**Faculty of Engineering, Ain Shams University.**

Computer Engineering and Software Program.

Final Project Document for

Quest

CSE232: Advance Software Engineering – Fall 2021

STUDENTS’ NAME

1. Nada Amr Attia 19P1621
2. Alaa Mohamed Mohamed Hamdy Ahmed 19P6621
3. Anas Salah Abdelrazik Saad Eldin 19P9033
4. Ahmed Amr MohyEldin Elgayar 19P8349
5. Ahmed Akram Mohamed 19P4777
6. Mohamed Saber Mohamed 19P9544

**Abstract:**

Quest is a website that helps people have fun in Egypt. You do not need to go to the same place every time because you do not know other places; this is where Quest turn comes in. It offers variety of places to visit, offering promotions and discounts and also provides extra services such as car renting and tour guides for foreigners and for citizens as well. With Quest, you do not need to worry about your next outing with your friends or family, your budget because Quest will always find you the best place to go to according to your preferences and also your budget. Quest helps you also organize trips so that you can search for hotels or even join a trip organized by Quest. Quest cares about the individuals as well as the businesses in Egypt; not only it provides entertainment for its users, but it also helps marketing for local businesses in Egypt thus helping Egypt economically. Quest is a joy-safer!

Table of Contents

[**1.0 Introduction** 2](#_Toc91864319)

[1.1 Purpose 2](#_Toc91864320)

[1.2 List of Definitions 2](#_Toc91864321)

[1.3 Scope 2](#_Toc91864322)

[1.4 List of References 2](#_Toc91864323)

[1.5 Overview 2](#_Toc91864324)

[**2.0 General Description** 2](#_Toc91864325)

[2.1 Product Perspective: 2](#_Toc91864326)

[2.2 General Capabilities 2](#_Toc91864327)

[2.3 General Constraints 2](#_Toc91864328)

[2.4 User Characteristics 3](#_Toc91864329)

[2.5 Environment Description 3](#_Toc91864330)

[2.6 Assumptions and Dependencies 3](#_Toc91864331)

[**3.0 Specific Requirements** 4](#_Toc91864332)

[3.1 Capability Requirements 4](#_Toc91864333)

[3.2 Constraint Requirements 5](#_Toc91864334)

[**4.0 Use-Case Diagram and the Narrative Description of all use cases** 6](#_Toc91864335)

[**5.0 Swimlane Diagrams of all use cases** 66](#_Toc91864336)

[**6.0 Noun Extraction and CRC Cards** 69](#_Toc91864337)

[6.1 Noun Extraction: 69](#_Toc91864338)

[6.2 CRC Cards: 70](#_Toc91864339)

[**7.0 Class Model** 72](#_Toc91864340)

[**8.0 State Diagram** 76](#_Toc91864341)

[**9.0 Interaction Diagrams** 77](#_Toc91864342)

[**10.0 Class Diagram** 79](#_Toc91864343)

[**11.0 Client-Object Relation Diagram** 80](#_Toc91864344)

[**12.0 OOAD Methodologies** 81](#_Toc91864345)

[12.1 Jacobson et al. Methodologies 81](#_Toc91864346)

[12.2 Object modelling Technique OMT 82](#_Toc91864347)

[**13.0 Comparative Analysis of the Output of the Adopted Methodologies** 91](#_Toc91864348)

[**14.0 Architectural Model** 93](#_Toc91864349)

[**15.0 Component Diagram** 98](#_Toc91864350)

[**16.0 Detailed Design** 99](#_Toc91864351)

[16.1 Database Schema 99](#_Toc91864352)

[**17.0 Testing** 100](#_Toc91864353)

[**18.0 Estimated Project Cost** 104](#_Toc91864354)

[18.1 Function Points 104](#_Toc91864355)

[18.2 COCOMO II Model 106](#_Toc91864356)

[18.3 Expert Judgement 108](#_Toc91864358)

[18.4 Comparative Analysis 108](#_Toc91864359)

[**19.0 User Guide** 110](#_Toc91864360)

[**20.0 Time Plan** 121](#_Toc91864361)

[**21.0 Appendices** vii](#_Toc91864362)

**List of Figures and Tables**

[Figure 1: Use Case diagram for Admin 6](file:///C:\Users\lenovo\Desktop\QuestDocument.docx#_Toc91864609)

[Figure 2: Use Case diagram for User 15](file:///C:\Users\lenovo\Desktop\QuestDocument.docx#_Toc91864610)

[Figure 3: Use Case diagram for Travel Agency 35](file:///C:\Users\lenovo\Desktop\QuestDocument.docx#_Toc91864611)

[Figure 4: Use Case diagram for Car Rental Agency 50](file:///C:\Users\lenovo\Desktop\QuestDocument.docx#_Toc91864612)

[Figure 5: Swimlane diagram for Travel Agency 66](file:///C:\Users\lenovo\Desktop\QuestDocument.docx#_Toc91864613)

[Figure 6: Swimlane diagram for Car Rental Agency 67](file:///C:\Users\lenovo\Desktop\QuestDocument.docx#_Toc91864614)

[Figure 7: Swimlane diagram for User 68](file:///C:\Users\lenovo\Desktop\QuestDocument.docx#_Toc91864615)

[Figure 8: Iteration 1 in Class diagram 72](file:///C:\Users\lenovo\Desktop\QuestDocument.docx#_Toc91864616)

[Figure 9: Iteration 2 in Class diagram 73](file:///C:\Users\lenovo\Desktop\QuestDocument.docx#_Toc91864617)

[Figure 10: Iteration 3 in Class Diagram 74](file:///C:\Users\lenovo\Desktop\QuestDocument.docx#_Toc91864618)

[Figure 11: Iteration 4 in Class diagram 75](#_Toc91864619)

[Figure 12: State Diagram 76](file:///C:\Users\lenovo\Desktop\QuestDocument.docx#_Toc91864620)

[Figure 13: Sequence diagram for user 77](file:///C:\Users\lenovo\Desktop\QuestDocument.docx#_Toc91864621)

[Figure 14: sequence diagram for Admin 78](file:///C:\Users\lenovo\Desktop\QuestDocument.docx#_Toc91864622)

[Figure 15: Class diagram 79](#_Toc91864623)

[Figure 16: Client Object diagram 80](file:///C:\Users\lenovo\Desktop\QuestDocument.docx#_Toc91864624)

[Figure 17: Level 0 in DFD diagram 83](#_Toc91864625)

[Figure 18: Level 1 in DFD diagram 83](file:///C:\Users\lenovo\Desktop\QuestDocument.docx#_Toc91864626)

[Figure 19: Level 2 for (1.0 Manager User) Component 84](file:///C:\Users\lenovo\Desktop\QuestDocument.docx#_Toc91864627)

[Figure 20: Level 2 for (2.0 Manage Activity) Component 85](file:///C:\Users\lenovo\Desktop\QuestDocument.docx#_Toc91864628)

[Figure 21: Level 2 for (Manage Agency) Component 85](file:///C:\Users\lenovo\Desktop\QuestDocument.docx#_Toc91864629)

[Figure 22: Level 2 (Manage Admin) Component 86](file:///C:\Users\lenovo\Desktop\QuestDocument.docx#_Toc91864630)

[Figure 23: Level 3 for (1.2 User Activities) Component 87](file:///C:\Users\lenovo\Desktop\QuestDocument.docx#_Toc91864631)

[Figure 24: Level 3 for (1.3 Manage User data) Component 88](file:///C:\Users\lenovo\Desktop\QuestDocument.docx#_Toc91864632)

[Figure 25: Level 3 (Activities Manager) Component 89](file:///C:\Users\lenovo\Desktop\QuestDocument.docx#_Toc91864633)

[Figure 26: DFD Diagram 90](file:///C:\Users\lenovo\Desktop\QuestDocument.docx#_Toc91864634)

[Figure 27: MVC Architecture 93](file:///C:\Users\lenovo\Desktop\QuestDocument.docx#_Toc91864635)

[Figure 28: Class diagram Architecture 94](file:///C:\Users\lenovo\Desktop\QuestDocument.docx#_Toc91864636)

[Figure 29: Layered Architecture 95](file:///C:\Users\lenovo\Desktop\QuestDocument.docx#_Toc91864637)

[Figure 30: Data Base Architecture 96](#_Toc91864638)

[Figure 31: Merge Architecture 97](file:///C:\Users\lenovo\Desktop\QuestDocument.docx#_Toc91864639)

[Figure 32: Component Diagram 98](file:///C:\Users\lenovo\Desktop\QuestDocument.docx#_Toc91864640)

[Figure 33: Data Base Schema 99](#_Toc91864641)

[Figure 34: Testing Graph 103](file:///C:\Users\lenovo\Desktop\QuestDocument.docx#_Toc91864642)

[Figure 35: COCOMO II graph and calculations 108](file:///C:\Users\lenovo\Desktop\QuestDocument.docx#_Toc91864643)

[Figure 36: Home Page in the Website 110](file:///C:\Users\lenovo\Desktop\QuestDocument.docx#_Toc91864644)

[Figure 37: Sign Up Page in the Website 111](file:///C:\Users\lenovo\Desktop\QuestDocument.docx#_Toc91864645)

[Figure 38: Login Page in the Website 111](file:///C:\Users\lenovo\Desktop\QuestDocument.docx#_Toc91864646)

[Figure 39: Entertainment Places Page in The Website 112](file:///C:\Users\lenovo\Desktop\QuestDocument.docx#_Toc91864647)

[Figure 40: Rent Care Page in the Website 113](#_Toc91864648)

[Figure 41: Filtering while choosing your rented car 114](#_Toc91864649)

[Figure 42: Filtering Results (Rent a Car) 114](#_Toc91864650)

[Figure 43: Reserve Trips Page in the Website 115](file:///C:\Users\lenovo\Desktop\QuestDocument.docx#_Toc91864651)

[Figure 44: Pay Page in the Website 115](#_Toc91864652)

[Figure 45: Admin Side View in the website 116](#_Toc91864653)

[Figure 46: Admin add a car to the system in the website 117](#_Toc91864654)

[Figure 47: Admin add a tour company to the system in the website 118](#_Toc91864655)

[Figure 48: Admin confirms tour requests 119](#_Toc91864656)

[Figure 49: Admin confirms car requests 120](#_Toc91864657)

[Figure 50: Time Plan 121](file:///C:\Users\lenovo\Desktop\QuestDocument.docx#_Toc91864658)

[Figure 51: Detailed Time Plan 122](file:///C:\Users\lenovo\Desktop\QuestDocument.docx#_Toc91864659)

# **1.0 Introduction**

## 1.1 Purpose

This document is intended to readers who would like to know how Quest is designed, its internal structure and its processing stages. This document covers everything about Quest including diagrams, user guide, estimated project cost, time plan…etc all included to help the reader understand everything about the website.

## 1.2 List of Definitions

* Methodologies: A methodology is explained as the science of methods.
* Method: is a set of procedures in which a specific goal is approached step by step
* Deployment: includes**all of the steps, processes, and activities that are required to make a software system or update available to its intended users**.
* DFD: Data Flow Diagram

## 1.3 Scope

Quest is a website that allows businesses in Egypt to market their businesses (restaurants, activities. Etc) on the website so that users might be able to view them, and select one of them for their next outings, thus acts as an advertising tool for businesses. While for users, it eases the difficulty of finding a place to go that matches his/her preferences, by providing offers, providing trips and promotions as well as services such as car renting, and tour guides assist in case in visiting historical places in Egypt.

## 1.4 List of References

* Roger S. Pressman, Software Engineering: A Practitioner's Approach, 8th Edition, McGraw-Hill, ISBN: 978-0078022128, 2014.
* The Object-Oriented Thought Process, 4th Edition, Matt Weisfeld, Addison-Wesley Professional, 2013.
* Ian Sommerville, Software Engineering, 10th Edition, Addison Wesley, ISBN: 0133943030, 2015.
* Stephen Schach, Object-Oriented and Classical Software Engineering, 8th Edition, McGraw-Hill, ISBN: 978-0073376189, 2010.
* J. Rumbaugh, I. Jacobson, and G. Booch, The Unified Modeling Language User Guide, 2nd Edition, Addison-Wesley, ISBN: 0321245628, 2004

## 1.5 Overview

This flow of this document is by first showing the diagrams that help us understand how Quest is designed, thus viewing the website from different viewpoints. Then, detailed testing process adopted, Estimated Project Cost, User guide and the Time plan and appendices for this project

# **2.0 General Description**

## 2.1 Product Perspective:

The client will not need any other system in parallel with Quest to get the service, the only system the client would have to access if the user selected to pay using credit/debit card and so the banking system will be used.

## 2.2 General Capabilities

* Provide users with a wide range of restaurants, activities, and hotels recommendations in Egypt’s
* Provide user with a list of offered trips all over Egypt

## General Constraints

#### **2.3.1 Constraints**

* The software is implemented as a website, using html, CSS, and JavaScript languages
* The website should have a low response time for better experience of users
* Website should be user-friendly and easy to use
* Website should have the ability to detect user’s location

#### **2.3.2 Background information**

* Any user having experience in using websites should be able to use Quest.
* The user must have a valid email and password to be able to sign-in.
* User must be able to fill up forms and contacts when needed online
* Background knowledge in Locations and addresses in Egypt or the ability to use GPS.

## User Characteristics

#### **2.4.1 User**

User will be able to search the contents of the website, add his preferences and filter the results. Reserve a ticket for a trip, be able to book in a hotel, request a tour guide, and rent a car.

#### **2.4.2 Local Businesses**

Request admin to market their local business on the website, edit their business description and products’ prices. Offer special discounts and promotions for Quest’s users

#### **Admin**

View Local businesses requests and have the choice to accept or decline their requests. Act as an intermediate between car rental agencies and the user. Also, contacts tour guides per request. Plans trips and also view user’s rating to improve quality of the website and its information reliability.

## Environment Description

For developers edit:

1) SQL server for database

2) Visual studio for editing

For deployment:

1) SQL server

2) AIS SERVER

3) .NET FRAMEWORK 4.6

## 2.6Assumptions and Dependencies

#### **2.6.1 Assumptions**

We assume all users have a basic knowledge in Egypt’s traffic and streets and therefore will be able to choose their preferred area of outings

#### **2.6.2 Dependencies**

The website depends on local businesses willing to market their businesses on Quest. It also depends on the seasonal promotions and offers to be able to provide its users with the best prices and promotions

Quest depends on user ratings to be able to enhance the website’s performance.

# **3.0 Specific Requirements**

3.1 Capability Requirements

* Provide users with a wide range of restaurants, activities, and hotels recommendations in Egypt’s
* Provide user with a list of offered trips all over Egypt
* Provide user with car renting facilities
* Provide users with a tour guide if requested.
* Act as a marketing agency for local businesses
* Allow user to rate places
* Allow user to reserve hotels
* Allow user to search outings and filter results
* Allow user to pay online or cash for the services
* Allow user to view restaurants and places descriptions and view other users’ recommendations and comments
* Allow admin to manipulate the main entertainment places in every spot
* Allow admin to manipulate main spots in capital
* Allow admin to add main entertainment places in every spot in the capital
* Allow admin to add main spots in the capital
* Allow admin to view list of users
* Allow admin to confirm registrations of travel agencies
* Allow admin to confirm registrations of car rental agencies
* Allow travel agencies to manipulate their data information
* Allow travel agencies to cancel user registrations
* Allow travel agencies to offer trips data and manipulate it
* Allow car rental agencies to manipulate their data information
* Allow car rental agencies to cancel user registrations
* Allow car rental agencies to manipulate data on offered cars

3.2 Constraint Requirements

* The software is implemented as a website, using html, CSS, and JavaScript languages
* The website should have a low response time of 5 seconds for better experience of users
* Website should be user-friendly and easy to use
* Website should have the ability to detect user’s location
* Website should be able to recover from errors within a minute maximum
* Website should follow the rules and law of Ministry of Tourism
* Website should have a license from the car agencies to market for their cars
* Website should have a contract or a license from travel agencies to promote for their trips and tour guides.

# **4.0 Use-Case Diagram and the Narrative Description of all use cases**

**A**

Figure : Use Case diagram for Admin

Use Case NameShow List of the Users

Related RequirementsNone

Goal in ContextThe Admin sends a request to the system to show the list of the

users

Preconditions None

Successful End ConditionTheList of the users is shown for the Admin

Failed End ConditionThe List of the Users isn’t shown for the Admin

Primary ActorsAdmin

Secondary ActorsNone

Trigger The Admin asks the system to show the list of the users

Main Flow **Step Action**

1. The Admin asks the system to show the list of the

users

1. The system verifies the request validation
2. The system displays list of the users

Extensions **Step Action**

* 1. The system rejects the request because it isn’t valid
  2. The list of the users isn’t shown for the Admin

Use Case NameConfirm the registration of the Travel Agency

Related RequirementsRequirementC.1

Goal in ContextThe Admin Confirms the travel agency’s registration

Preconditions The travel agency sends a request to register in the system

Successful End ConditionTheTravel agency’s registration is confirmed

Failed End ConditionThe Travel agency’s registration is rejected

Primary ActorsAdmin

Secondary ActorsTravel Agency

Trigger The Admin confirms the travel agency’s registration

Main Flow **Step Action**

1. The Admin asks the system to get the registration of

travel agency

1. The Admin verifies the registration validation
2. The Admin submits the acceptance of the registration
3. The system saves the acceptance of the registration

Extensions **Step Action**

* 1. The admin rejects the registration
  2. The system saves that the registration is rejected

Use Case NameConfirm the registration of the Car Rental Agency

Related RequirementsRequirementD.1

Goal in ContextThe Admin Confirms the car rental agency’s registration

PreconditionsThe car rental agency sends a request to register in the system

Successful End ConditionThecar rental agency’s registration is confirmed

Failed End ConditionThe car rental agency’s registration is rejected

Primary ActorsAdmin

Secondary ActorsCar Rental Agency

Trigger The Admin confirms the car rental agency’s registration

Main Flow **Step Action**

1. The Admin asks the system to get the registration of

car rental agency

1. The Admin verifies the registration validation
2. The Admin submits the acceptance of the registration
3. The system saves the acceptance of the registration

Extensions **Step Action**

* 1. The admin rejects the registration
  2. The system saves that the registration is rejected

**Use Case Name** Add the main spots in the Capital

**Related Requirements** None

**Goal in Context** The Admin requests to add the main spots in the Capital

**Preconditions** None

**Successful End Condition** The main spots in the Capital is added to the system

**Failed End Condition** The main spots in the Capital isn’t added to the system

**Primary Actors** Admin

**Secondary Actors** None

**Trigger** The Admin sends a request to add the main spots in the Capital to

the system

**Main Flow Step Action**

1. The Admin asks the system to add the main spots in

the Capital

1. The system verifies the request validation
2. The system adds the main spots in the Capital

**Extensions Step Action**

* 1. The system rejects the request because it isn’t valid
  2. The system doesn’t add the main spots in the Capital

Use Case NameAdd the main entertainment places in every spot in the Capital

Related RequirementsRequirementA.4

Goal in ContextThe Admin requests to add the main entertainment places in every

spot in the Capital

Preconditions The Admin adds the main spots in the Capital

Successful End ConditionThe main entertainment places in every spot in the Capital is added

to the system

Failed End ConditionThe main entertainment places in every spot in the Capital isn’t

added to the system

Primary ActorsAdmin

Secondary ActorsNone

Trigger The Admin send a request to add the main entertainment places in

every spot in the Capital to the system

Main Flow **Step Action**

* 1. The Admin asks the system to add the main

entertainment places in every spot in the

Capital

* 1. The system verifies the request validation

**Include :: Determine the category**

* 1. The system adds the main entertainment places in

every spot in the Capital

**Extensions Step Action**

* 1. The system rejects the request because it isn’t valid
  2. The system doesn’t add the main entertainment

places in every spot in the Capital

**Use Case Name** Determine the Category

**Related Requirements** RequirementA.5

**Goal in Context** The main entertainment places in every spot in the Capital need to

be added to its category

**Preconditions** The main entertainment places in every spot in the Capital is being

added to the system

Successful End ConditionThe main entertainment places in every spot in the Capital is added

to its category

Failed End ConditionThe main entertainment places in every spot in the Capital isn’t

added to its category

Primary ActorsAdmin

Secondary ActorsNone

Trigger Define the category ofthe main entertainment places in every spot

in the Capital while being added to the system

Main Flow  **Step Action**

1. The admin defines the main entertainment places

category

1. The system verifies the existence of the category
2. The system adds the main entertainment places

Extensions **Step Action**

* 1. The system isn’t able to find the category that

requested by the admin

* 1. The system doesn’t add the main entertainment

places in every spot in the Capital

**Use Case Name** Manipulate the main entertainment places in every spot in the

Capital

**Related Requirements** RequirementA.5

**Goal in Context** The Admin requests to manipulate the main entertainment places

in every spot in the Capital

**Preconditions** The Admin adds the main entertainment places in every spot in the

Capital

**Successful End Condition** The main entertainment places in every spot in the Capital is

manipulated

**Failed End Condition** The main entertainment places in every spot in the Capital isn’t

manipulated

**Primary Actors** Admin

**Secondary Actors** None

**Trigger** The Admin send a request to manipulate the main entertainment

places in every spot in the Capital to the system

**Main Flow Step Action**

* 1. The Admin asks the system to manipulate the main entertainment places in every spot

in the Capital

2. The system verifies the existence of the

entertainment places

3. The system manipulates the main entertainment

places in every spot in the Capital

**Extensions Step Action**

* 1. The system isn’t able to find the entertainment places

to apply the modification

* 1. The system doesn’t manipulate the main

entertainment places in every spot in the

Capital

**Use Case Name** Manipulate the main spots in the Capital

**Related Requirements** RequirementA.4

**Goal in Context** The Admin requests to manipulate the main spots in the Capital

**Preconditions** The Admin adds the main spots in the Capital

**Successful End Condition** The main spot in the Capital is manipulated

**Failed End Condition** The main spot in the Capital isn’t manipulated

**Primary Actors** Admin

**Secondary Actors** None

**Trigger** The Admin send a request to manipulate the main spots in the

Capital to the system

**Main Flow Step Action**

1. The Admin asks the system to manipulate the main

spots in the Capital

2. The system verifies the existence of the spots

3. The system manipulates the main spots in the Capital

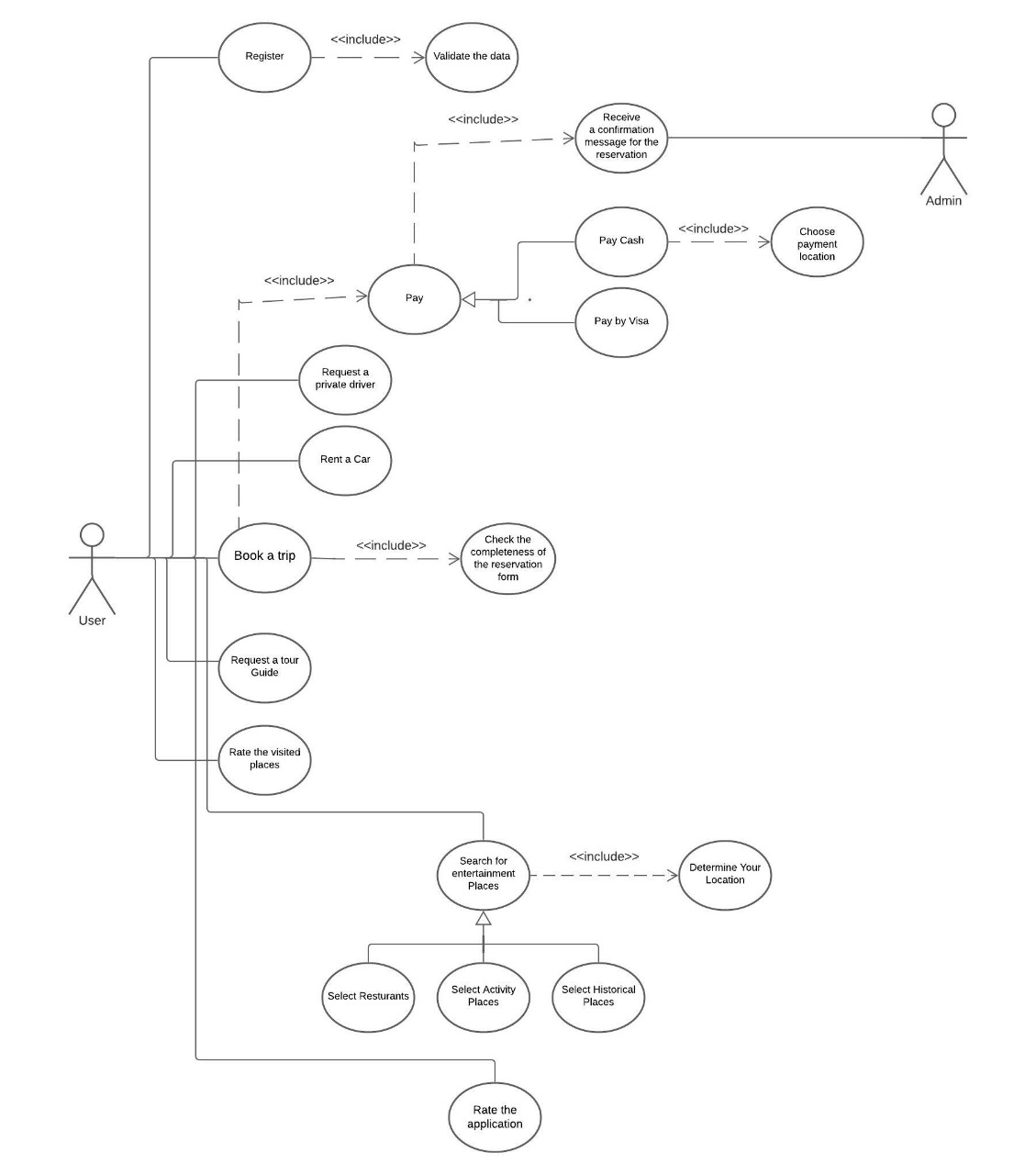
**Extensions Step Action**

* 1. The system isn’t able to find the spots to apply the

modification

* 1. The system doesn’t manipulate the main spots in the

Capital



**B**

Figure : Use Case diagram for User

**Use Case Name** Register

**Related Requirements** None

**Goal in Context** The User sends a request to register in the system

**Preconditions** None

**Successful End Condition** TheUser is registered in the system

**Failed End Condition** The User isn’t registered in the system

**Primary Actors** User

**Secondary Actors** None

**Trigger** The User asks the system to register

**Main Flow Step Action**

1. The User asks the system to register

2. The User sends the data of registration to the system

3. The system checks the validation of the registration

data

**Include :: Validate the data**

1. The User is registered in the system

**Extensions Step Action**

2.1 The system rejects the registration data because it

isn’t valid

2.2 The User isn’t registered in the system

**Use Case Name** Validate the Data

**Related Requirements** RequirementB.1

**Goal in Context** The User needs to be registered in the system

**Preconditions** The User is trying to register in the system

**Successful End Condition** The User registration data is valid

**Failed End Condition** The User registration data isn’t valid

**Primary Actors** User

**Secondary Actors** None

**Trigger** Check the validation of the User’s registration data while registering

in the system

**Main Flow Step Action**

1. The system receives the registration data of the User

2. The system validates the registration data

3. The system accepts the registration data

**Extensions Step Action**

* 1. The system finds that the registration data isn’t valid

2.2 The system rejects the registration data

**Use Case Name** Book a trip

**Related Requirements** None

**Goal in Context** The user needs to book a trip

**Preconditions** The user should be registered in the system

**Successful End Condition** The user has booked a trip

**Failed End Condition** The user hasn’t booked a trip

**Primary Actors** User

**Secondary Actors** None

**Trigger** The user asks the system to book a trip

**Main Flow Step Action**

1. The user sends a request to book a trip
2. The system checks the validation of this request
3. The user fills a form with his desirable trip

**Include :: Check the Completeness of filling the form**

1. The system receives the form with the user’s booked trip data

**Extensions Step Action**

* 1. The system checks that the request isn’t valid
  2. The system rejects the form
  3. The user doesn’t book a trip

**Use Case Name** Check the Completeness of filling the form

**Related Requirements** RequirementB.3

**Goal in Context** The user fills the form completely with all the needed data

**Preconditions** The user is trying to fill a form to book a trip

**Successful End Condition** The user form is valid

**Failed End Condition** The user form isn’t valid

**Primary Actors** User

**Secondary Actors** None

**Trigger** Check the Completeness of the user’s data while filling the form

**Main Flow Step Action**

1. The user fills the form with his desirable booked trip
2. The system checks if the form has been filled completely
3. The system accepts the form

**Extensions Step Action**

2.1 The system finds that the data in the form isn’t

complete

2.2 The system rejects the form

**Use Case Name** Pay

**Related Requirements** RequirementB.3

**Goal in Context** The user Pays the requirement payment

**Preconditions** The user fills a form to book a trip

**Successful End Condition** The user payment is accepted

**Failed End Condition** The user payment is rejected

**Primary Actors** User

**Secondary Actors** None

**Trigger** The user asks the system to pay the requirement payment

**Main Flow Step Action**

1. The user sends a request to pay the payment
2. The system checks the validation of this request
3. The system accepts the request of the user payment

**Extensions Step Action**

2.1 The system finds that the request isn’t valid

2.2 The system rejects the request of the user payment

**Use Case Name** Pay Cash

**Related Requirements** RequirementB.5

**Goal in Context** The user Pays the requirement payment Cash

**Preconditions** The user sends a request to pay the requirement payment

**Successful End Condition** The user payment is accepted

**Failed End Condition** The user payment is rejected

**Primary Actors** User

**Secondary Actors** None

**Trigger** The user asks the system to pay the requirement payment

**Main Flow Step Action**

1. The user sends a request to pay the payment Cash
2. The system checks the validation of this request
3. The user payment is accepted

**Include :: Receive a message with a location for the Payment**

**Extensions Step Action**

2.1 The system finds that the request isn’t valid

2.2 The user payment is rejected

**Use Case Name** Pay by Visa

**Related Requirements** RequirementB.5

**Goal in Context** The user Pays the requirement payment by Visa

**Preconditions** The user sends a request to pay the requirement payment

**Successful End Condition** The user payment is accepted

**Failed End Condition** The user payment is rejected

**Primary Actors** User

**Secondary Actors** None

**Trigger** The user asks the system to pay the requirement payment

**Main Flow Step Action**

1. The user sends a request to pay the payment by Visa
2. The system checks the validation of this request
3. The user payment is accepted

**Extensions Step Action**

2.1 The system finds that the request isn’t valid

2.2 The user payment is rejected

**Use Case Name** Receive a message with a location for the Payment

**Related Requirements** RequirementB.6

**Goal in Context** The user receives a message with a location for the Payment

**Preconditions** The user sends a request to pay the requirement payment Cash

**Successful End Condition** The user receives a message with a location for the cash payment

**Failed End Condition** The user doesn’t receive a message with a location for the cash

payment

**Primary Actors** User

**Secondary Actors** None

**Trigger** The user asks the system to pay the requirement payment

**Main Flow Step Action**

1. The user sends a request to pay the payment Cash
2. The system checks the validation of this request
3. The user receives a message with a location for the cash payment

**Extensions Step Action**

2.1 The system finds that the request isn’t valid

2.2 The user doesn’t receive a message with a location

for the cash payment

**Use Case Name** Receive a Confirmation message for the reservation

**Related Requirements** RequirementB.5

**Goal in Context** The user receives a confirmation message for the reservation

**Preconditions** The user sends a request to pay the requirement payment

**Successful End Condition** The user receives a confirmation message for the reservation

**Failed End Condition** The user doesn’t receive a confirmation message for the reservation

**Primary Actors** User

**Secondary Actors** Admin

**Trigger** The user needs to get a confirmation message for the reservation

**Main Flow Step Action**

1. The System sends the user’s reservation details to the Admin
2. The Admin checks the validation of the user reservation
3. The system sends a confirmation message for the reservation

**Extensions Step Action**

2.1 The Admin finds that the user reservation isn’t valid

2.2 The user doesn’t receive a confirmation message for

the reservation

**Use Case Name** Rent a Car

**Related Requirements** None

**Goal in Context** The user needs to rent a car

**Preconditions** The user should be registered in the system

**Successful End Condition** The user has rented a car

**Failed End Condition** The user doesn’t have a rented a car

**Primary Actors** User

**Secondary Actors** None

**Trigger** The user asks the system to rent a car

**Main Flow Step Action**

1. The user sends a request to rent a car
2. The system checks the validation of this request
3. The system accepts the request
4. The user has rented a car

**Extensions Step Action**

2.1 The system checks that the request isn’t valid

2.2 The system rejects the request

2.3 The user doesn’t have a rented a car

**Use Case Name** Request a Private Driver

**Related Requirements** RequirementB.10

**Goal in Context** The user needs to request a private driver

**Preconditions** The user has rented a car

**Successful End Condition** The user has a private driver to his rented car

**Failed End Condition** The user doesn’t have a private driver to his rented car

**Primary Actors** User

**Secondary Actors** None

**Trigger** The user asks the system to request a private driver

**Main Flow Step Action**

1. The user sends a request to request a private driver
2. The system checks the validation of this request
3. The system accepts the request
4. The user has a private driver to his rented car

**Extensions Step Action**

2.1 The system checks that the request isn’t valid

2.2 The system rejects the request

2.3 The user doesn’t have a private driver to his rented

car

**Use Case Name** Request a Tour Guide

**Related Requirements** RequirementB.3

**Goal in Context** The user needs to request a tour guide

**Preconditions** The user has booked a trip

**Successful End Condition** The user has a tour guide in his booked trip

**Failed End Condition** The user doesn’t have a tour guide in his booked trip

**Primary Actors** User

**Secondary Actors** None

**Trigger** The user asks the system to request a tour guide

**Main Flow Step Action**

1. The user sends a request to request a tour guide
2. The system checks the validation of this request
3. The system accepts the request
4. The user has a tour guide in his booked trip

**Extensions Step Action**

2.1 The system checks that the request isn’t valid

2.2 The system rejects the request

2.3 The user doesn’t have a tour guide in his booked trip

**Use Case Name** Rate the visited places

**Related Requirements** None

**Goal in Context** The user needs to rate the visited places

**Preconditions** The user has searched for entertainment places

**Successful End Condition** The user has rated the visited places

**Failed End Condition** The user hasn’t rated the visited places

**Primary Actors** User

**Secondary Actors** None

**Trigger** The user asks the system to rate the visited places

**Main Flow Step Action**

1. The user sends a request to rate the visited places
2. The system checks the validation of this request
3. The system accepts the request
4. The user rates the visited places
5. The system saves the rate

**Extensions Step Action**

2.1 The system checks that the request isn’t valid

2.2 The system rejects the request

2.3 The user hasn’t rated the visited places

**Use Case Name** Search for Entertainment Places

**Related Requirements** None

**Goal in Context** The user needs to Search for Entertainment Places

**Preconditions** The user should be registered in the system

**Successful End Condition** The user has searched for entertainment places

**Failed End Condition** The user hasn’t searched for entertainment places

**Primary Actors** User

**Secondary Actors** None

**Trigger** The user asks the system to search for entertainment places

**Main Flow Step Action**

1. The user sends a request to search for entertainment

places

1. The system checks the validation of this request

**Include :: Determine your location**

1. The user has searched for entertainment

**Extensions Step Action**

* 1. The system checks that the request isn’t valid

2.2 The user hasn’t searched for entertainment places

**Use Case Name** Select Restaurants

**Related Requirements** Requirement B.14

**Goal in Context** The user needs to Select Restaurants

**Preconditions** The user should have searched for an entertainment places

**Successful End Condition** The user has selected restaurants

**Failed End Condition** The user hasn’t selected restaurants

**Primary Actors** User

**Secondary Actors** None

**Trigger** The user asks the system to select restaurants

**Main Flow Step Action**

1. The user sends a request to select restaurants
2. The system checks the validation of this request
3. The user has selected restaurants

**Extensions Step Action**

* 1. The system checks that the request isn’t valid

2.2 The user hasn’t selected restaurants

**Use Case Name** Select Activity Places

**Related Requirements** Requirement B.14

**Goal in Context** The user needs to Select Activity Places

**Preconditions** The user should have searched for an entertainment places

**Successful End Condition** The user has selected Activity Places

**Failed End Condition** The user hasn’t selected Activity Places

**Primary Actors** User

**Secondary Actors** None

**Trigger** The user asks the system to select Activity Places

**Main Flow Step Action**

1. The user sends a request to select Activity Places
2. The system checks the validation of this request
3. The user has selected Activity Places

**Extensions Step Action**

* 1. The system checks that the request isn’t valid

2.2 The user hasn’t selected Activity Places

**Use Case Name** Select Historical Places

**Related Requirements** Requirement B.14

**Goal in Context** The user needs to Select Historical Places

**Preconditions** The user should have searched for an entertainment places

**Successful End Condition** The user has selected Historical Places

**Failed End Condition** The user hasn’t selected Historical Places

**Primary Actors** User

**Secondary Actors** None

**Trigger** The user asks the system to select Historical Places

**Main Flow Step Action**

1. The user sends a request to select Historical Places
2. The system checks the validation of this request
3. The user has selected Historical Places

**Extensions Step Action**

* 1. The system checks that the request isn’t valid

2.2 The user hasn’t selected Historical Places

**Use Case Name** Determine Your Location

**Related Requirements** Requirement B.14

**Goal in Context** The user has defined his location while searching for entertainment

places

**Preconditions** The user is searching for entertainment places

**Successful End Condition** The user defined his location

**Failed End Condition** The user didn’t define his location

**Primary Actors** User

**Secondary Actors** None

**Trigger** Define the location ofthe user while searching for entertainment

places

**Main Flow Step Action**

1. The user defines his location
2. The system verifies the existence of the location
3. The system allows the user to search for

entertainment places

**Extensions Step Action**

* 1. The system isn’t able to find the location that

requested by the user

* 1. The system doesn’t allow the user to search for

entertainment places

**Use Case Name** Rate the Application

**Related Requirements** None

**Goal in Context** The user needs to rate the application

**Preconditions** The user should be registered in the system

**Successful End Condition** The user has rated the application

**Failed End Condition** The user hasn’t rated the application

**Primary Actors** User

**Secondary Actors** None

**Trigger** The user asks the system to rate the application

**Main Flow Step Action**

1. The user sends a request to rate the application
2. The system checks the validation of this request
3. The system accepts the request
4. The user rates the application
5. The system saves the rate

**Extensions Step Action**

2.1 The system checks that the request isn’t valid

2.2 The system rejects the request

2.3 The user hasn’t rated the application

**C**

Diagram

Description automatically generated

Figure : Use Case diagram for Travel Agency

**Use Case Name** Register

**Related Requirements** None

**Goal in Context** The travel agency sends a request to register in the system

**Preconditions** None

**Successful End Condition** Thetravel agency is registered in the system

**Failed End Condition** The travel agency isn’t registered in the system

**Primary Actors** Travel Agency

**Secondary Actors** None

**Trigger** The travel agency asks the system to register

**Main Flow Step Action**

1. The travel agency asks the system to register

2. The travel agency sends the data of registration to the

system

3. The system checks the validation of the registration

data

**Include :: Validate the data**

4. The travel agency is registered in the system

**Extensions Step Action**

2.1 The system rejects the registration data because it

isn’t valid

2.2 The travel agency isn’t registered in the system

**Use Case Name** Validate the Data

**Related Requirements** RequirementC.1

**Goal in Context** The travel agency need to be registered in the system

**Preconditions** The travel agency is trying to register in the system

**Successful End Condition** The travel agency registration data is valid

**Failed End Condition** The travel agency registration data isn’t valid

**Primary Actors** Travel Agency

**Secondary Actors** None

**Trigger** Check the validation of the travel agency’s registration data while

registering in the system

**Main Flow Step Action**

1. The system receives the registration data of the

travel agency

2. The system validates the registration data

3. The system accepts the registration data

**Extensions Step Action**

* 1. The system finds that the registration data isn’t valid

2.2 The system rejects the registration data

**Use Case Name** Fill a Form with the travel agency data

**Related Requirements** None

**Goal in Context** The travel agency needs to fill a form with its data

**Preconditions** The travel agency should be registered in the system

**Successful End Condition** The travel agency fills a form with its data

**Failed End Condition** The travel agency doesn’t fill a form with its data

**Primary Actors** Travel Agency

**Secondary Actors** None

**Trigger** The travel agency asks the system to fill a form with its data

**Main Flow Step Action**

1. The travel agency sends a request to fill a form with

its data

1. The system checks the validation of this request
2. The travel agency adds its data to the form

**Include :: Check the Completeness of filling the form**

1. The system receives the form with the travel

agency’s data

**Extensions Step Action**

* 1. The system checks that the request isn’t valid
  2. The system rejects the form

2.3 The travel agency doesn’t fill a form with its data

**Use Case Name** Check the Completeness of filling the form

**Related Requirements** RequirementC.3

**Goal in Context** The travel agency fills the form completely with all the needed data

**Preconditions** The travel agency is trying to fill a form with its data

**Successful End Condition** The travel agency form is valid

**Failed End Condition** The travel agency form isn’t valid

**Primary Actors** Travel Agency

**Secondary Actors** None

**Trigger** Check the Completeness of the travel agency’s data while filling the

form

**Main Flow Step Action**

1. The travel agency fill the form with its data

1. The system check if the form has been filled

completely

1. The system accepts the form

**Extensions Step Action**

2.1 The system finds that the data in the form isn’t

complete

2.2 The system rejects the form

**Use Case Name** Pay

**Related Requirements** RequirementC.3

**Goal in Context** The travel agency Pays the requirement payment

**Preconditions** The travel agency fills a form with its data

**Successful End Condition** The travel agency payment is accepted

**Failed End Condition** The travel agency payment is rejected

**Primary Actors** Travel Agency

**Secondary Actors** None

**Trigger** The travel agency asks the system to pay the requirement payment

**Main Flow Step Action**

1. The travel agency sends a request to pay the

payment

1. The system checks the validation of this request
2. The system accepts the request of the travel agency

payment

**Extensions Step Action**

2.1 The system finds that the request isn’t valid

2.2 The system rejects the request of the travel agency

payment

**Use Case Name** Pay Cash

**Related Requirements** RequirementC.5

**Goal in Context** The travel agency Pays the requirement payment Cash

**Preconditions** The travel agency sends a request to pay the requirement payment

**Successful End Condition** The travel agency payment is accepted

**Failed End Condition** The travel agency payment is rejected

**Primary Actors** Travel Agency

**Secondary Actors** None

**Trigger** The travel agency asks the system to pay the requirement payment

**Main Flow Step Action**

1. The travel agency sends a request to pay the

payment Cash

1. The system checks the validation of this request
2. The travel agency payment is accepted

**Include :: Receive a message with a location for the Payment**

**Extensions Step Action**

2.1 The system finds that the request isn’t valid

2.2 The travel agency payment is rejected

**Use Case Name** Pay by Visa

**Related Requirements** RequirementC.5

**Goal in Context** The travel agency Pays the requirement payment by Visa

**Preconditions** The travel agency sends a request to pay the requirement payment

**Successful End Condition** The travel agency payment is accepted

**Failed End Condition** The travel agency payment is rejected

**Primary Actors** Travel Agency

**Secondary Actors** None

**Trigger** The travel agency asks the system to pay the requirement payment

**Main Flow Step Action**

1. The travel agency sends a request to pay the

payment by Visa

1. The system checks the validation of this request
2. The travel agency payment is accepted

**Extensions Step Action**

2.1 The system finds that the request isn’t valid

2.2 The travel agency payment is rejected

**Use Case Name** Receive a message with a location for the Payment

**Related Requirements** RequirementC.6

**Goal in Context** The travel agency receives a message with a location for the

Payment

**Preconditions** The travel agency sends a request to pay the requirement payment

Cash

**Successful End Condition** The travel agency receives a message with a location for the cash

payment

**Failed End Condition** The travel agency doesn’t receive a message with a location for the

cash payment

**Primary Actors** Travel Agency

**Secondary Actors** None

**Trigger** The travel agency asks the system to pay the requirement payment

**Main Flow Step Action**

1. The travel agency sends a request to pay the

payment Cash

1. The system checks the validation of this request
2. The travel agency receives a message with a location

for the cash payment

**Extensions Step Action**

2.1 The system finds that the request isn’t valid

2.2 The travel agency doesn’t receive a message with a

location for the cash payment

**Use Case Name** Receive a Confirmation message for the registration

**Related Requirements** RequirementC.5

**Goal in Context** The travel agency receives a confirmation message for the

registration

**Preconditions** The travel agency sends a request to pay the requirement payment

**Successful End Condition** The travel agency receives a confirmation message for the

registration

**Failed End Condition** The travel agency doesn’t receive a confirmation message for the

registration

**Primary Actors** Travel Agency

**Secondary Actors** Admin

**Trigger** The travel agency needs to get a confirmation message for the

registration

**Main Flow Step Action**

1. The System sends the travel agency’s registration

details to the Admin

1. The Admin checks the validation of the travel

agency registration

1. The system sends a confirmation message for the

registration

**Extensions Step Action**

2.1 The Admin finds that the travel agency registration

isn’t valid

* 1. The travel agency doesn’t receive a confirmation

message for the registration

**Use Case Name** Manipulate the travel agency’s Data

**Related Requirements** None

**Goal in Context** The travel agency requests to manipulate its data

**Preconditions** The travel agency fills a form with its data

**Successful End Condition** The travel agency’s Data is manipulated

**Failed End Condition** The travel agency’s Data isn’t manipulated

**Primary Actors** Travel Agency

**Secondary Actors** None

**Trigger** The Travel Agency sends a request to manipulate its data

**Main Flow Step Action**

1. The Travel Agency asks the system to manipulate its

data

2. The system verifies the existence of the data

3. The system manipulates the data of the travel agency

**Extensions Step Action**

2.1 The system isn’t able to find the travel agency’s data

to apply the modification

2.2 The system doesn’t manipulate the travel agency data

**Use Case Name** Cancel yourRegistration

**Related Requirements** None

**Goal in Context** The travel agency sends a request to cancel its registration in the

system

**Preconditions** The travel agency should be registered in the system

**Successful End Condition** Thetravel agency isn’t registered in the system

**Failed End Condition** The travel agency still registered in the system

**Primary Actors** Travel Agency

**Secondary Actors** None

**Trigger** The travel agency asks the system to cancel its registration

**Main Flow Step Action**

1. The travel agency asks the system to cancel its

register

1. The system checks the existence of the travel agency

registration

1. The travel agency isn’t registered in the system

**Include :: Receive a Confirmation message**

**Extensions Step Action**

2.1 The system rejects the request because the travel

agency registration doesn’t exist

2.2 The travel agency still registered in the system

**Use Case Name** Fill a Form with the offered trips

**Related Requirements** None

**Goal in Context** The travel agency needs to fill a form with the offered trips

**Preconditions** The travel agency should be registered in the system

**Successful End Condition** The travel agency fills a form with the offered trips

**Failed End Condition** The travel agency doesn’t fill a form with the offered trips

**Primary Actors** Travel Agency

**Secondary Actors** None

**Trigger** The travel agency asks the system to fill a form with the offered

trips

**Main Flow Step Action**

1. The travel agency sends a request to fill a form with

the offered trips

1. The system checks the validation of this request

3. The travel agency adds the offered trips to the form

4. The system receives the form with the offered trips

**Include :: Receive a confirmation message**

**Extensions Step Action**

2.1 The system checks that the request isn’t valid

* 1. The system rejects the form
  2. The travel agency doesn’t fill a form with the offered

trips

**Use Case Name** Manipulate the offered trips’ Data

**Related Requirements** None

**Goal in Context** The travel agency requests to manipulate the offered trips’ data

**Preconditions** The travel agency fills a form with the offered trips data

**Successful End Condition** The offered trips’ Data is manipulated

**Failed End Condition** The offered trips’ Data isn’t manipulated

**Primary Actors** Travel Agency

**Secondary Actors** None

**Trigger** The Travel Agency sends a request to manipulate the offered trips’

data

**Main Flow Step Action**

1. The Travel Agency asks the system to manipulate the

offered trips’ data

2. The system verifies the existence of the data

3. The system manipulates the data of the offered trips

**Extensions Step Action**

* 1. The system isn’t able to find the offered trips’ data to

apply the modification

2.2 The system doesn’t manipulate the offered trips data

**Use Case Name** Receive a Confirmation message

**Related Requirements** RequirementC.11, RequirementC.12, RequirementC.13

**Goal in Context** The travel agency receives a Confirmation message

**Preconditions** The travel agency sends a request to the system

**Successful End Condition** The travel agency receives a Confirmation message

**Failed End Condition** The travel agency doesn’t receive a Confirmation message

**Primary Actors** Travel Agency

**Secondary Actors** Admin

**Trigger** The travel agency asks the system for a Confirmation message

**Main Flow Step Action**

1. The travel agency sends a request to the system
2. The system checks the validation of this request
3. The travel agency receives a Confirmation message

**Extensions Step Action**

2.1 The system finds that the request isn’t valid

2.2 The travel agency doesn’t receive a Confirmation

message

Diagram

Description automatically generated

Figure : Use Case diagram for Car Rental Agency

**D**

**Use Case Name** Register

**Related Requirements** None

**Goal in Context** The car rental agency sends a request to register in the system

**Preconditions** None

**Successful End Condition** Thecar rental agency is registered in the system

**Failed End Condition** The car rental agency isn’t registered in the system

**Primary Actors** Car Rental Agency

**Secondary Actors** None

**Trigger** The car rental agency asks the system to register

**Main Flow Step Action**

1. The car rental agency asks the system to register

1. The car rental agency sends the data of registration

to the system

1. The system checks the validation of the registration

data

**Include :: Validate the data**

1. The car rental agency is registered in the system

**Extensions Step Action**

2.1 The system rejects the registration data because it

isn’t valid

2.2 The car rental agency isn’t registered in the system

**Use Case Name** Validate the Data

**Related Requirements** RequirementD.1

**Goal in Context** The car rental agency need to be registered in the system

**Preconditions** The car rental agency is trying to register in the system

**Successful End Condition** The car rental agency registration data is valid

**Failed End Condition** The car rental agency registration data isn’t valid

**Primary Actors** Car Rental Agency

**Secondary Actors** None

**Trigger** Check the validation of the car rental agency’s registration data

while registering in the system

**Main Flow Step Action**

1. The system receives the registration data of the car

rental agency

1. The system validates the registration data
2. The system accepts the registration data

**Extensions Step Action**

* 1. The system finds that the registration data isn’t valid

2.2 The system rejects the registration data

**Use Case Name** Fill a Form with the car rental agency data

**Related Requirements** None

**Goal in Context** The car rental agency needs to fill a form with its data

**Preconditions** The car rental agency should be registered in the system

**Successful End Condition** The car rental agency fills a form with its data

**Failed End Condition** The car rental agency doesn’t fill a form with its data

**Primary Actors** Car Rental Agency

**Secondary Actors** None

**Trigger** The car rental agency asks the system to fill a form with its data

**Main Flow Step Action**

1. The car rental agency sends a request to fill a form

with its data

1. The system checks the validation of this request
2. The car rental agency adds its data to the form

**Include :: Check the Completeness of filling the form**

1. The system receives the form with the car rental

agency’s data

**Extensions Step Action**

* 1. The system checks that the request isn’t valid

2.2 The system rejects the form

* 1. The car rental agency doesn’t fill a form with its data

**Use Case Name** Check the Completeness of filling the form

**Related Requirements** RequirementD.3

**Goal in Context** The car rental agency fills the form completely with all the needed

data

**Preconditions** The car rental agency is trying to fill a form with its data

**Successful End Condition** The car rental agency form is valid

**Failed End Condition** The car rental agency form isn’t valid

**Primary Actors** Car Rental Agency

**Secondary Actors** None

**Trigger** Check the Completeness of the car rental agency’s data while filling

the form

**Main Flow Step Action**

1. The car rental agency fills the form with its data

1. The system check if the form has been filled

completely

1. The system accepts the form

**Extensions Step Action**

2.1 The system finds that the data in the form isn’t

complete

2.2 The system rejects the form

**Use Case Name** Pay

**Related Requirements** RequirementD.3

**Goal in Context** The car rental agency Pays the requirement payment

**Preconditions** The car rental agency fills a form with its data

**Successful End Condition** The car rental agency payment is accepted

**Failed End Condition** The car rental agency payment is rejected

**Primary Actors** Car Rental Agency

**Secondary Actors** None

**Trigger** The car rental agency asks the system to pay the requirement

payment

**Main Flow Step Action**

1. The car rental agency sends a request to pay the

payment

1. The system checks the validation of this request
2. The system accepts the request of the car rental

agency payment

**Extensions Step Action**

2.1 The system finds that the request isn’t valid

2.2 The system rejects the request of the car rental

agency payment

**Use Case Name** Pay Cash

**Related Requirements** RequirementD.5

**Goal in Context** The car rental agency Pays the requirement payment Cash

**Preconditions** The car rental agency sends a request to pay the requirement

payment

**Successful End Condition** The car rental agency payment is accepted

**Failed End Condition** The car rental agency payment is rejected

**Primary Actors** Car Rental Agency

**Secondary Actors** None

**Trigger** The car rental agency asks the system to pay the requirement

payment

**Main Flow Step Action**

1. The car rental agency sends a request to pay the

payment Cash

1. The system checks the validation of this request
2. The car rental agency payment is accepted

**Include :: Receive a message with a location for the Payment**

**Extensions Step Action**

2.1 The system finds that the request isn’t valid

2.2 The car rental agency payment is rejected

**Use Case Name** Pay by Visa

**Related Requirements** RequirementD.5

**Goal in Context** The car rental agency Pays the requirement payment by Visa

**Preconditions** The car rental agency sends a request to pay the requirement

payment

**Successful End Condition** The car rental agency payment is accepted

**Failed End Condition** The car rental agency payment is rejected

**Primary Actors** Car Rental Agency

**Secondary Actors** None

**Trigger** The car rental agency asks the system to pay the requirement

payment

**Main Flow Step Action**

1. The car rental agency sends a request to pay the

payment by Visa

1. The system checks the validation of this request
2. The car rental agency payment is accepted

**Extensions Step Action**

2.1 The system finds that the request isn’t valid

2.2 The car rental agency payment is rejected

**Use Case Name** Receive a message with a location for the Payment

**Related Requirements** RequirementD.6

**Goal in Context** The car rental agency receives a message with a location for the

Payment

**Preconditions** The car rental agency sends a request to pay the requirement

payment Cash

**Successful End Condition** The car rental agency receives a message with a location for the

cash payment

**Failed End Condition** The car rental agency doesn’t receive a message with a location for

the cash payment

**Primary Actors** Car Rental Agency

**Secondary Actors** None

**Trigger** The car rental agency asks the system to pay the requirement

payment

**Main Flow Step Action**

1. The car rental agency sends a request to pay the

payment Cash

1. The system checks the validation of this request
2. The car rental agency receives a message with a

location for the cash payment

**Extensions Step Action**

2.1 The system finds that the request isn’t valid

2.2 The car rental agency doesn’t receive a message with

a location for the cash payment

**Use Case Name** Receive a Confirmation message for the registration

**Related Requirements** RequirementD.5

**Goal in Context** The car rental agency receives a confirmation message for the

registration

**Preconditions** The car rental agency sends a request to pay the requirement

payment

**Successful End Condition** The car rental agency receives a confirmation message for the

registration

**Failed End Condition** The car rental agency doesn’t receive a confirmation message for

the registration

**Primary Actors** Car Rental Agency

**Secondary Actors** Admin

**Trigger** The car rental agency needs to get a confirmation message for the

registration

**Main Flow Step Action**

1. The System sends the car rental agency’s registration

details to the Admin

1. The Admin checks the validation of the car rental

agency registration

1. The system sends a confirmation message for the registration

**Extensions Step Action**

2.1 The Admin finds that the car rental agency

registration isn’t valid

* 1. The car rental agency doesn’t receive a confirmation

message for the registration

**Use Case Name** Manipulate the car rental agency’s Data

**Related Requirements** None

**Goal in Context** The car rental agency requests to manipulate its data

**Preconditions** The car rental agency fills a form with its data

**Successful End Condition** The car rental agency’s Data is manipulated

**Failed End Condition** The car rental agency’s Data isn’t manipulated

**Primary Actors** Car Rental Agency

**Secondary Actors** None

**Trigger** The car rental Agency sends a request to manipulate its data

**Main Flow Step Action**

1. The car rental Agency asks the system to manipulate

its data

2. The system verifies the existence of the data

3. The system manipulates the data of the car rental

agency

**Extensions Step Action**

* 1. The system isn’t able to find the car rental agency’s

data to apply the modification

* 1. The system doesn’t manipulate the car rental agency

data

**Use Case Name** Cancel yourRegistration

**Related Requirements** None

**Goal in Context** The car rental agency sends a request to cancel its registration in

the system

**Preconditions** The car rental agency should be registered in the system

**Successful End Condition** Thecar rental agency isn’t registered in the system

**Failed End Condition** The car rental agency still registered in the system

**Primary Actors** Car Rental Agency

**Secondary Actors** None

**Trigger** The car rental agency asks the system to cancel its registration

**Main Flow Step Action**

1. The car rental agency asks the system to cancel its

register

1. The system checks the existence of the car rental

agency registration

1. The car rental agency isn’t registered in the system

**Include :: Receive a Confirmation message**

**Extensions Step Action**

2.1 The system rejects the request because the car rental

agency registration doesn’t exist

2.2 The car rental agency still registered in the system

**Use Case Name** Fill a Form with the offered cars data

**Related Requirements** None

**Goal in Context** The car rental agency needs to fill a form with the offered cars data

**Preconditions** The car rental agency should be registered in the system

**Successful End Condition** The car rental agency fills a form with the offered cars data

**Failed End Condition** The car rental agency doesn’t fill a form with the offered cars data

**Primary Actors** Car Rental Agency

**Secondary Actors** None

**Trigger** The car rental agency asks the system to fill a form with the offered

cars data

**Main Flow Step Action**

1. The car rental agency sends a request to fill a form

with the offered cars data

1. The system checks the validation of this request
2. The car rental agency adds the offered cars data to

the form

1. The system receives the form with the offered cars

data

**Include :: Receive a confirmation message Include :: Determine the availability of requesting a private driver**

**Extensions Step Action**

* 1. The system checks that the request isn’t valid
  2. The system rejects the form
  3. The car rental agency doesn’t fill a form with the

offered cars data

**Use Case Name** Manipulate the offered cars’ Data

**Related Requirements** None

**Goal in Context** The car rental agency requests to manipulate the offered cars’ data

**Preconditions** The car rental agency fills a form with the offered cars Data

**Successful End Condition** The offered cars’ Data is manipulated

**Failed End Condition** The offered cars’ Data isn’t manipulated

**Primary Actors** Car Rental Agency

**Secondary Actors** None

**Trigger** The car rental Agency sends a request to manipulate the offered

cars’ data

**Main Flow Step Action**

1. The car rental Agency asks the system to manipulate

the offered cars’ data

2. The system verifies the existence of the data

3. The system manipulates the data of the offered cars

**Extensions Step Action**

* 1. The system isn’t able to find the offered cars’ data to

apply the modification

2.2 The system doesn’t manipulate the offered cars data

**Use Case Name** Receive a Confirmation message

**Related Requirements** RequirementD.11, RequirementD.12, RequirementD.13

**Goal in Context** The car rental agency receives a Confirmation message

**Preconditions** The car rental agency sends a request to the system

**Successful End Condition** The car rental agency receives a Confirmation message

**Failed End Condition** The car rental agency doesn’t receive a Confirmation message

**Primary Actors** Car Rental Agency

**Secondary Actors** Admin

**Trigger** The car rental agency asks the system for a Confirmation message

**Main Flow Step Action**

1. The car rental agency sends a request to the system

1. The system checks the validation of this request
2. The car rental agency receives a Confirmation

message

**Extensions Step Action**

2.1 The system finds that the request isn’t valid

2.2 The car rental agency doesn’t receive a Confirmation

message

**Use Case Name** Determine the availability of requesting a private driver

**Related Requirements** RequirementD.12

**Goal in Context** The car rental agency needs to determine the availability of

requesting a private driver

**Preconditions** The car rental agency is trying to fill a form with the offered cars

data

**Successful End Condition** The offered cars have the availability of requesting a private driver

**Failed End Condition** The offered cars don’t have the availability of requesting a private

driver

**Primary Actors** Car Rental Agency

**Secondary Actors** None

**Trigger** Check the availability of requesting a private driver for the offered

cars

**Main Flow Step Action**

1. The car rental agency checks the availability of

requesting a private driver

1. The car rental agency confirms the availability of

requesting a private driver

1. The system saves the received information

**Extensions Step Action**

2.1 The car rental agency cancels the availability of

requesting a private driver

# **Diagram Description automatically generated5.0 Swimlane Diagrams of all use cases**

Figure : Swimlane diagram for Travel Agency

Diagram

Description automatically generated

Figure : Swimlane diagram for Car Rental Agency

Diagram

Description automatically generated

Figure : Swimlane diagram for User

# **6.0 Noun Extraction and CRC Cards**

## 6.1 Noun Extraction:

**Stage 1: Concise Problem Definition**

Quest will help its users find restaurants, activity places, historical places according to your preference of place, and also will be their guide when they want to go on a trip all over Egypt.

**Stage 2: Informal Strategy**

Quest will help its users find restaurants, activity places, historical places according to your preference of place, and also will be their guide when they want to go on a trip all over Egypt through travel agencies. Quest will provide the user with all the main entertainment places in every spot in the capital , whether they want a dine-in restaurant , a place to do an activity such as escape rooms in the area you chose. The website will help local businesses to be known much more in the country; it works as a marketing agency for local businesses and fun-saver to its users. Quest will provide its users with trips all over the governorates of the country, giving them all the information, they need about hotels, activities, and the historical places to visit while going on a trip. Also, it’s the best for foreigners coming to Egypt, as they will be able to request a tour guide to aid them while going on a trip or visiting historical places of Egypt to make their experience easy, enjoyable, comfortable and full of fun. Also, the user doesn’t need to worry about the transportation to go on a trip ,wherever the place is; Quest will provide all means of transportation whatever you’re a group of people or only one person. Also, car renting agencies will offer the website some promotions and discounts for its users and so the users they can request to rent a private car as a way of transportation to go wherever they want to in Egypt that they might not be able to reach easily with other transportation means. To pay the services the user can choose between paying by Credit or Debit Card or Cash through Fawry/Vodafone Cash

**Stage 3: Formalize the Strategy**

The nouns that can be identified are:

Users,restaurants,activity places,historical places,guide,trip,entertainment places,capital,dine in,local businesses,governates,hotels,activities,foreigners,comfortable ,transportation,car renting agencies,promotions,doiscounts,credit,debit,cash,travel agencies.

Exclude nouns that lie outside the problem boundary:

Governates,foreigners,local businesses.

Exclude abstract nouns that represent ideas or quantities that have no physical existence:

Comfortable,promotions,discounts

Candidate Classes are:

User,Entertainment places,admin,travel agency,car agency,trip,payment,car,capital

## 6.2 CRC Cards:

|  |
| --- |
| User |
| Register in the Application  Send message to Payment to get Payment Info  Send message to Payment to select payment method (cash-visa)  Send message to payment to show required payments  Send message to Car Agency to Request a driver  Send message to Car Agency to Rent a car  Send message to Car to get Cars Info  Send message to Car to Show available Cars  Send message to Travel Agency to Book a trip  Send message to Travel Agency to Request a tour guide  Send message to Entertainment Place to Rate Visited Places  Send message to Entertainment Place to Search for Entertainment Places  Send message to Entertainment Place to Select Restaurants  Send message to Entertainment Place to Select Activity Places  Send message to Entertainment Place to Select Historical Places  Rate the Application  Get Requested Cars  Get booked trips  Send message to trip to get trips info  Send message to trip to show available trips  Send message to Capital Spot to Search for spots in the capital |
| Payment – Car Agency – Car – Travel Agency – Entertainment places – Trip – Capital Spot |

|  |
| --- |
| Admin |
| Send message to entertainment place to Add entertainment places  Send message to entertainment place to Manipulate entertainment places data  Send message to payment to show recorded payments  Show list of users  Confirm travel agency’s registration  Confirm car agency’s registration  Send message to Capital Spot to add spots  Send message to Capital Spot manipulate spots data |
| Entertainment Place – Payment – Capital Spot |

|  |
| --- |
| Payment |
| Get Payment Info  Show required payments  Select payment method (cash-visa)  Record payments  Show recorded payments |
|  |

|  |
| --- |
| Trip |
| Get trips Info  Show available trips  Add trips  Manipulate trips data |
|  |

|  |
| --- |
| Travel Agency |
| Book a trip  Record tour guide Requests  Request a tour guide  Send message to user to get booked trips  Send message to trip to add trips  Send message to trip to manipulate trips data  Register in the Application  Send message to Payment to get Payment Info  Send message to Payment to select payment method (cash-visa)  Cancel the registration |
| Trip – Payment – User |

|  |
| --- |
| Car Agency |
| Request a driver  Rent a car  Record driver requests  Send message to user to get requested cars  Send message to Car to add cars  Send message to Car to manipulate cars data  Register in the Application  Send message to Payment to get Payment Info  Send message to Payment to select payment method (cash-visa)  Cancel the registration |
| Car – Payment – User |

|  |
| --- |
| Car |
| Get Cars Info  Show available Cars  Add cars  Manipulate cars data |
|  |

|  |
| --- |
| Entertainment Place |
| Rate Visited Places  Search for Entertainment Places  Select Restaurants  Select Activity Places  Select Historical Places  Add entertainment places  Manipulate entertainment places data |
|  |

|  |
| --- |
| Capital Spot |
| Search for spots in the capital  Add spots  Manipulate spots data |
|  |

# **7.0 Class Model**

**Steps for the Class Model:**

After finishing the Client object diagram, we will follow number of iterations to get our detailed class diagram:

**First Iteration**: as shown in figure [8] we have written down our main classes and their needed attributes

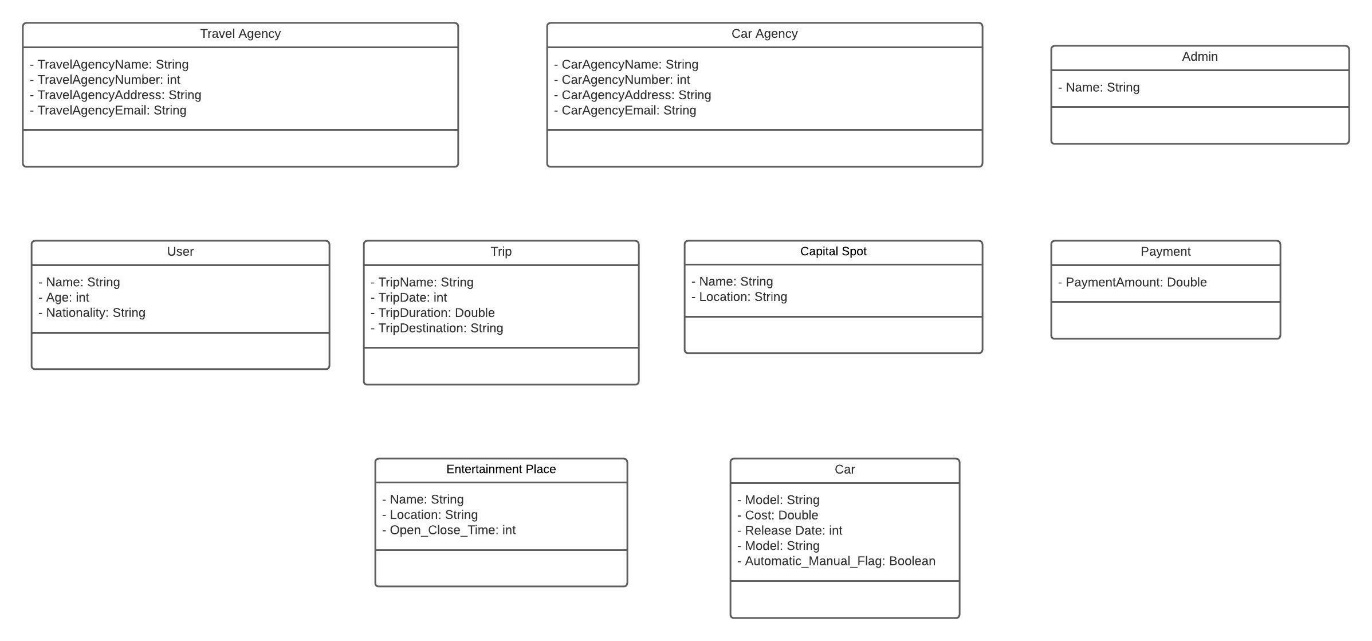


Figure 8: Iteration 1 in Class diagram

**Second Iteration**: as shown in figure [9] we have found that we need to inherit some classes; we will inherit Pay\_by\_Visa and Pay\_Cash from Payment Class, also we will inherit Restaurant, Activity Place, and Historical Place. Moreover, we will add the attributes for the new added classes in this iteration.

**Diagram

Description automatically generated**

Figure 9: Iteration 2 in Class diagram

**Third Iteration**: as shown in figure [10], in this iteration we have added the relations between our classes

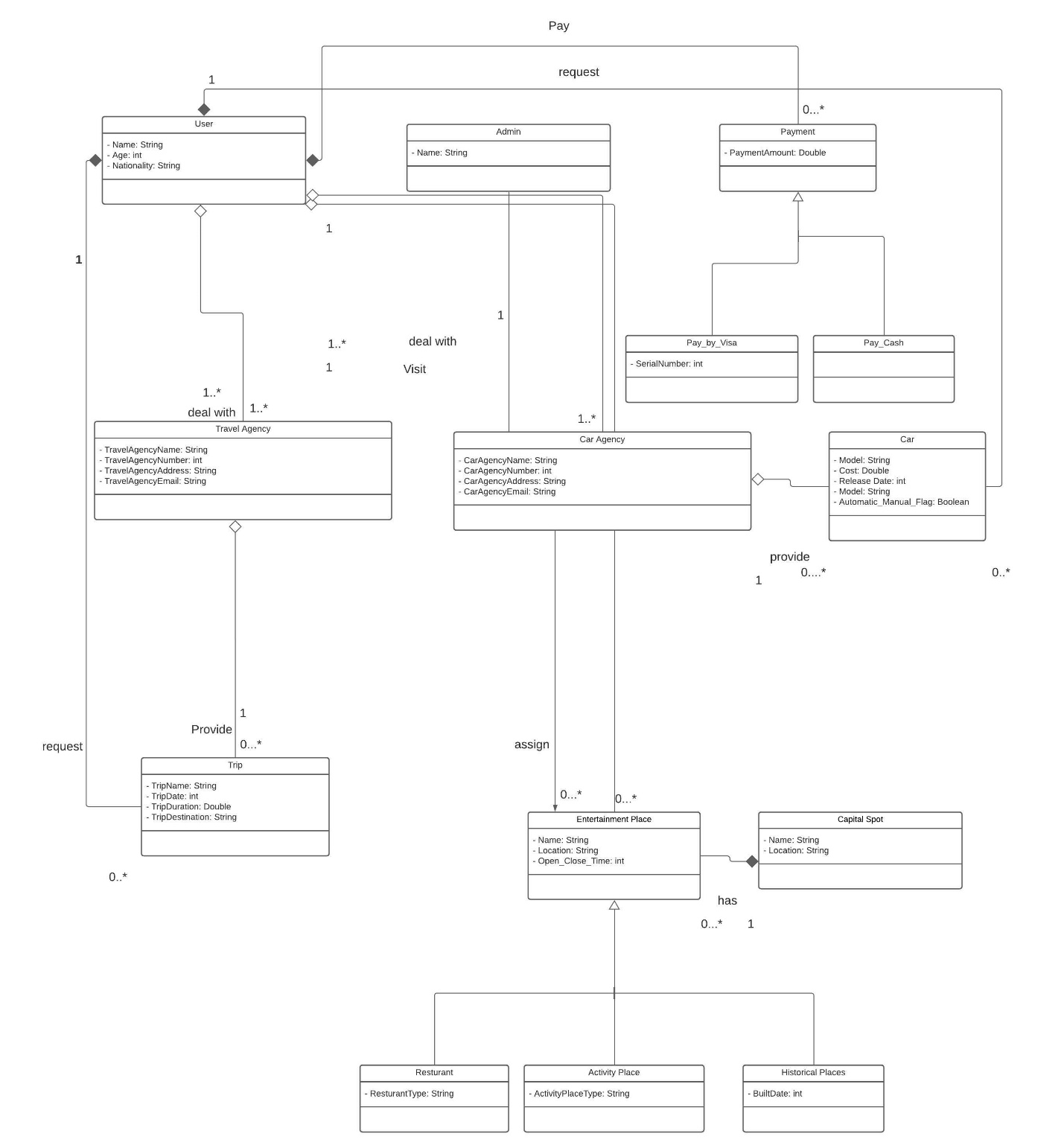


Figure 10: Iteration 3 in Class Diagram

**Fourth Iteration**: In this iteration we have added the operations (functions) for each class in our class diagram, and this is the final iteration for our class diagram figure [11].

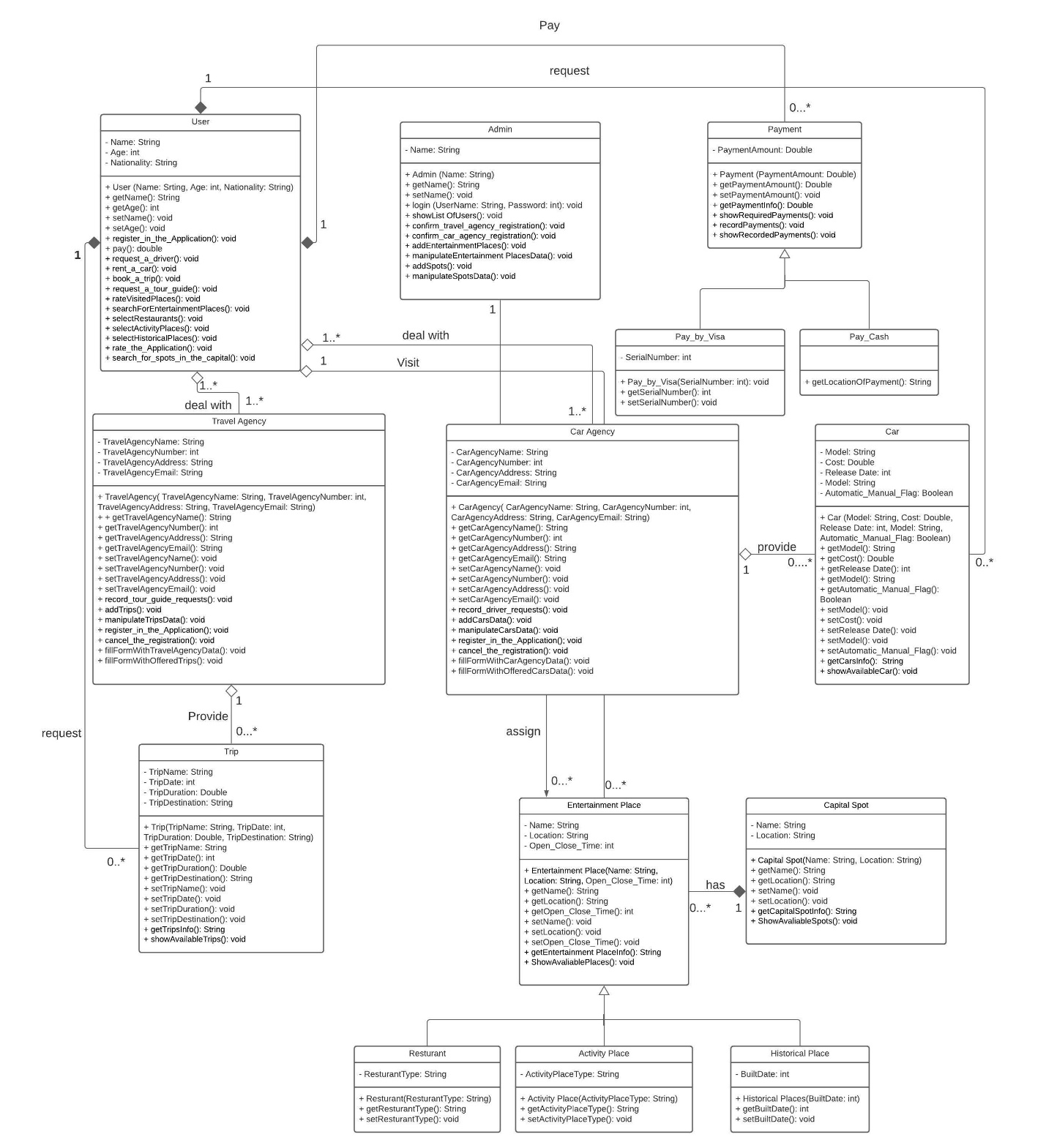


Figure 11: Iteration 4 in Class diagram

# **Chart, box and whisker chart Description automatically generated8.0 State Diagram**

Figure 12: State Diagram

# **9.0 Interaction Diagrams**

User requests a trip from Gui then the gui retrieving the data from the customer controller API in which it calls the Data base. Gui can prompt the user to fill in the form of the customer information. User can request a tour guide or trip via the GUI and submit a trip review.

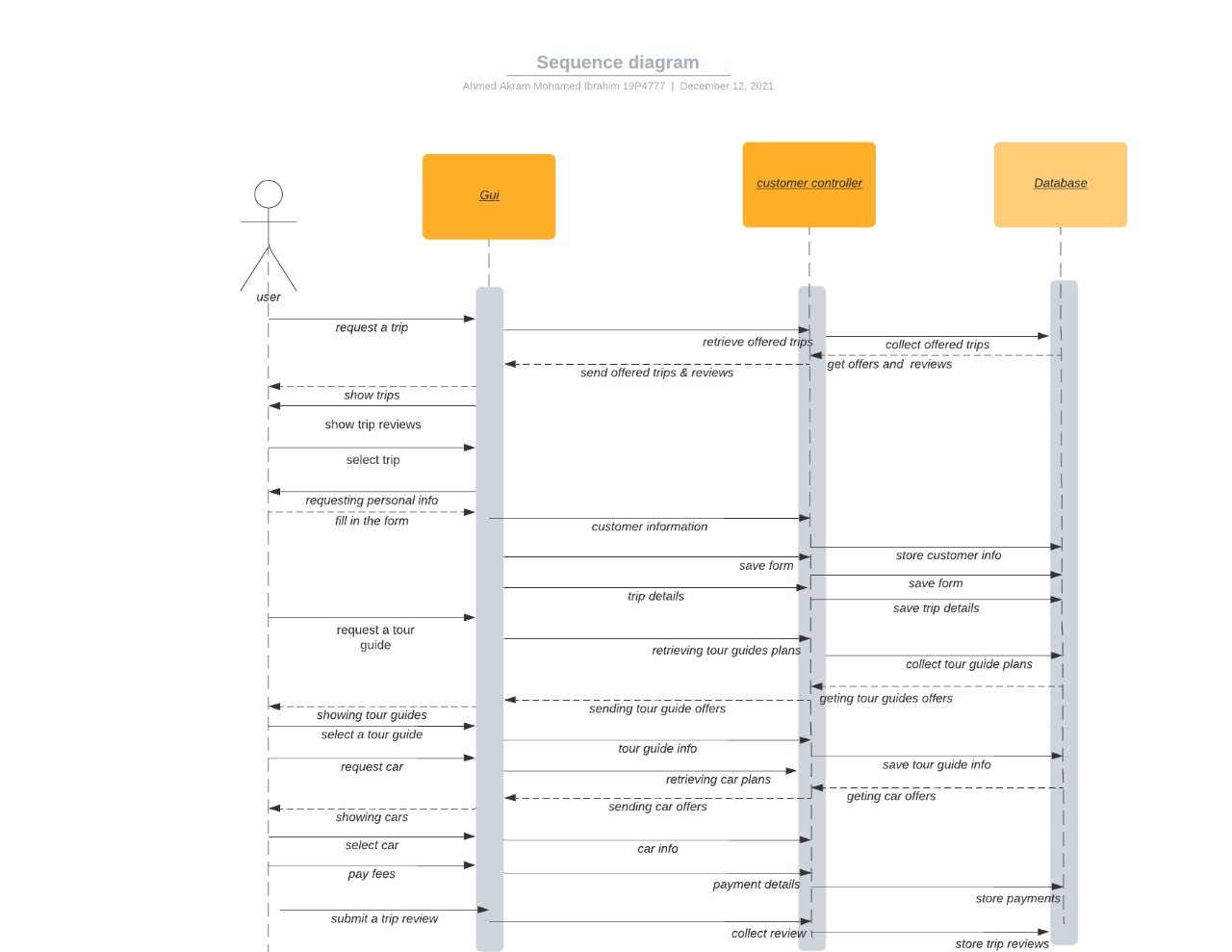


Figure 13: Sequence diagram for user

Admin can request the list of registration via the gui to be retrieved by the travel manager API. Admin can accept and confirm registration and deals. Admin can manipulate the places of entertainment in the city via the GUI

Timeline

Description automatically generated

Figure 14: sequence diagram for Admin

# **10.0 Class Diagram**

Diagram

Description automatically generated

Figure 15: Class diagram

# **11.0 Client-Object Relation Diagram**



Figure 16: Client Object diagram

# **12.0 OOAD** **Methodologies**

## 12.1 Jacobson et al. Methodologies

In Jacobson methodology it totally relies on describing the whole project into scenarios, and then each scenario is then converted to its equivalent conventional use case where it is some way different than today’s UML use case diagrams.

• Use case is used to identify system requirements

• Use case shows interactions between user and the system

• Use case captures the system responsibility to the user

• An Abstract use case is not complete and has no actors that initiated it, but it

is used by another use case.

• inheritance could be used in several levels.

• The user’s relationship reuses common behavior in different use cases

• abstract use cases also are the ones that have uses or extends relationship.

#### **12.1.1 Object oriented analysis for Jacobson et al. Methodologies**

1. Use case modelling is used as mentioned before no extend or include used.

Normal scenarios and exception scenarios are made.

2. Class modeling breaking down the system into classes with their attributes.

In this section CRC cards are not used and replace by Use cases

3. Dynamic modeling showing action by each class also used by the use case

showing event flow.

#### **12.1.2 Object oriented design for Jacobson et al. Methodologies**

1. Interaction diagrams is implemented for scenario by using sequence diagrams or collaboration
2. Detailed class diagram is used after implementing CRC cards. An object which sends message to object x is called client of object x.
3. Client of object is used, an object which is not a client of another object must be initiated.
4. Implementing the code by using PDL

## 12.2 Object modelling Technique OMT

**Analysis** -The results are objects and dynamic and functional models

**System Design** - The result is a structure of the basic architecture of the system. Object Design - This phase produces a design document, consisting of detailed objects and dynamic and functional models Implementation

**OMT models**

1.Object model focuses

• Structure of objects in the system

• Identity of relationships to other object, attributes and

operations

• Object diagram/Class Diagram in section 10

• Interconnection between classes by association line

2. Dynamic involves states, events, and state diagrams on the model: State Diagram

in section 8

3. Functional model shows how data flows through Data-Flow Diagram

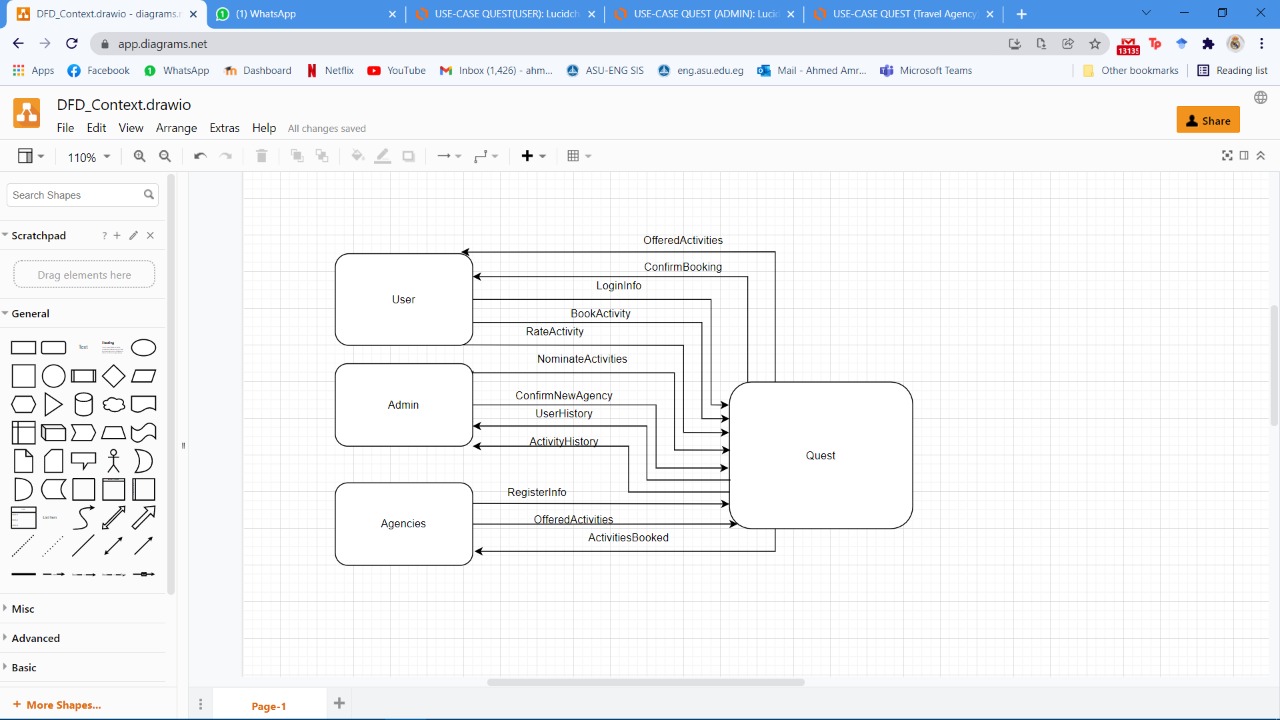


Figure 17: Level 0 in DFD diagram

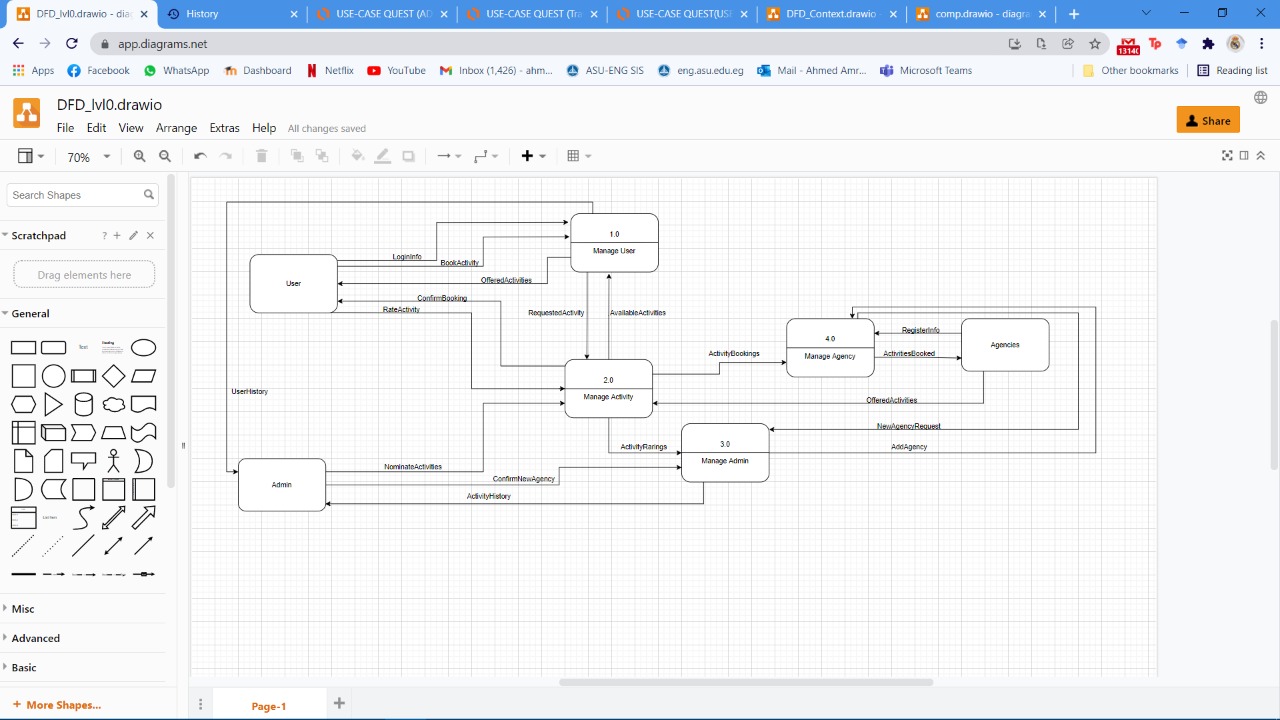


Figure 18: Level 1 in DFD diagram

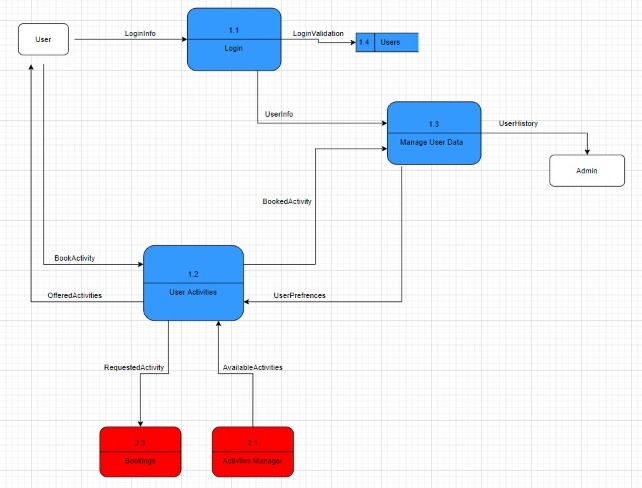


Figure 19: Level 2 for (1.0 Manager User) Component

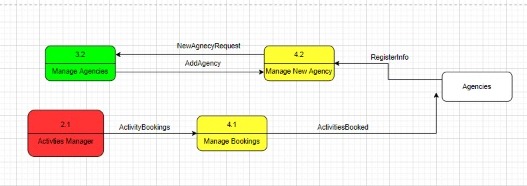
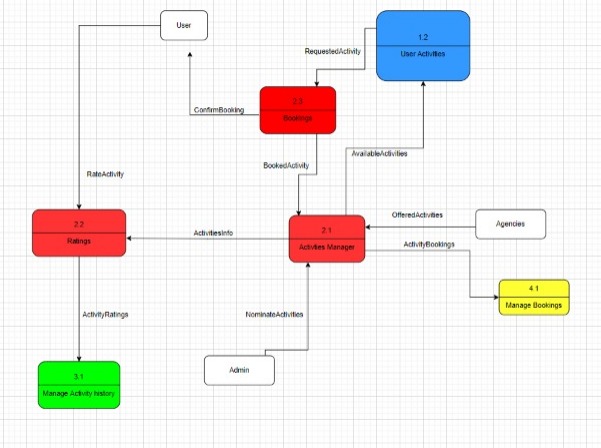


Figure 20: Level 2 for (2.0 Manage Activity) Component

Figure 21: Level 2 for (Manage Agency) Component

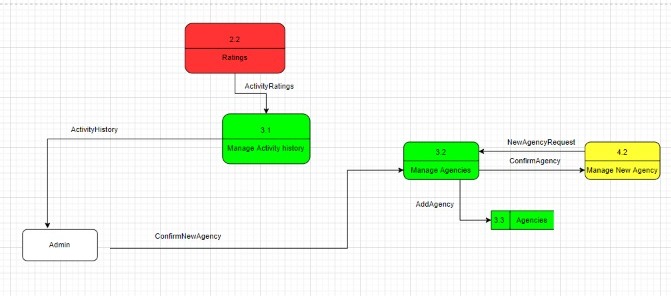


Figure 22: Level 2 (Manage Admin) Component

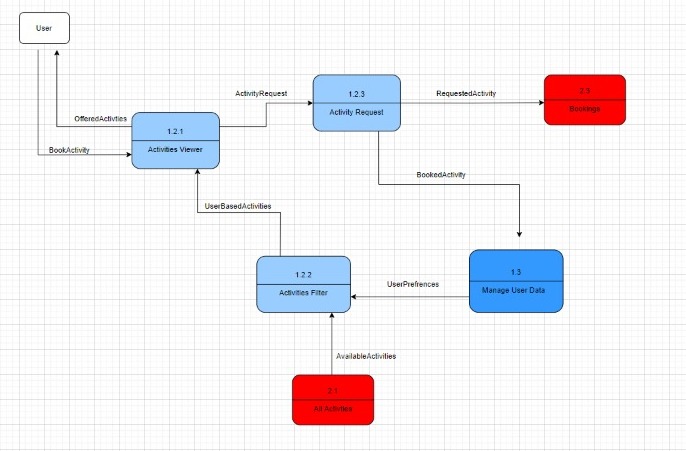


Figure 23: Level 3 for (1.2 User Activities) Component

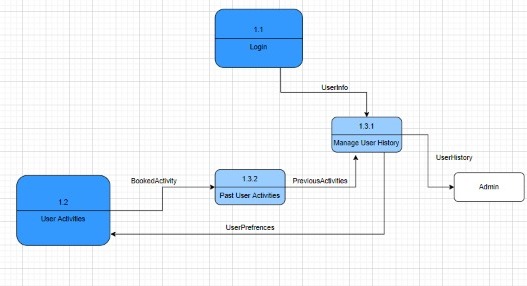


Figure 24: Level 3 for (1.3 Manage User data) Component

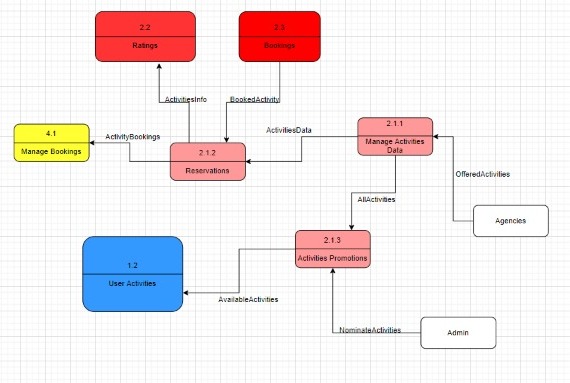


Figure 25: Level 3 (Activities Manager) Component

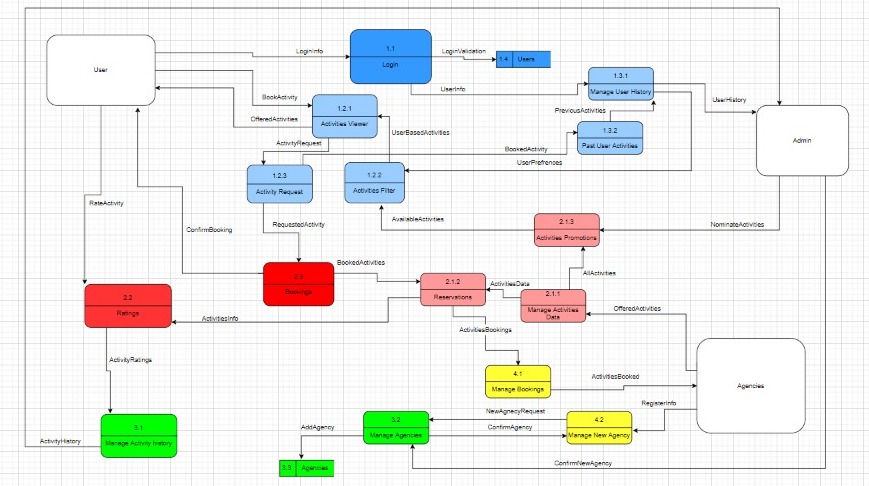


Figure 26: DFD Diagram

**Phases of OMT**

1. Analysis phase that requires breaking down the system into models of the OMT

models (object, dynamic, functional)

2. System design

Determining all the system architecture in this phase and data storage

3. Object design

It is the classification of objects into different classes and attributes

4. Implementation

Summing up the prepared design into software

# **13.0 Comparative Analysis of the Output of the Adopted Methodologies**

* **Jacobson methodology** This traceability enables reuse of analysis and design work, possibly much bigger factors in the reduction development time then reuse of code unlike the OMT methodology.
* **Unlike OMT, Jacobson methodology** focuses on the concept of the use cases
* In OMT analysis, the results are shown as object and dynamic and functional model, while in Jacobson analysis uses cases is used.
* In Jacobson methodology use cases is used to describe one main flow of events.

The extends relationship is used when you have one use case that is like another use case but does a bit more. In essence extends the functionality of the original use case (like a subclass)

* Unlike **OMT,** Jacobson’s use cases are

1. for scenarios for understanding the system requirement
2. can be an interaction between user and the system to capture the goal of the user and the responsibility of the system.
3. Non formal text with no clear flow of event
4. Easy to read with clear flow of event
5. Formal style using pseudocode

* use case description in Jacobson methodology must contain

1. How and when the use case begins and ends.
2. The interaction between the use case and its actors including when the interaction occurs and what is exchanged.
3. Exception to the flow of events

**Advantages of using Jacobson methodology**

1. Enable reuse of analysis and design work
2. Faster and demand less time
3. More simple
4. Cover the entire life cycle
5. Stress on traceability

**Disadvantages of using Jacobson methodology**

1. it only depends on the use case design lacking some details.

**Advantage of OMT methodology**

1. Produce reusable code
2. Can be extended
3. Detailed design rather than Jacobson design
4. Class attributes, method, inheritance, and association also can be expressed easily

**Disadvantage of OMT methodology**

1. Demand relatively higher time than Jacobson methodology

2. More complex

# **14.0 Architectural Model**

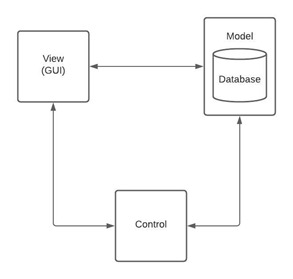
**MVC Architecture**: It is an application design model comprised of three interconnected parts. They include the model (data), the view (user interface/GUI), and the controller (processes that handle input). The MVC model or "pattern" is commonly used for developing modern user interfaces. It is an application design model comprised of three interconnected parts. They include the model (data), the view (user interface/GUI), and the controller (processes that handle input).

Figure 27: MVC Architecture

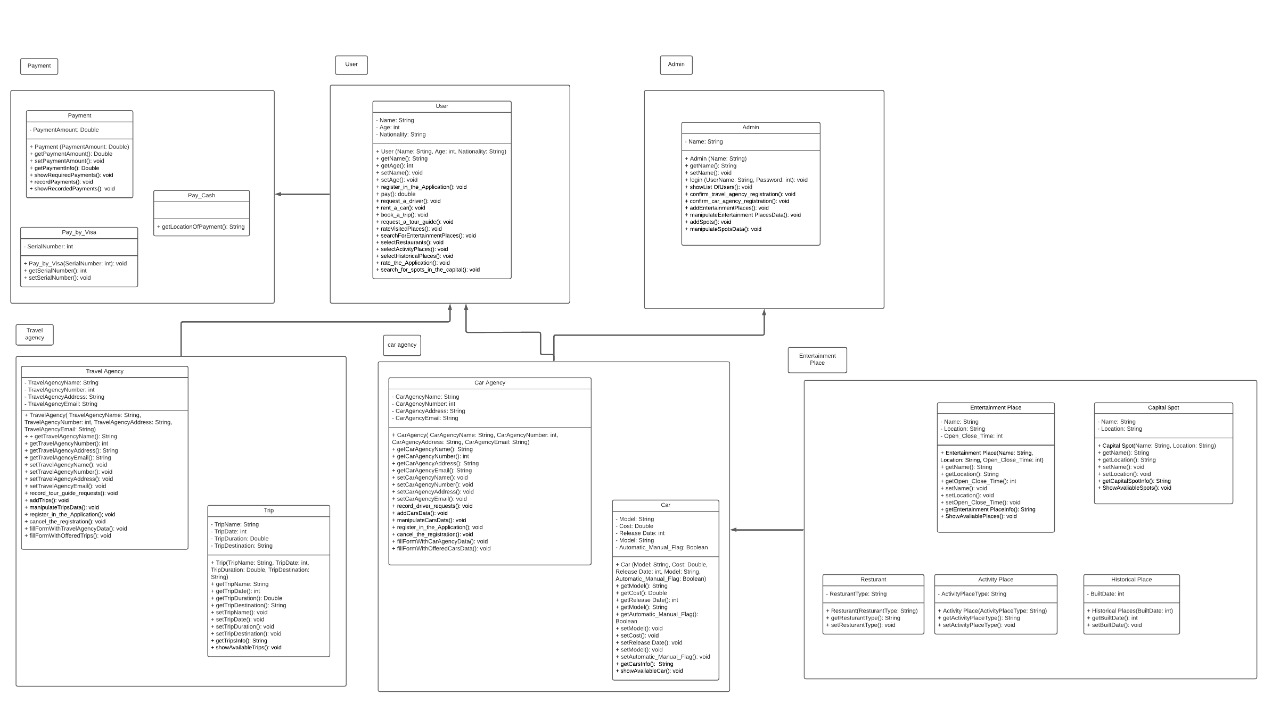


Figure 28: Class diagram Architecture

**Layered Architecture:**

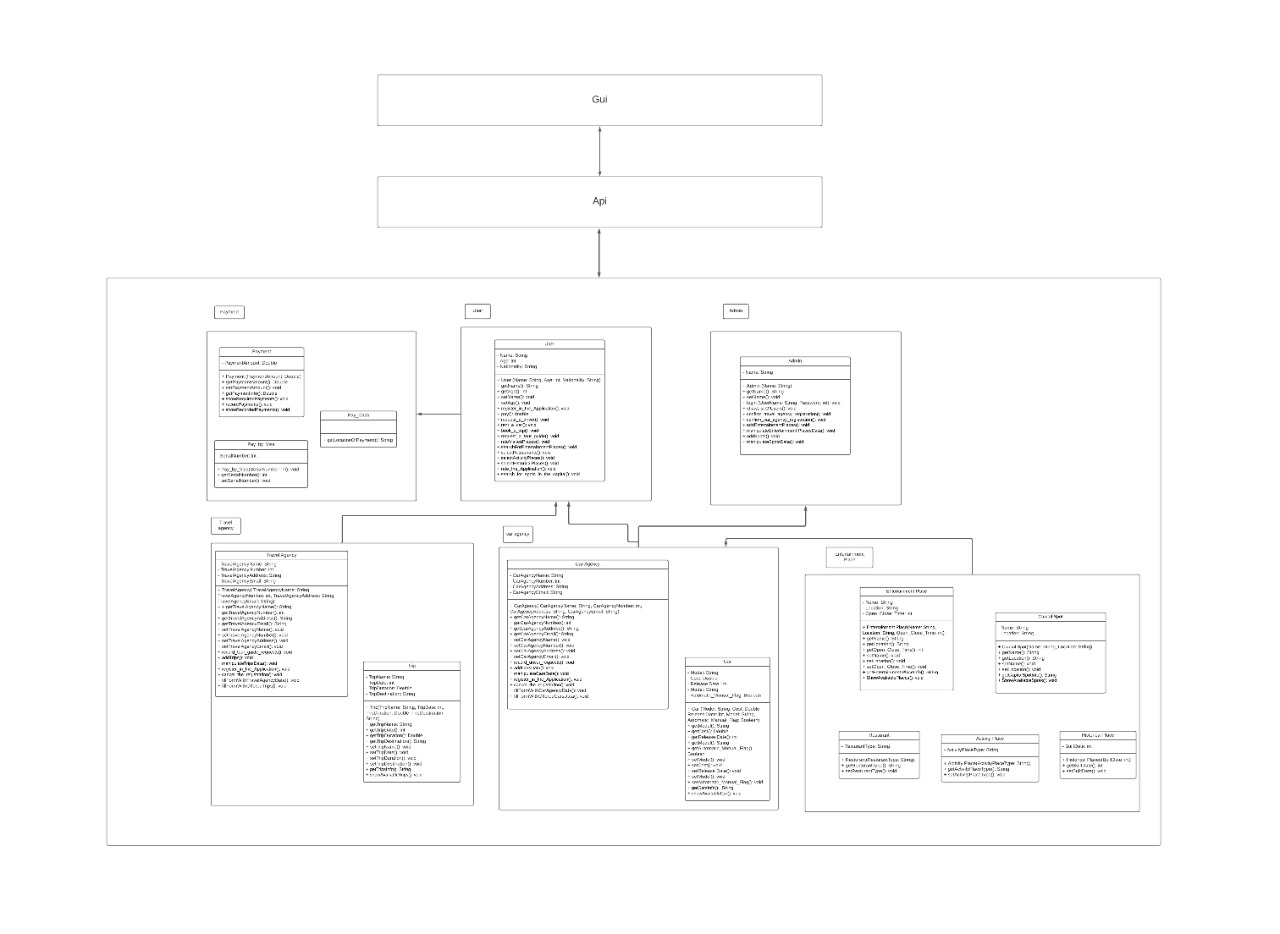
This type of architecture is done to handle different level of abstraction so it changes any component that form each layer above each other, all layers must run sequentially (Cannot be skipped).

Figure 29: Layered Architecture

**Data Centered Architecture:**

This type of architecture shows, different components that communicate through shared data repositories the components access a shared data structure and are relatively independent, in that, they interact only through the data store. This type is used to show the components that access the database.

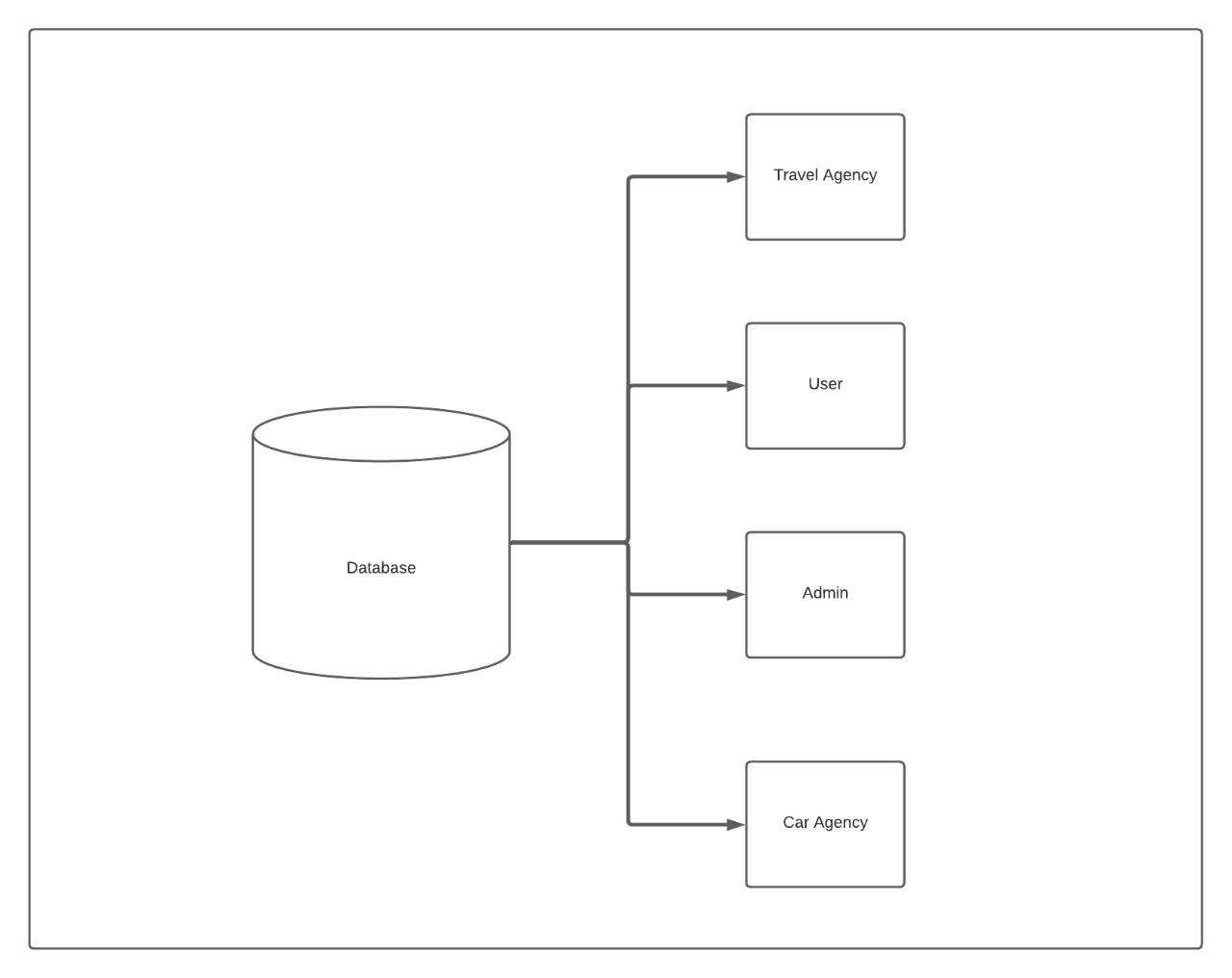


Figure 30: Data Base Architecture

**Merge Architecture:** Type of architecture where all mentioned previous types (MVC, layered and data centered) are combined and show full architecture style of the system

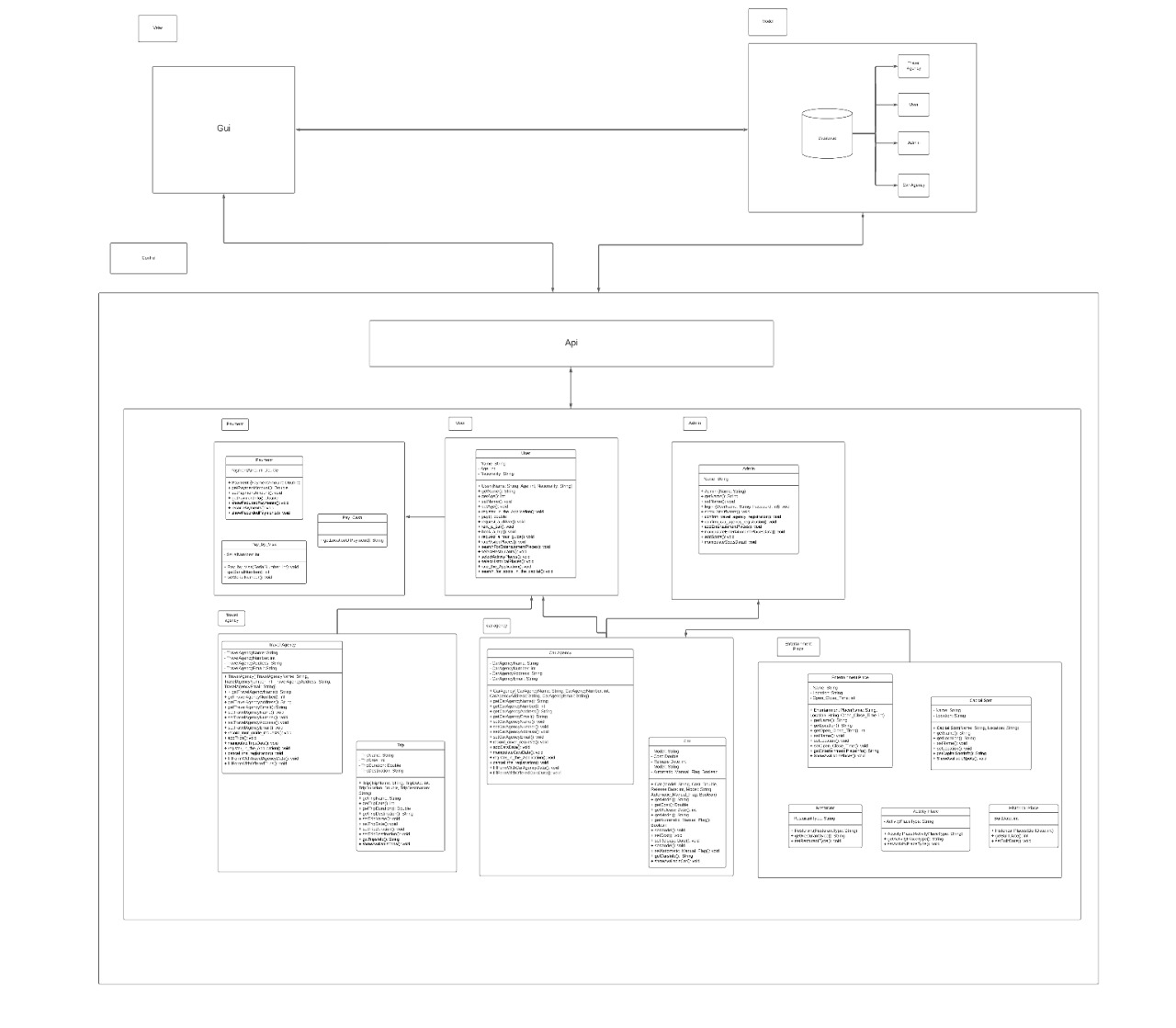


Figure 31: Merge Architecture

# **15.0 Component Diagram**

Each component is an independent, executable entity. It does not have to be compiled before it is used with other components. The services offered by a component are made available through an interface and all component interactions take place through that interface. This enhances the reusability of components. For example the UserManager component: It requires the following interfaces userinfo, user review, and reservation while providing the following interfaces reservedactivities, userreview, and user history this component can be generalized for reusability in other systems. Also the travelmanager and carmanager components, which require agencyinfo and edittrip/editcars provides newagencyrequest and availabletrips/offeredcars the adminmanager component requires newagencyrequest and provides nominateactivities interface. Finally the activitiesmanager component requires nominateactivities, userreview, reservedactivity, availabletrips and offered cars while providing activityhistory and displaysuggestedandavailbleactivities.

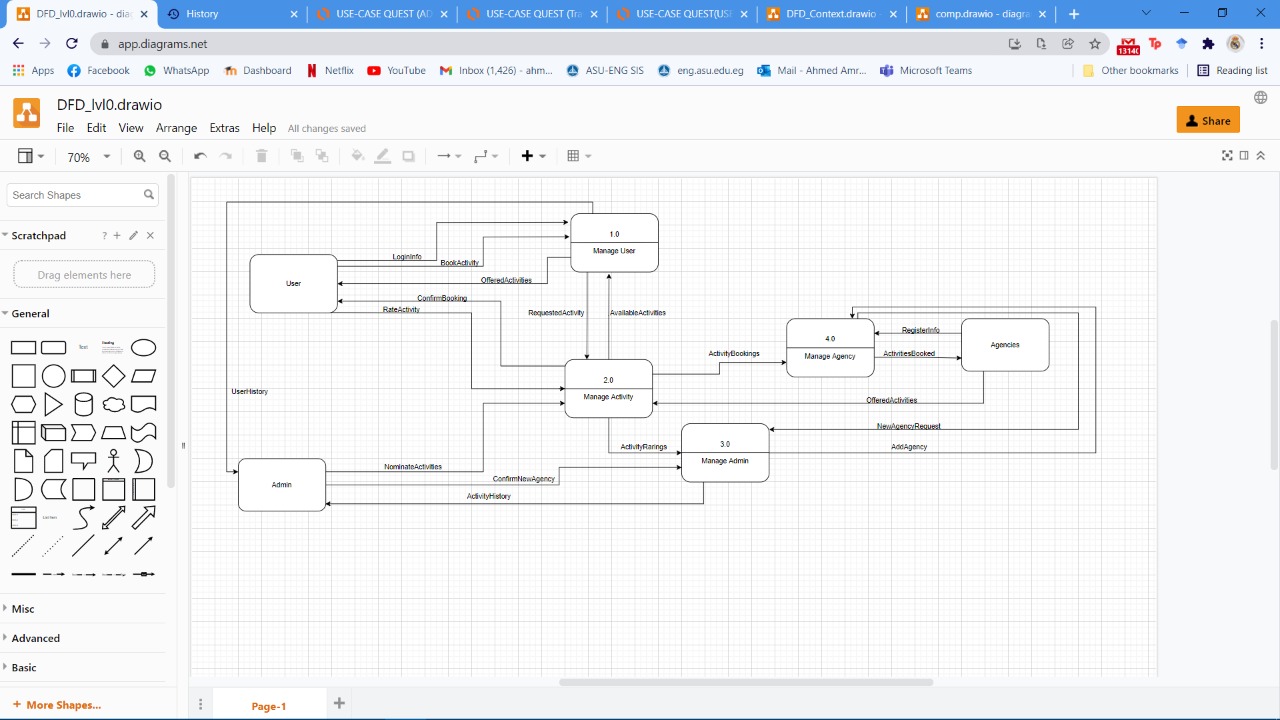


Figure 32: Component Diagram

# **16.0 Detailed Design**

You will find the detailed design files in the zip file of this project

## 16.1 Database Schema

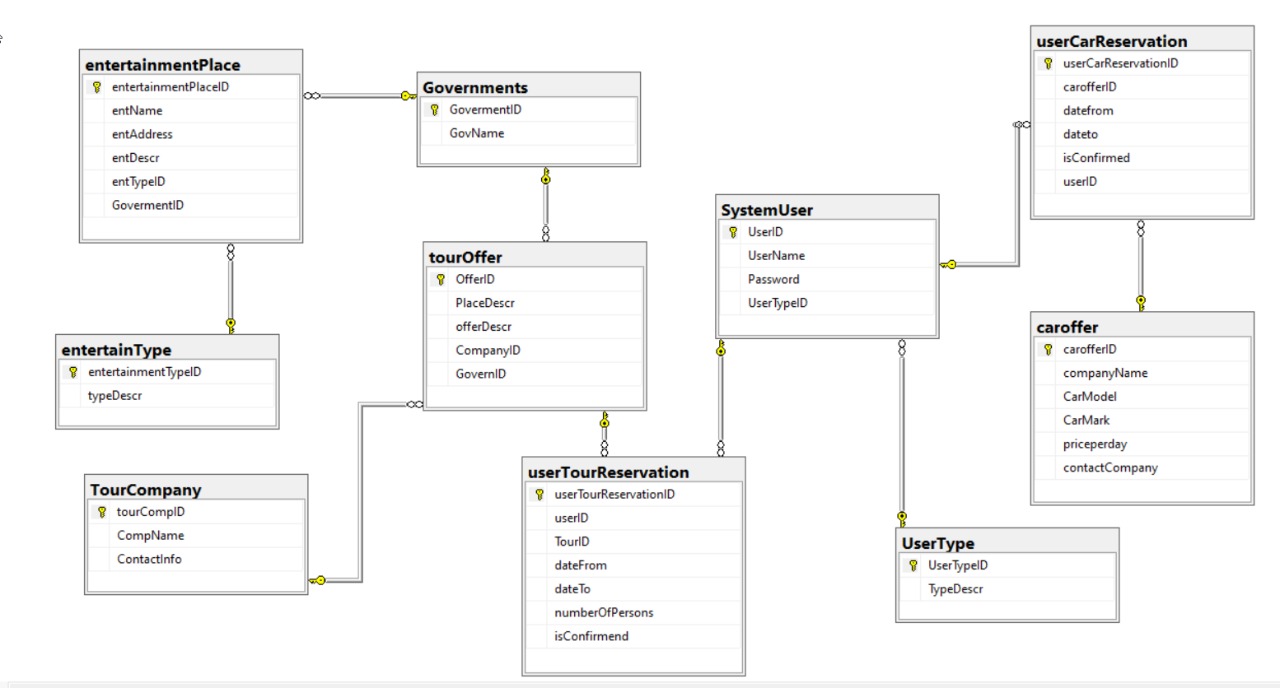


Figure 33: Data Base Schema

# **17.0 Testing**

**Software Testing:** is a method to check whether the actual software product matches expected requirements and to ensure that software product is Defect free.

The process of testing has several stages and many types; we will discuss the types of testing which we can apply in our project

* First, we will apply **class level testing:** We use this kind of testing to notice the behavior of the object by invoking its operations because may some restrictions are required in the order of invoking them mainly because some operations may not be invoked inside the object unless another operation is invoked first to set certain states (attributes) of the object.

We will apply class level testing for every class in our project: User, Admin, Payment (which is inherited by pay\_by\_visa and pay\_cash), travel agency, trip, car agency, environment places (which is inherited Restaurant, Activity Place, and Historical Place), car, and capital spot.

* **User class possible sequence:**

register\_in\_the\_Application,pay,[book\_a\_trip|rent\_a\_car|request\_a\_driver|requesta tourguide|rateVisitedPlaces|searchForEntertainmentPlaces|selectRestaurants|selectActivityPlaces|selectHistoricalPlaces|rate\_the\_Application|search\_for\_spots\_in \_the Capital

* **Admin class possible sequence:**

Login, [showListOfUsers|confirm\_travel\_agency\_regestiration|confirm\_car\_travel\_agency\_

regestiration|addEntertainmentPlaces|mainipulateEntertainmentPlacesData|addSpots|manipulateSpotsData

* **Payment possible sequence:**

[showRequiredPayments|recordPayments|showRecordedPayments

* **Pay\_by\_Visa**

[getSerialNumber|setSerialNumber

* **Pay\_Cash**

[getLocationOfPayment

* **Travel Agency**

register\_in\_the\_Application,[addTrips|record\_tour\_guide\_requests|manipulateTripsData| fillFormWithTravelAgencyData| fillFormWithOfferedTrips, cancel\_the\_registration

* **Car Agency**

register\_in\_the\_Application,[addCarsData|record\_driver\_requests|fillFormWithCarAgencyData|fillFormWithOfferedCarsData|manipulateCarsDatacancel\_the\_registration

* **Car**

[getCarsInfo|showAvailableCar

* **Trip**

[getTripsInfo|showAvailableTrips

* **Entertainment Place**

[getEntertainment PlaceInfo|ShowAvaliablePlaces

* **Resturant**

[getResturantType|setResturantType

* **Activity Place**

[getActivityPlaceType|setActivityPlaceType

* **Historical Place**

[getBuiltDate|setBuiltDate

* **Capital Spot**

[getCapitalSpotInfo|ShowAvaliableSpots

* Second our next step in testing is **unit testing**: in this type Individual components are tested independently in our project we have 5 components (User Manager, Admin Manager, Activity Manager, Travel Manager, Car Manager)
* Third step in the testing sequence is **integration testing**: in this type Involves building a system from its components and testing it for problems that arise from component interactions So in our project we will define the components which will deal with each other and test them:
* User Manager Component and Activity Manager
* User Manager Component and Travel Manager
* User Manager Component and Car Manager
* Admin Manager and Activity Manager
* Admin Manager Component and Travel Manager
* Admin Manager Component and Car Manager
* Fourth step in testing sequence is **system testing**: in this type we test all our project together
* Fifth step in testing is **User Acceptance testing** this stage of testing is divided into 2 types:
* **Alpha testing**: is a type of acceptance testing; performed to identify all possible issues/bugs before releasing the product to everyday users or the public. The focus of this testing is to simulate real users by using a black box and white box techniques. The aim is to carry out the tasks that a typical user might perform. Alpha testing is carried out in a lab environment and the testers are our team members who are working on this project.
* **Beta/acceptance testing**: Beta Testing of a product is performed by "real users" of the software application in a "real environment" and can be considered as a form of external User Acceptance Testing. Beta version of the software is released to a limited number of end-users of the product to obtain feedback on the product quality. Beta testing reduces product failure risks and provides increased quality of the product through customer validation. It is the final test before shipping a product to the customers. Direct feedback from customers is a major advantage of Beta Testing. This testing helps to test the product in customer's environment.
* There are some more types of testing which we have applied on our project:
* **Security testing:** verifies that protection mechanisms built into a system will, in fact, protect it from improper penetration.
* **Stress testing**: executes a system in a manner that demands resources in abnormal quantity, frequency, or volume.
* **Regression Testing**: is defined as a type of software testing to confirm that a recent program or code change has not adversely affected existing features. It is a full or partial selection of already executed test cases which are re-executed to ensure existing functionalities work fine.
* **Smoke testing**: also known as “Build Verification Testing”, is a type of software testing that comprises of a non-exhaustive set of tests that aim at ensuring that the most important functions work. The result of this testing is used to decide if a build is stable enough to proceed with further testing.
* **Sanity Testing**: is a kind of Software Testing performed after receiving a software build, with minor changes in code, or functionality, to ascertain that the bugs have been fixed and no further issues are introduced due to these changes. The goal is to determine that the proposed functionality works roughly as expected. If sanity test fails, the build is rejected to save the time and costs involved in a more rigorous testing.



* Now we have finished our testing for the project,

and this figure[34] illustrates all the main stages of

testing which we apply on our project.

Figure 34: Testing Graph

# **18.0 Estimated Project Cost**

Software is a dicey business because for it to be considered successful it needs to meet all the requirements, be delivered on time, and last but not least be within the budget. That is why early project cost estimation is required to determine whether the project will be within the client’s budget or not. The cost estimation is affected by several things including the efforts, tools, duration, and overheads. The cost estimation for the Quest project will be using the function points methodology.

## 18.1 Function Points

1 Function points table

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| User Input | User Output | Inquires | Files | Interface |
| 1. Enter User Info 2. Enter agency Info 3. Enter offered activities 4. Edit activity 5. Book activity | 1. Show offered activities 2. Booking confirmation | 1. Special request for activities 2. Payment method | 1. User Info 2. Admin Info 3. Agency info 4. Activities reviews | 1. Payment with visa/fawry |

2 Function point calculation

|  |  |  |  |
| --- | --- | --- | --- |
| Measurement Parameter | count | Weighing factor | Total |
| Number of user inputs | 5 | 4 | 20 |
| Number of user outputs | 2 | 4 | 8 |
| Number of inquires | 2 | 6 | 12 |
| Number of files | 4 | 10 | 40 |
| Number of interfaces | 1 | 5 | 5 |
|  |  |  | ∑ =85 |

3 Function point fourteen factors

|  |  |
| --- | --- |
| 1-Backup and recovery | 2 |
| 2-Data communication | 4 |
| 3-Distributed processing function | 3 |
| 4-Is performance critical? | 2 |
| 5-Existing operating environment | 3 |
| 6-Online data entry | 5 |
| 7-Input transaction built over multiple screens | 3 |
| 8-Master files updated online | 4 |
| 9-Complexity of inputs, outputs, files, and inquiries | 2 |
| 10-Complexity of processing | 2 |
| 11-Code design for re-use | 3 |
| 12-Are conversion/installation included in design | 1 |
| 13-Multiple installations | 0 |
| 14-Application designed to facilitate change by user | 1 |
|  | ∑ = 35 |



FP = 85.425

## 18.2 COCOMO II Model

startCOCOMO, 1

MonteCarlo, MonteCarlo\_Off

AutoCalculate, On

size\_type, Function Points

function\_points, 85.425

language, 3rd Generation Language

prec, Nominal

flex, High

rely, High

data, High

cplx, Nominal

ruse, High

docu, Very\_High

resl, Low

team, Very\_High

acap, Nominal

pcap, Nominal

pcon, Nominal

apex, Low

pexp, Low

ltex, High

pmat, Nominal

time, Nominal

stor, High

pvol, Low

tool, Very\_High

site, Nominal

sced, High

software\_maintenance, Off

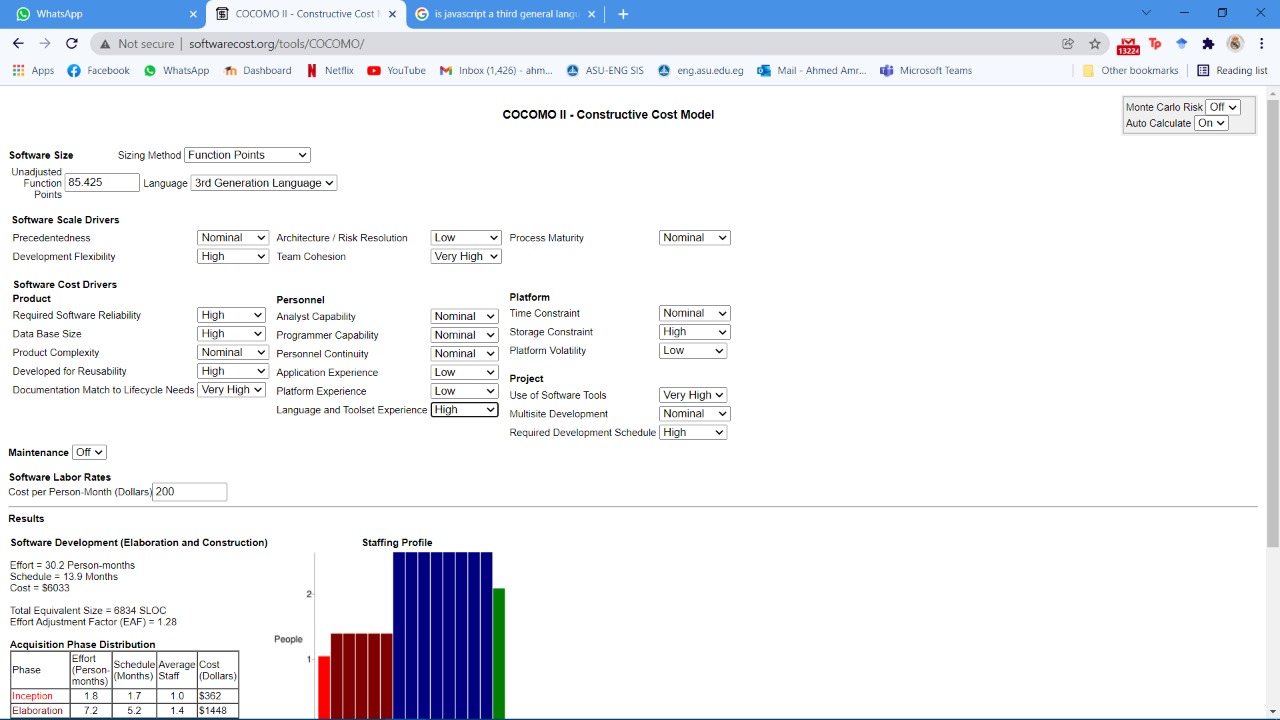
software\_labor\_cost\_per\_PM, 200

software\_EAF, 1.28

size\_exponent, 1.0818

schedule\_exponent, 0.314

software\_effort, 30.2

software\_schedule, 13.9

# 

Figure 35: COCOMO II graph and calculations

## 18.3 Expert Judgement

We asked an expert who had domain knowledge in web apps and after carefully viewing our factors and seeing that the programming capability, application experience and platform experience are low he estimated that the project should not cost over 5000 dollars

## 18.4 Comparative Analysis

**FPA analysis and COCOMO II:**

Function Point Analysis (FPA) is a method of projecting size, while the Constructive Cost Model (COCOMO) is a method of estimating resources and schedules. Both tools were developed primarily from experience with third generation language (3GL) development environments. Provisions for interchanging the tools to estimate fourth generation language (4GL) environments are not always straight-forward -- particularly when estimating a form-based environment. Both FPA and COCOMO models can be adapted to estimate the costs of developing projects using non-procedural fourth generation languages. These tools can be used together to estimate size and project costs, and to assist in assessing the sensitivity of various project costs to development parameters such as programmer skill level and the user's concern for reliability.

**Expert Judgement Estimation:**

One or more experts in both software development and the application domain use their experience to predict software costs. Process iterates until some consensus is reached

**Advantages:** Relatively cheap estimation method. Can be accurate if experts have direct experience of similar systems

**Disadvantages:** Very inaccurate if there are no experts!

# **19.0 User Guide**

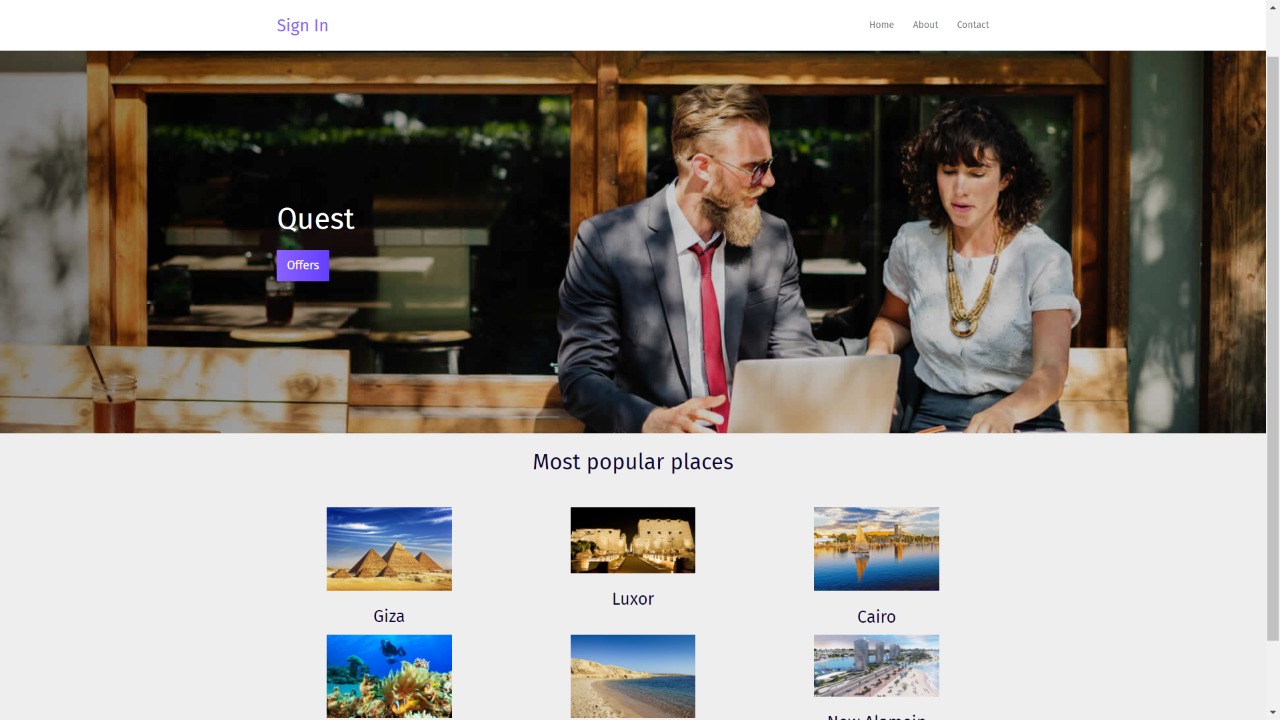


Figure 36: Home Page in the Website

Landing page when you first open the website

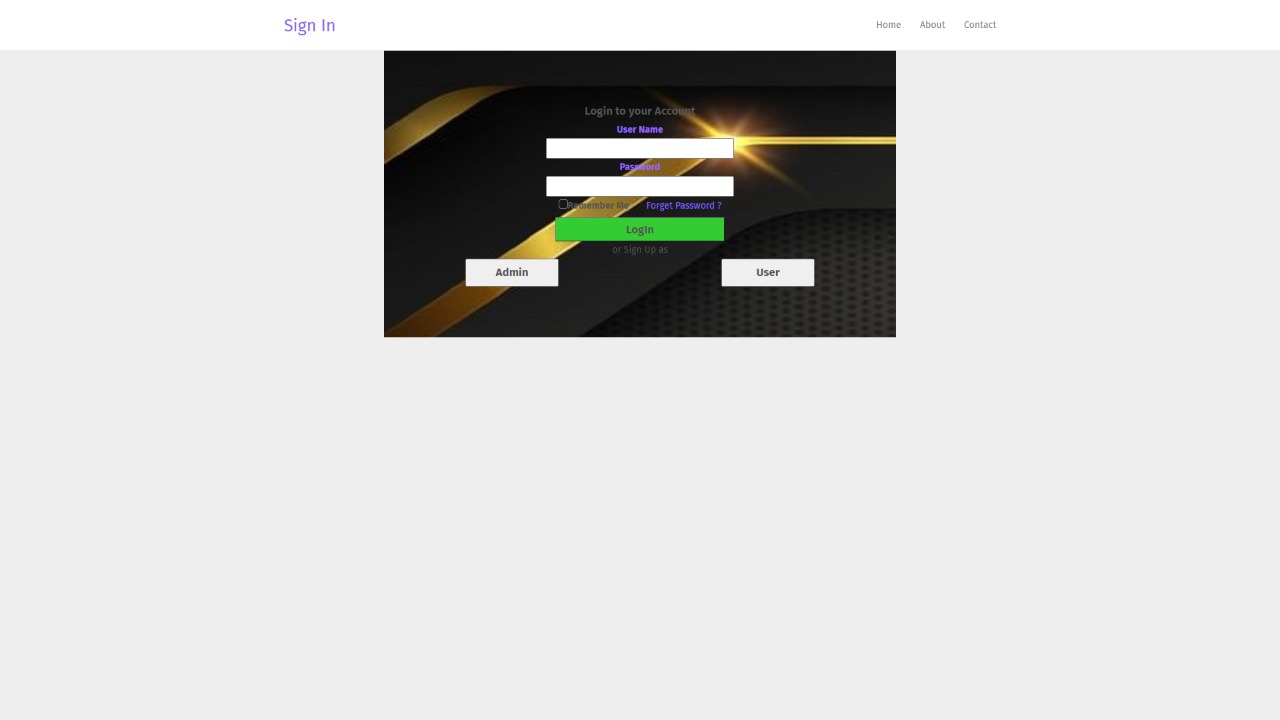
As a client, you either sign up if you are new or log in

Figure 37: Sign Up Page in the Website

Graphical user interface, website

Description automatically generated

Figure 38: Login Page in the Website

Graphical user interface, application, table, Excel

Description automatically generated

Figure 39: Entertainment Places Page in The Website

If you clicked on places, all entertainment places will be available for you by filtering the results according to your preferences

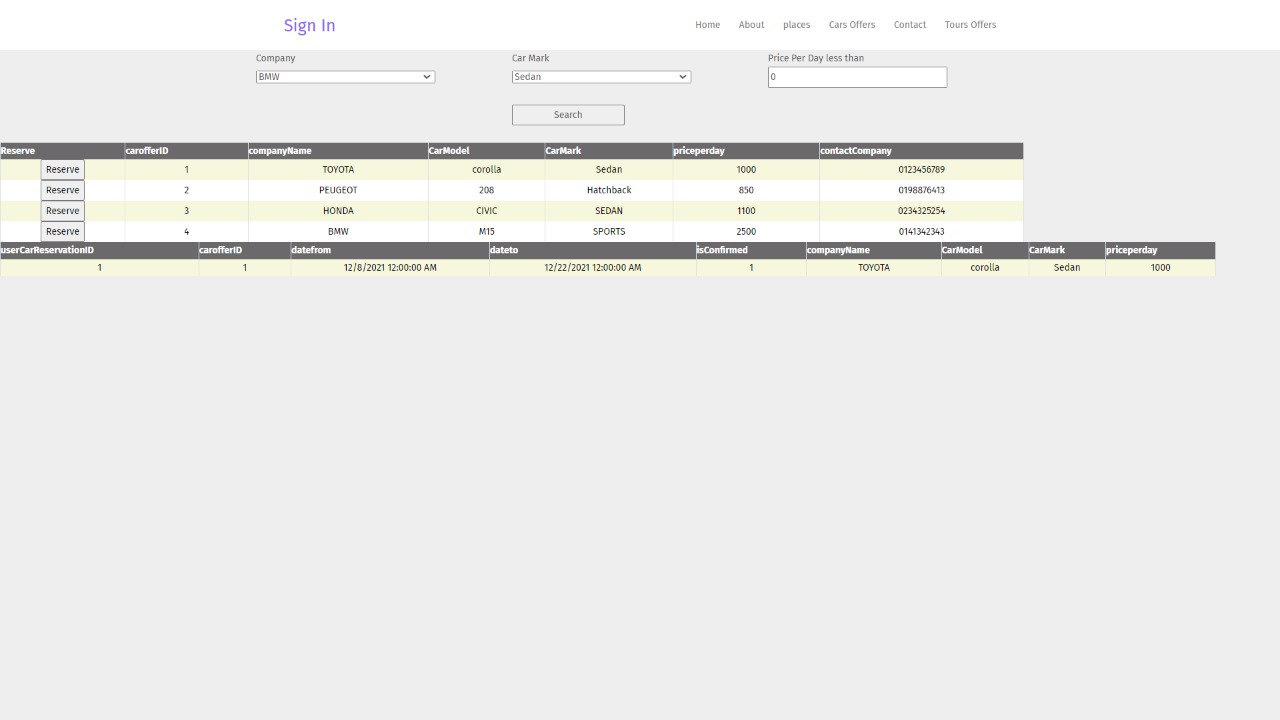


Figure 40: Rent Care Page in the Website

If you want to rent a car you click on cars offered and filter the results according to your preference and reserve the car you want by clicking on reserve button

Graphical user interface

Description automatically generated with medium confidence

Figure 41: Filtering while choosing your rented car

Filtering results

Graphical user interface

Description automatically generated

Figure 42: Filtering Results (Rent a Car)

Graphical user interface

Description automatically generatedif you want to view offered tours, click on tour offers, filter the results, and reserve the tour you like by clicking on reserve button

Figure 43: Reserve Trips Page in the Website

Graphical user interface, application, Word

Description automatically generated

Figure 44: Pay Page in the Website

To pay for your services

Admin side view:

If the user signed in as an admin

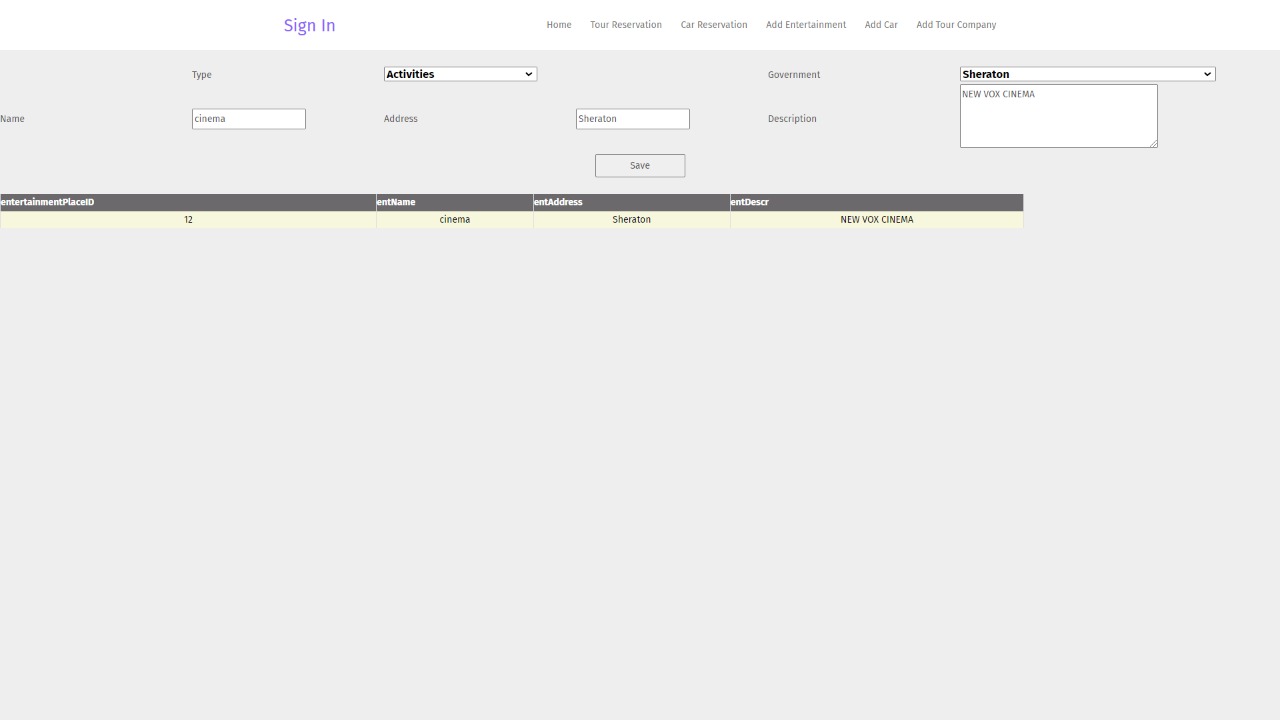


Figure 45: Admin Side View in the website

If the admin clicked on add entertainment, he will be able to add/modify entertainment places on the website

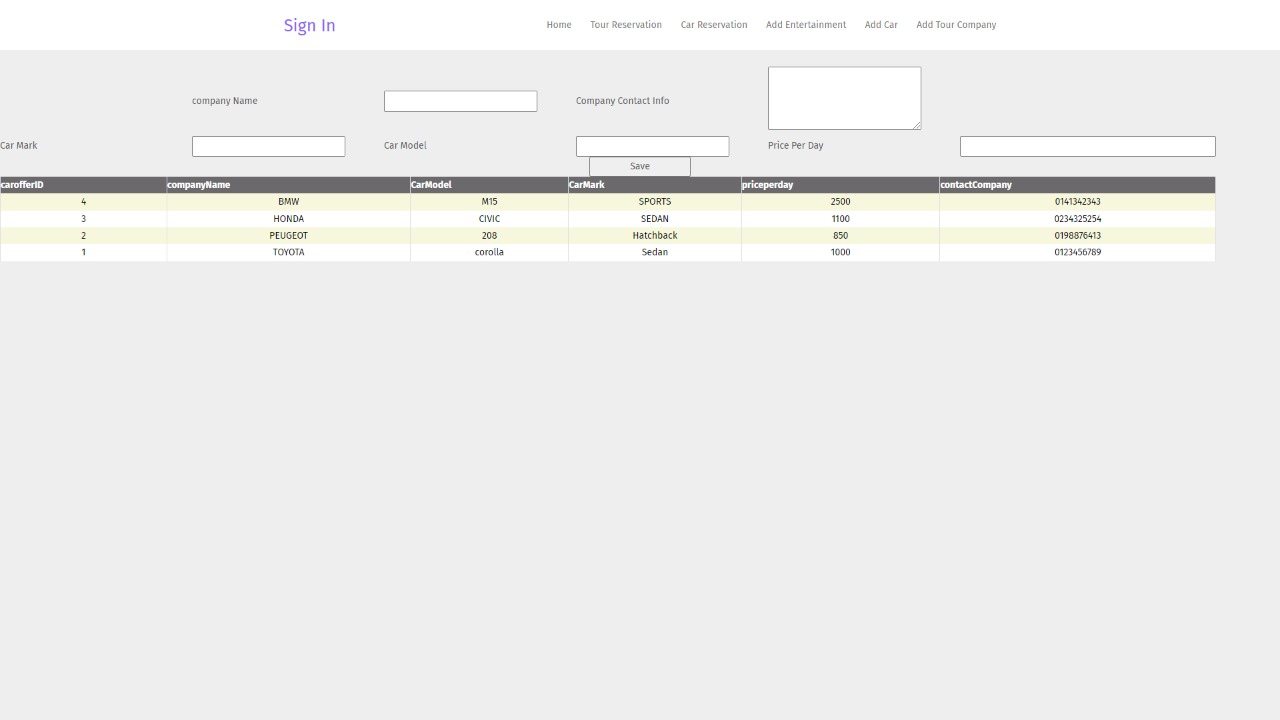


Figure 46: Admin add a car to the system in the website

If the admin clicked on add car, he will be able to add/modify offered cars on the website



Figure 47: Admin add a tour company to the system in the website

If the admin clicked on add tour company, he will be able to add/modify offered tours on the website

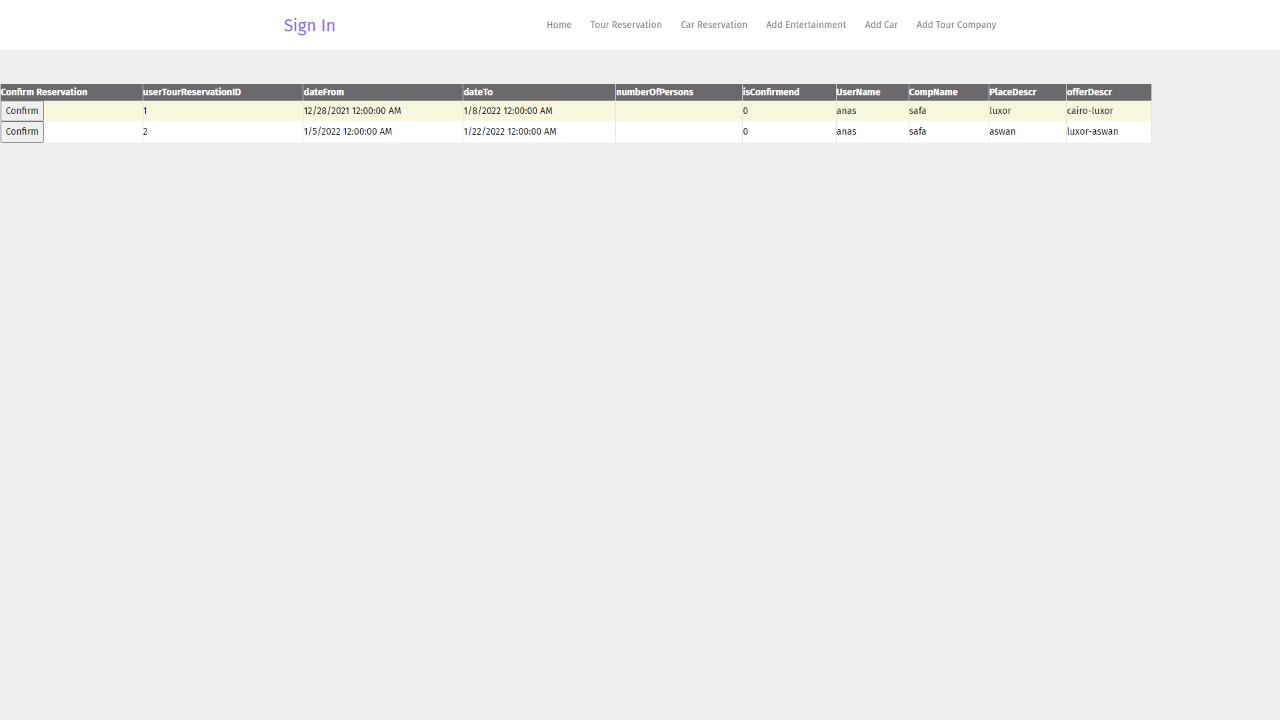


Figure 48: Admin confirms tour requests

Admin can confirm tour requests

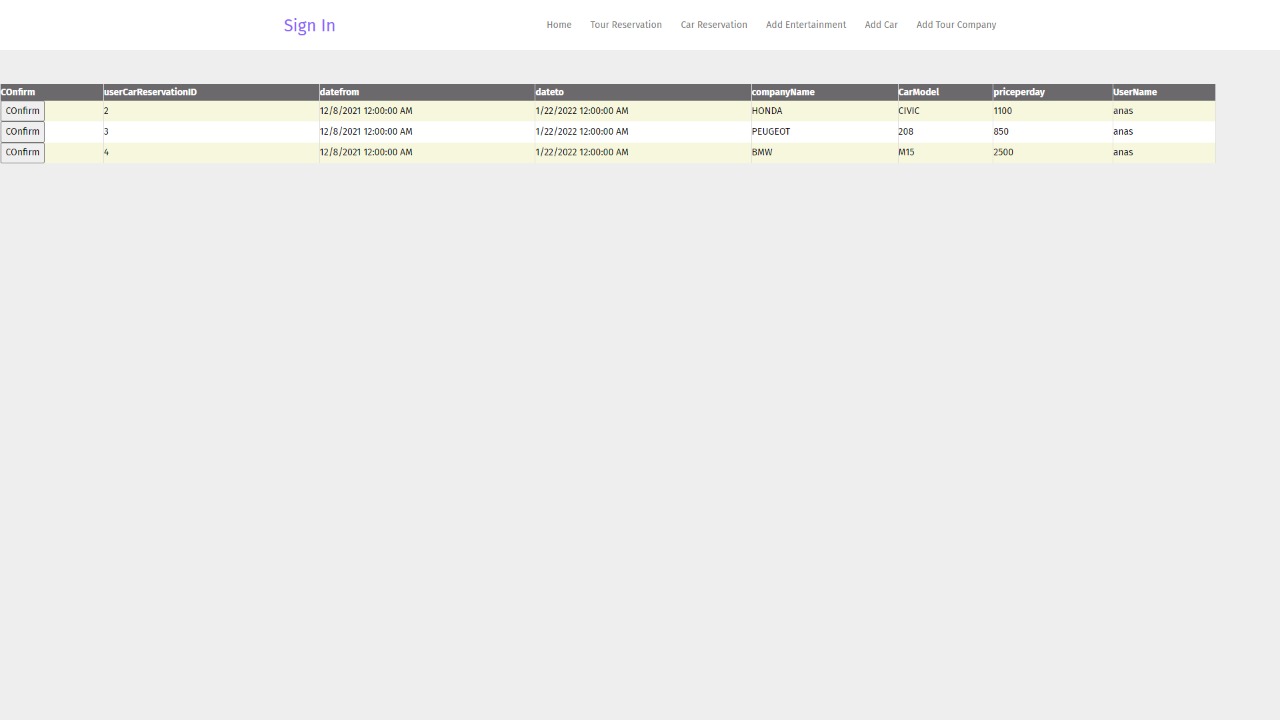


Figure 49: Admin confirms car requests

Admin can confirm car requests

# Graphical user interface, application, table, Excel Description automatically generated**20.0 Time Plan**

Figure 50: Time Plan



Figure 51: Detailed Time Plan

Time Plan is in the zip file

# **21.0 Appendices**