Expected Output: confirmation message

2.3.12.4 Interacts to: Company

2.3.12.5 Process: gets the parameters from the View and sends them to the Model to store them

2.3.13 EditTrip

2.3.13.1 Why: enables a Company of editing/updating trip details

2.3.13.2 Expected Input: tripId, tripDetails

2.3.13.3 Expected Output: confirmation message

2.3.13.4 Interacts to: Company

2.3.13.5 Process: Gets the parameters for the view and sends them to the model to update the trip details.

2.3.14 **CancelTrip**

2.3.14.1 Why: enables a Company of canceling trip but still viewable for the users

2.3.14.2 Expected Input: tripId, cancelMessage

2.3.14.3 Expected Output: Confirmation Message

2.3.14.4 Interacts to: Company

2.3.14.5 Process: gets a tripId and mark it as canceled and add the cancel message to the trip details

2.3.15 **RemoveTrip**

2.3.15.1 Why: enable a Company of deleting a trip

2.3.15.2 Expected Input: tripId

2.3.15.3 Expected Output: Confirmation message

2.3.15.4 Interacts to: Company

2.3.15.5 Process: gets a trip id and calls the TripModel to delete the trip from the database

2.3.16 **Report**

2.3.16.1 Why: enable a user of Re-porting other user

2.3.16.2 Expected Input: userId, reportMessage

2.3.16.3 Expected Output: confirmation Message

2.3.16.4 Interacts to: Customer or Company

2.3.16.5 Process: gets a reported user id and calls ReportModel to add new report with the report message.

## Non-functional Requirements

**2.4.1 ID: QR1**

2.4.1.1 TAG: Usability

2.4.1.2 TITLE: Prominent search feature

2.4.1.3 DESCRIPTION: The search feature should be prominent and easy to find for the user.

2.4.1.4 WHY: In order to for a user to find the search feature easily

**2.4.2 ID: QR2**

2.4.2.1 TAG: Usability

2.4.2.2 TITLE: Usage of the search feature

2.4.2.3 DESCRIPTION: The different search options should be evident, simple and easy to understand.

2.4.2.4 WHY: In order to for a user to perform a search easily.

**2.4.3 ID: QR3**

2.4.3.1 TAG: Usability

2.4.3.2 TITLE: Usage of the result in the list view

2.4.3.3 DESCRIPTION: The results displayed in the list view should be ordered by user selected attributes or by default ordered by departure date and company rate. Selecting trip should be in one tap

2.4.3.4 WHY: In order to make the user able to find his trip easily.

**2.4.4 ID: QR4**

2.4.4.1 TAG: Response Time

2.4.4.2 TITLE: The fastness of the search

2.4.4.3 SCALE: The response time of a search

2.4.4.4 MEASURE: Measurements obtained from 1000 searches during testing.

2.4.4.5 MUST: No more than 2 seconds 100% of the time.

2.4.4.6 WISH: No more than 1 second 100% of the time.

**2.4.5 ID: QR5**

2.4.5.1 TAG: Reliability

2.4.5.2 TITLE: The fault tolerance of the system.

2.4.5.3 SCALE: If the system loses the connection to the Internet or the system gets some strange input, the user should be informed.

2.4.5.4 MEASURE: Measurements obtained from trials during testing.

2.4.5.5 MUST: 100% of the time.

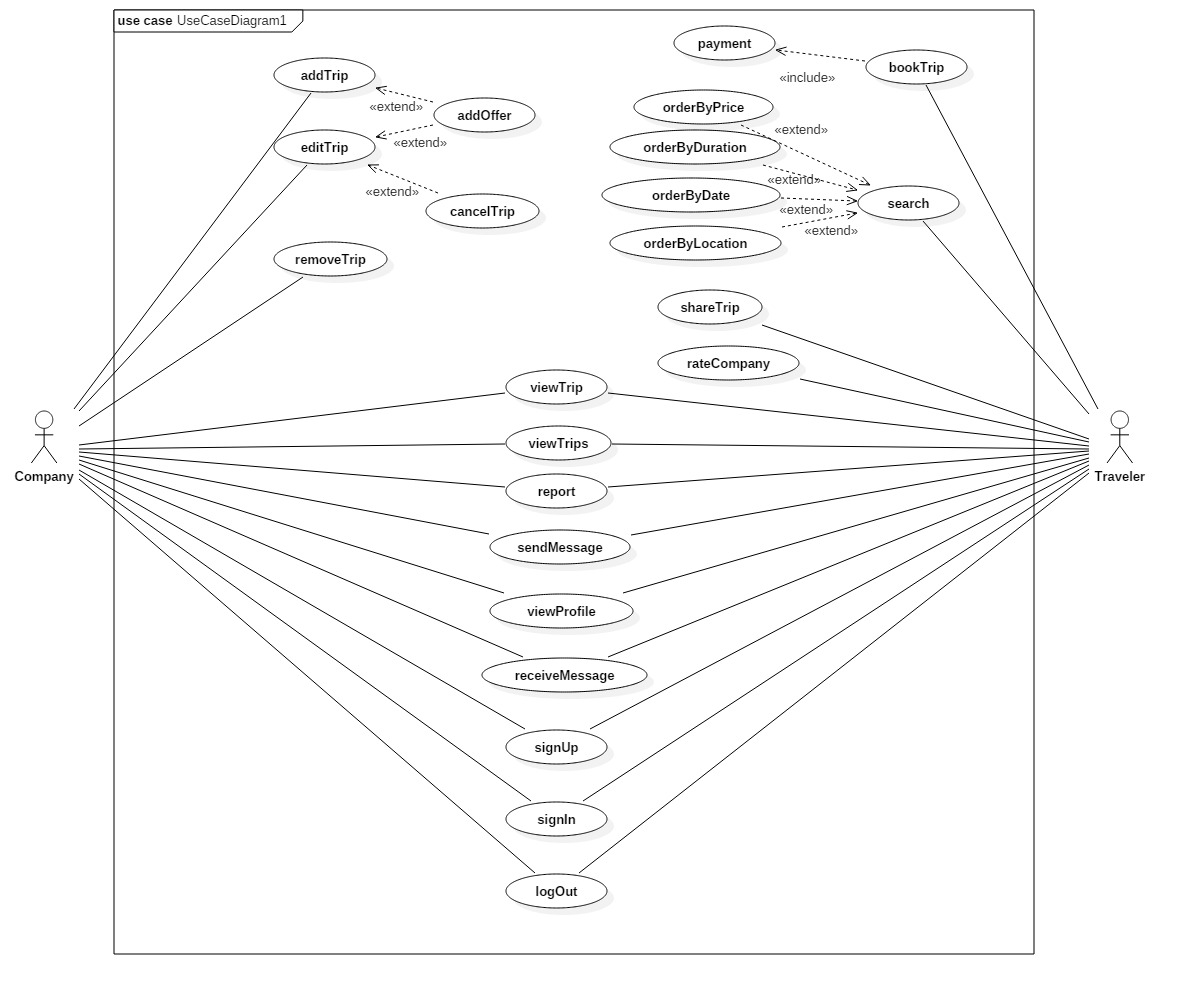
**2.4.6 ID: QR6**

2.4.6.1 TITLE: Internet Connection

2.4.6.2 DESCRIPTION: The application should be connected to the Internet.

2.4.6.3 WHY: In order for the application to communicate with the database.

## Use-case Diagram

 This is the Fs7ny’s Use-case Diagram that shows the functions of the system and the actor who can use it.

## Sample Use-cases

|  |  |  |
| --- | --- | --- |
| Use Case ID: | 001 | |
| Use Case Name: | Sign Up | |
| Actors: | Company, Customer | |
| Pre-conditions: |  | |
| Post-conditions: | The user logs in and accesses the contents of the website and can make his/her allowed functions | |
| Flow of events: | **User Action** | **System Action** |
| The user registers as company or Traveler |  |
|  | The system displays the form to user to enter his/her email and password….and all the required information |
| The user enters all the required information |  |
|  | The system validates the user's input and saves the data in the data-base. |
| Exceptions: | **User Action** | **System Action** |
| **1.** The user enters his/her information and the e-mail is not valid. |  |
|  | The system displays an error |
|  | **2.** The user enters an email that already exists in the system data-base |  |
|  |  | The system replies with error. |
| Includes: | Validations | |
| Notes and Issues: | The validation function would be between the system and the user, it will send message to the email the user entered to verify, and on the other side it has a data to check if the data that the user entered is unique or not. | |

|  |  |  |
| --- | --- | --- |
| Use Case ID: | 002 | |
| Use Case Name: | logIn | |
| Actors: | Company, general User | |
| Pre-conditions: | The user enters his (Username, Password) correctly. | |
| Post-conditions: | The user can logs in and accesses the contents of the website and can make his/her allowed functions | |
| Flow of events: | **User Action** | **System Action** |
| The user enters his (Username, Password) |  |
|  | The system validates the input of the user. |
|  | The system enters the user to the site. |
| Exceptions: | **User Action** | **System Action** |
| **1.** The user enters his/her Username or Password and one of them is not Correct. |  |
|  | The system displays “Wrong Username or Password” |
|  | **2.** user forgot his Password  User selects “Forgot Password” |  |
|  |  | System asks the user to enter his e-mail.  System sends a recovery mail to the user with a link to create new Password. |
|  | Company follows the link. |  |
|  | Company enters a new Password and press submit. |  |
|  |  | System updates the password in the data-base |
| Includes: | Validations | |

|  |  |  |
| --- | --- | --- |
| Use Case ID: | 003 | |
| Use Case Name: | addTrip | |
| Actors: | Company | |
| Pre-conditions: | Logged-in user. | |
| Post-conditions: | Company saves the Trip | |
| Flow of events: | **User Action** | **System Action** |
|  | Company chooses “Add Trip” |  |
|  |  | Site displays the form that contains the information needed for creating a new Trip |
|  | Company inputs the information and press “Add” |  |
|  |  | System saves the information in the data-base |

|  |  |  |
| --- | --- | --- |
| Use Case ID: | 004 | |
| Use Case Name: | editTrip | |
| Actors: | Company | |
| Pre-conditions: | Logged-in user. | |
| Post-conditions: | Company add changes to the Trip. | |
| Flow of events: | **User Action** | **System Action** |
|  | Company chooses “Edit Trip” |  |
|  |  | Site displays the information of the Trip. |
|  | Company inputs the changes and press “save” |  |
|  |  | System saves the changes in the data-base |
| Includes: | AddOffer | |

|  |  |  |
| --- | --- | --- |
| Use Case ID: | 005 | |
| Use Case Name: | bookTrip | |
| Actors: | Traveler | |
| Pre-conditions: | Logged-in user. | |
| Post-conditions: | Traveler books the trip. | |
| Flow of events: | **User Action** | **System Action** |
|  | Traveler chooses “Book Trip” |  |
|  |  | Site displays the payment form to let the user enter his credit-card ID. |
|  | Traveler inputs his data and clicks “Book”. |  |
| Exceptions: | **User Action** | **System Action** |
|  | user enters invalid credit-card |  |
|  |  | System displays “Invalid credit-card” message to the user. |

|  |  |  |
| --- | --- | --- |
| Use Case ID: | 006 | |
| Use Case Name: | viewTrips | |
| Actors: | Company, general User | |
| Pre-conditions: | Logged-in user. | |
| Post-conditions: | User views all trips on the system. | |
| Flow of events: | **User Action** | **System Action** |
|  | User chooses “view trips” |  |
|  |  | System views all Trips available in the site. |

|  |  |  |
| --- | --- | --- |
| Use Case ID: | 007 | |
| Use Case Name: | viewTrip | |
| Actors: | Company, general User | |
| Pre-conditions: | Logged-in user. | |
| Post-conditions: | User views the information of a certain trip. | |
| Flow of events: | **User Action** | **System Action** |
|  | User selects certain trip and chooses “View Trip” |  |
|  |  | System displays all the information about the trip. |

|  |  |  |
| --- | --- | --- |
| Use Case ID: | 008 | |
| Use Case Name: | Search | |
| Actors: | Traveler | |
| Pre-conditions: | Logged-in user. | |
| Post-conditions: | Traveler finds trips those meet his requirements. | |
| Flow of events: | **User Action** | **System Action** |
|  | Traveler chooses “Search” and selects the order he wants the trips to be viewed into. |  |
|  |  | System views all Trips that match user requirements. |
| Exceptions: | **User Action** | **System Action** |
|  | Travelersearches for requirements that doesn’t match any trip in the system. |  |
|  |  | System displays “no matching result” message. |

|  |  |  |
| --- | --- | --- |
| Use Case ID: | 009 | |
| Use Case Name: | shareTrip | |
| Actors: | Traveller | |
| Pre-conditions: | Logged-in user. | |
| Post-conditions: | Traveller shares the trip to the social media. | |
| Flow of events: | **User Action** | **System Action** |
|  | Traveller selects a certain trip and clicks “share” |  |
|  |  | System asks the user to choose the area to share to. |

|  |  |  |
| --- | --- | --- |
| Use Case ID: | 010 | |
| Use Case Name: | viewProfile | |
| Actors: | Company, general User | |
| Pre-conditions: | Logged-in user. | |
| Post-conditions: | User views the profile of the company. | |
| Flow of events: | **User Action** | **System Action** |
|  | User selects a certain user and selects “view profile” |  |
|  |  | System loads the profile of the selected user. |

|  |  |  |
| --- | --- | --- |
| Use Case ID: | 011 | |
| Use Case Name: | Report | |
| Actors: | Company, Traveller | |
| Pre-conditions: | Logged-in user | |
| Post-conditions: | User reports another user. | |
| Flow of events: | **User Action** | **System Action** |
|  | User selects a certain user and clicks “report” |  |
|  |  | System takes action towards the user. |

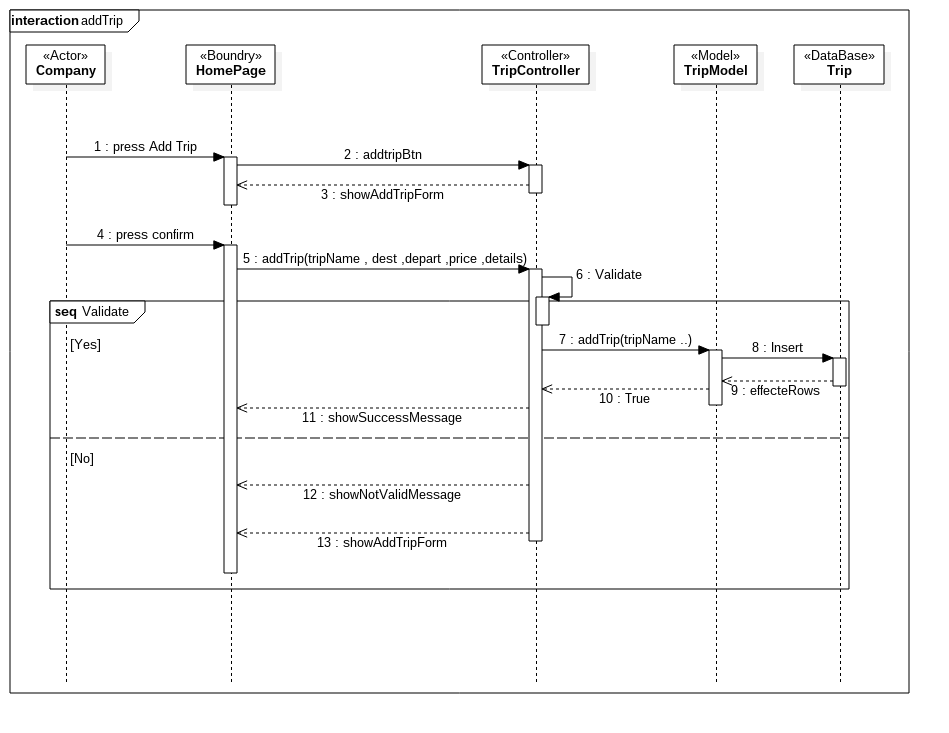
|  |  |  |
| --- | --- | --- |
| Use Case ID: | 012 | |
| Use Case Name: | rateCompany | |
| Actors: | Traveller | |
| Pre-conditions: | A registered user with internet connection. | |
| Post-conditions: | User gives rate to the company | |
| Flow of events: | **User Action** | **System Action** |
|  | User selects a certain company and selects “rate” |  |

|  |  |  |
| --- | --- | --- |
| Use Case ID: | 013 | |
| Use Case Name: | removeTrip | |
| Actors: | Company | |
| Pre-conditions: | Logged-in user. | |
| Post-conditions: | Company removes the trip from the system. | |
| Flow of events: | **User Action** | **System Action** |
|  | Company selects a certain trip and chooses “remove” |  |
|  |  | System removes all the trip data from the database. |

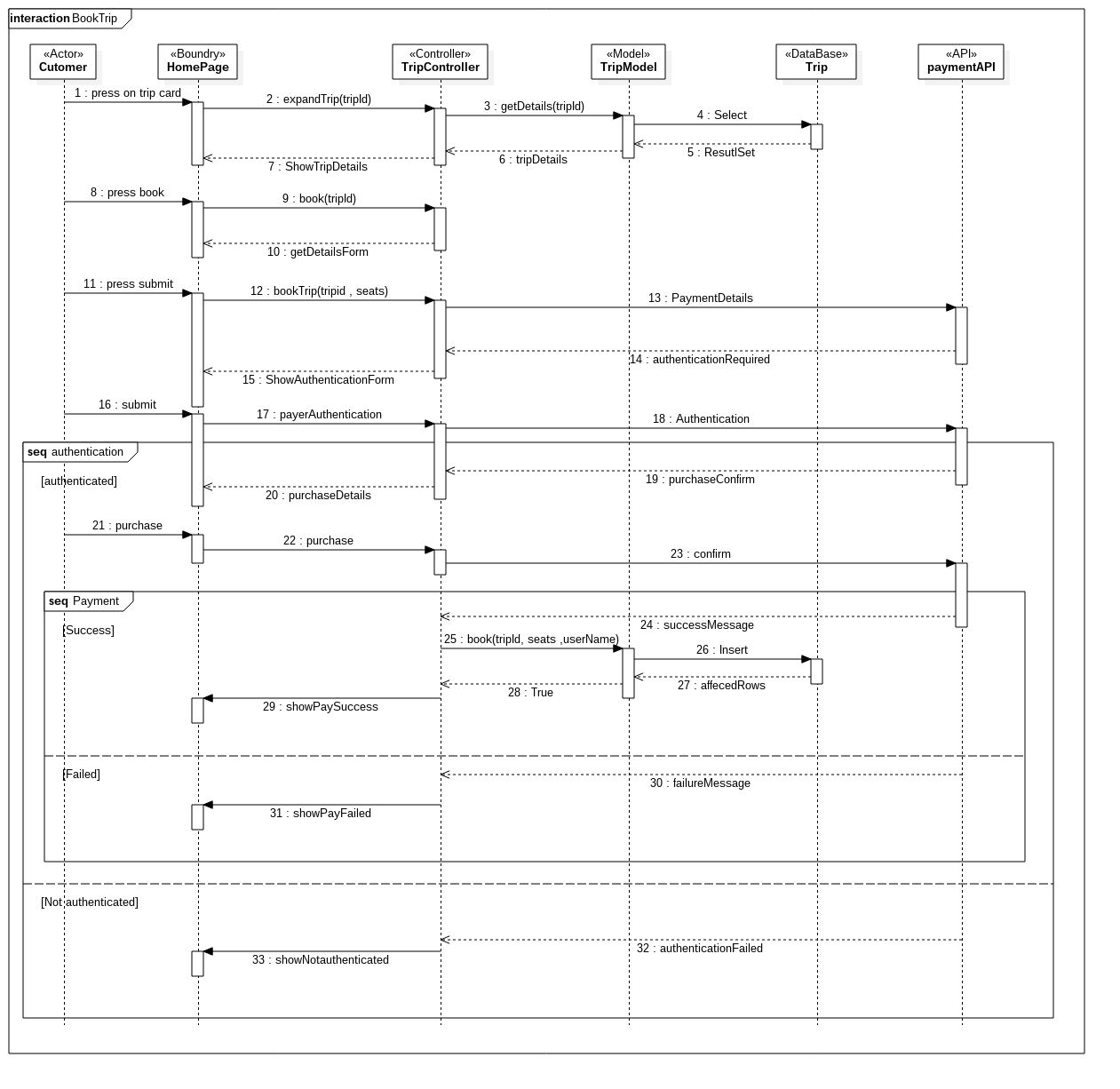
## Class Diagram

## Sequence Diagram

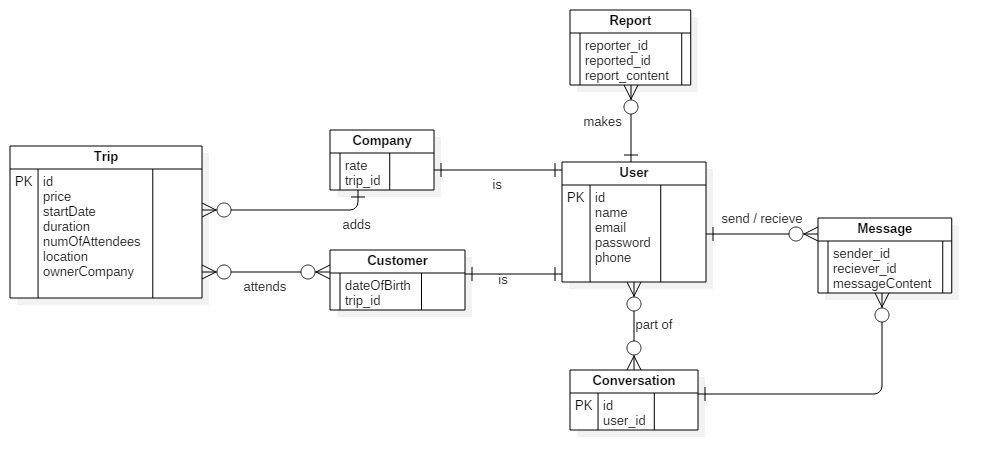
### Add Trip



### Book Trip

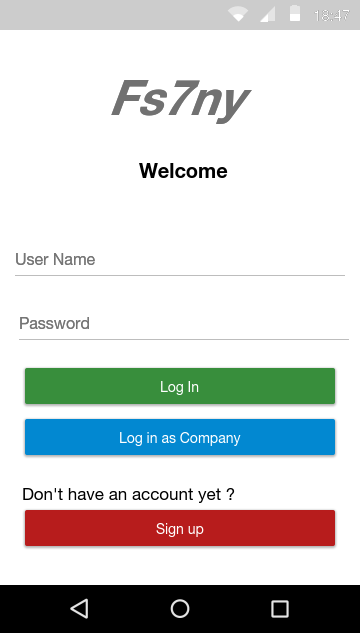


## Entity relationship diagram



## proto type

### Login page



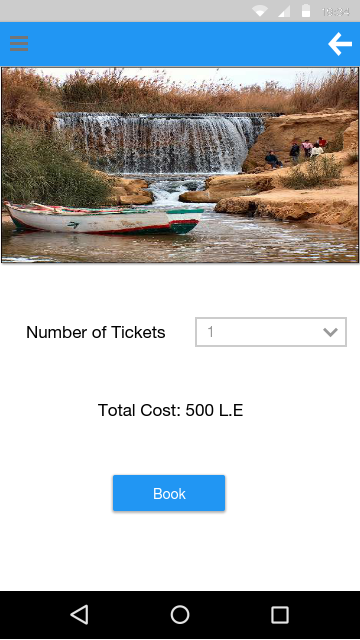
### Home page



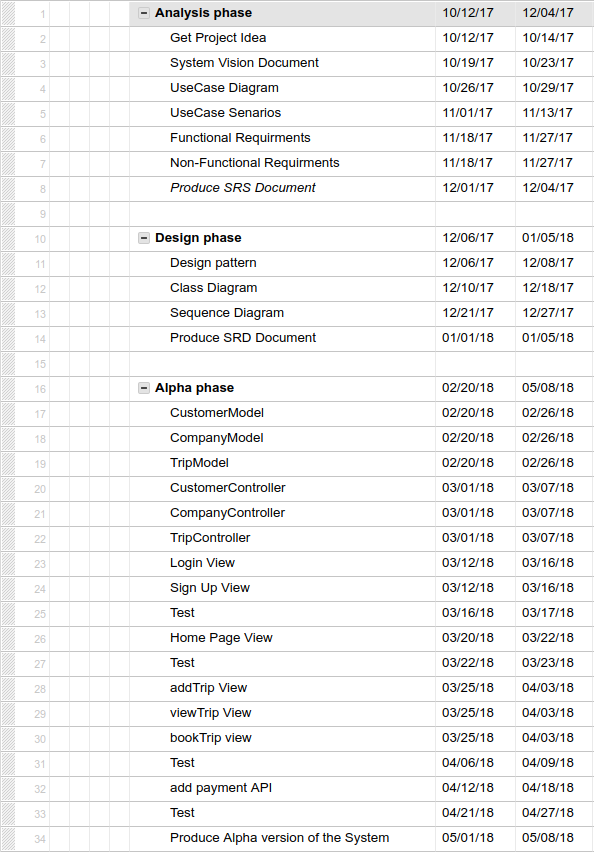
### View Trip



### Book Trip page



## Time plan



Chapter Three

# Conclusion

Finally we find that Fs7ny platform meets the solution for the mentioned problem as it provides these features:

* For customer
  + Search for a trip and find the best offer
  + Book ticket(s) and pay online
* For company
  + Add trip(s)
  + Edit a Trip and add offers
  + Sell tickets online