Industry Visit Tour System

**1.0 Preface:**

This is first Version of "The Industry Visit Tour System".

**Writenne By:**

* Ahmed Ashraf
* Ahmed Rafaa
* Amaal Mohamed
* Marwan Elsayed
* Mohamed Nasr
* Mohamed Anter
* Omnia Mohmed

**Date**: 19/4/2021

**Version:** 1.0.0

# Introduction:

* This system help people or college to visit Industry Company by booking industrial tour packages online.
* The System consist of two type of users : Users and Admins.
* User need to register onto the system then login to access the system.
* User can view packages and choose the best of them by modern filter tools, also he/she can specify the number of visitors for the tour that he/she want, based on the number of visitors the system calculate the cost of tour and pay online by credit card like Visa card or Master Card.
* After booking is successful, user receives an email notifying the booking details.
* The system is managed by admin that add subjects or places or packages details and specify the details for each person for industry visit also can view all added package details , and users details.
* The system has integration of the most popular payment gateways, as well as can process credit card payments.
* Many selection and filter tools allows customer to comparison shop.
* It is easy to use, robust, secured Website.

# Requirements and Features:

# User requirements include two type of services :

# User services : that help people or college to use system services [1.3:1.5] .

# Admin services : that help admins to manage the systems [1.6].

# [4.1]User services:

[4.1.1] **Registration and Login:**

New user needs to register first to access the system by filling up the basic registration details then he can login.

[4.1.2] **View places :**

User can see the list of added places with its details. User can search industry according to subjects or places.

[4.1.3] **view** **package**:

Can view all the packages, which are added by the admin with their details.

[4.1.4] **select package**:

the user selects desired packages as per entered destination location.

[4.1.5] **payment online**:

Once the user selects the package, he/she can proceed with payment.

[4.1.6] **Email notification**:

Once the payment is successful, user will receive a notification via email.

**[4.2]Admin services:**

[4.2.1]**Login**:

Admin need to login first in order to manage the system.

[4.2.2]**Add Subject**:

Can add subjects such as IT, Botany, Art, Hotel Management, etc., which is educational, related.

[4.2.3]**Add Place**:

Admin can add all the industrial related places such as Infosys, L&T, Wipro, etc.

[4.2.4]**Add / View Package Details**:

Admin defines a package detail for each person for industry visit. Also can view all the added package details.

[4.2.5]**View Booking Details:**

System allows admin to view booking details with the name of the bookie and its details.

[4.2.6]**View User Details:**

All the registered users will be displayed here with their details.

# System Specification:

**Functional requirements:**

**[5.1] User Registration:**

To use system, user shall register to the system then he can login and use system and this done by some ways it is:

1. user enter to registration page
2. then enter some information to system
3. then click register button.
4. System record this information to database and create a account to the user
5. Finally, he can login into system

**[5.2] User login:**

login into system this function is more important and it done by :

1.user enter to login page if he registered in the system.

2.enter his/her username and password

3. then user clicks in login button .

4.then the system check if user is recorded in the system or not :

* if user is recorded , system move user to next page.
* If user is not recorded system print "Login failed!!"

**[5.3] view place:**

System shall to show the places that are recorded in the system that admin added it done by :

1.user click in Places button.

2.then system show places on the screen.

**[5.4] view package:**

System shall to show the packages that are recorded in the system that admin added,

it done by :

1.user click in packages button.

2.then system show packages on the screen.

**[5.5] Select package:**

After System show packages on screen ,user can select a packages that he wants and reserve it ,this function done by :

1.System show packages on screen.

2.User select the package that he prefers by click on it.

3.System show the reservation form to the user.

4.User fill the form and click on next button to move to payment page to confirm the reservation.

**[5.6] Payment online:**

After finshing of selecting package user should payment online and confirm reservation this done by :

1. System shows payment Page.

2.User fill information about his/her card.

3.Then System revise this information:

1. If Info is right then system confirm reservation and record it and print on screen reservation details.
2. Else print on screen "wrong in info please fill again" and user shall fill Information again.

**[5.7]Email Notification:**

This feature is for user that make a reservation, the system send a notification before 24 hours to the user to remind him about the package.

**[5.8]Admin Login:**

1. System show option that will be "Login As Admin", this option is for admins to access the System and manage it this done by :
2. System shows in User Login page "Login As Admin" .
3. Admin Click it .
4. Then system move to Admin Login Page.
5. Admin enter the username and password and click on Login button.
6. System Check if username and password are recorded in the database.
7. If they are recorded in database, Admin access to the system,
8. Else print on screen "Not found!! Login Again".

[5.9] **Add places or Subjects:**

After successful Login for admin he can do some operation like add places or subjects and this is done by:

1. After login, system shows show some features for Admin like add places or add subject.
2. Admin select what he wants ,and system move him to the next page to do this operation.
3. Admin fill fields about this operation then submit it,
4. Then system check if all of these fields are filled or not:
5. If All are filled system record this information about this operation.
6. Else system print on screen "Fill all Fields" .

**[5.10] Add or delete or change Package:**

Anthor feature for Admin, it is add/delete/change/view package details, it done by:

1.After successful login admin can click in packages buttons to manipulate it.

2.then System show for him all packages and some option Like and add, delete and change package.

3.Admin do what he wants, and system confirm and record it on database.

**[5.11] view reservations:**

Anthor feature for Admin it is show all details about reservation for Admin :

1. Admin Click on Reservation button.
2. Then system show all Reservation and their details.

**[5.12] View User Details:**

This is feature for Admin, it is view all registered users with their details :

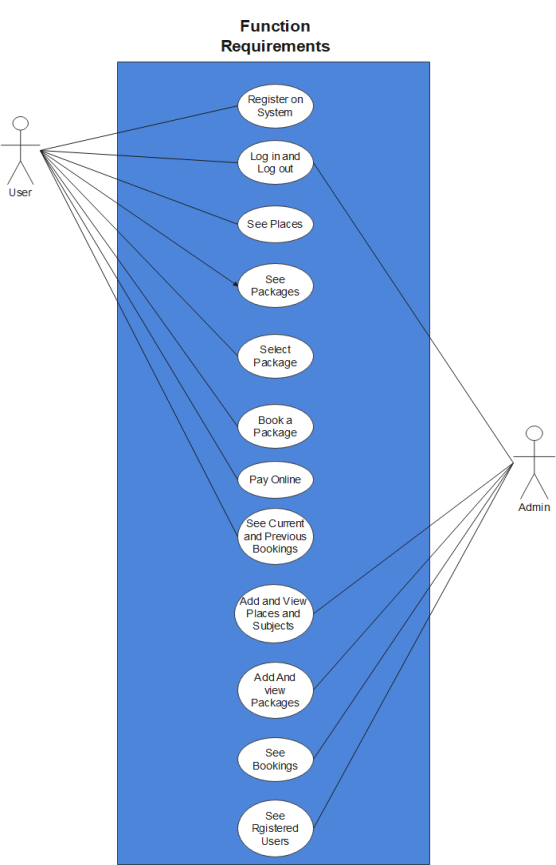
1.Admin click on Users button,

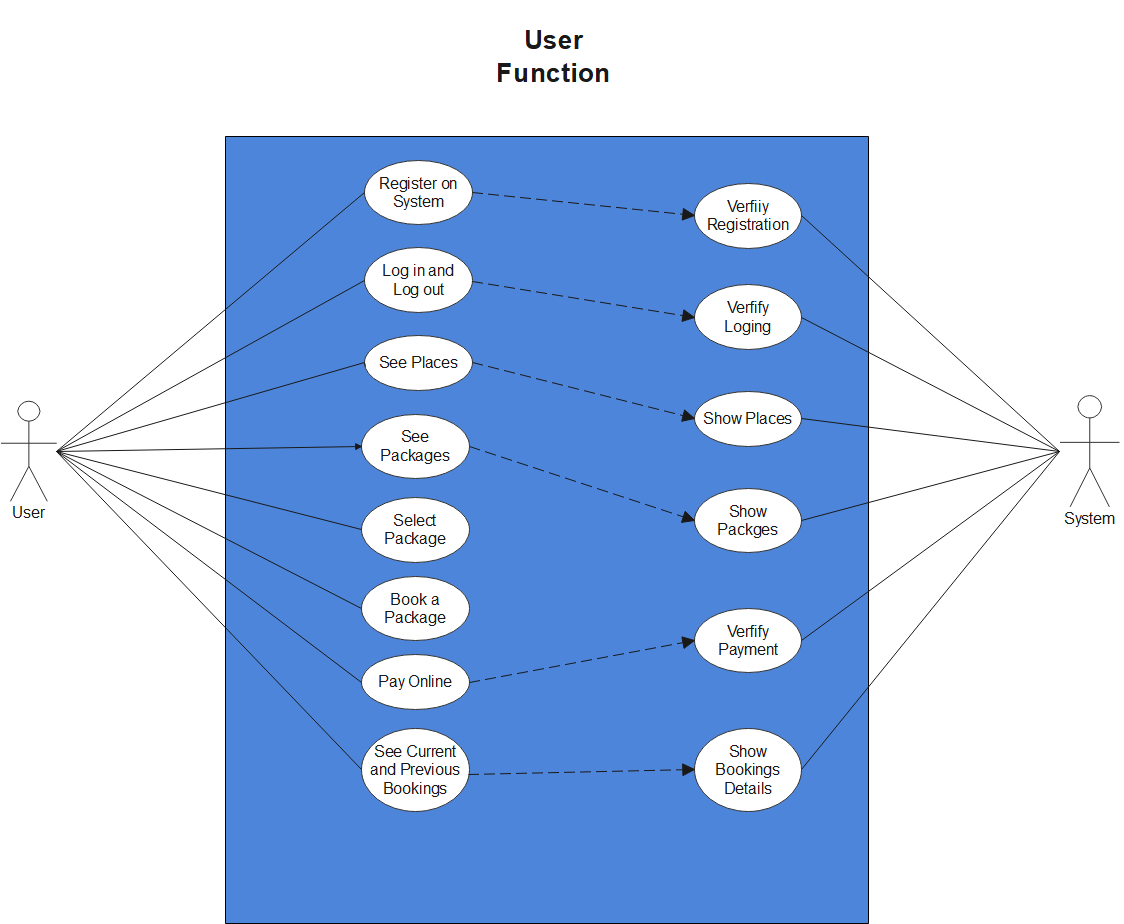
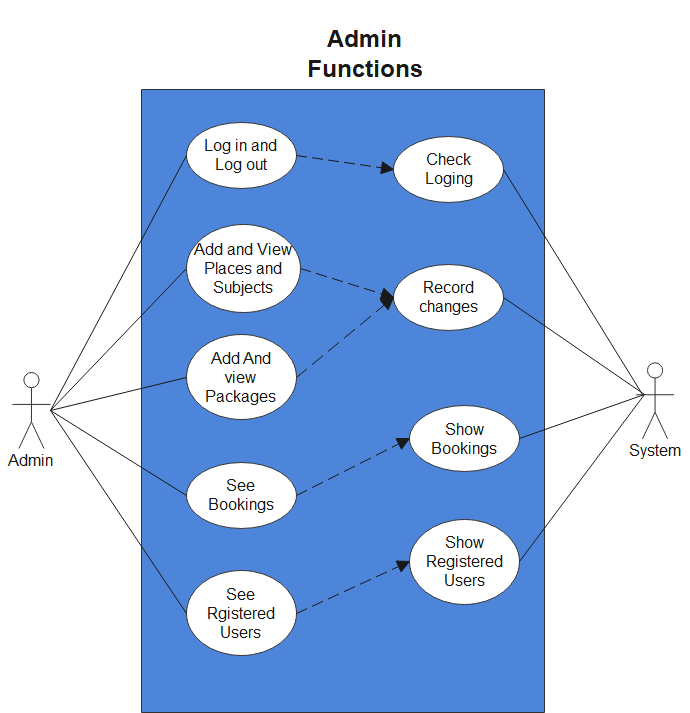
2. System Show all users with their details.

**[5.13] Filter Tools:**

It is for user to search easier about a good price for package

Appendices:

**8.1 Use-cases diagrams:**



**The USE-Case Description of User Functions:**

1-Use-case Registration:

|  |  |
| --- | --- |
| **1.Name and identifier:** | User Registration |
| **2.Initiator:** | system and user |
| **3.Goal:** | the user can be enter to pages through the registration |
| **4.Pre-condition:** | The user should be filling his/her the information |
| **5.Post-condition:** | The user can enter to system and finish his services. |
| **6. The Assumption:** | user enter to the system to use the software even can achieve his registering the information |
| **7-The Main success scenarios:**   1. user enter to registration page 2. then enter some information to system 3. Then click register button. 4. System record this information to database and create a account to the user 5. Finally, he can login into system | |

**2- Use-case login:**

|  |  |
| --- | --- |
| **1.Name and identifier:** | User login. |
| **2.Initiator:** | System and user. |
| **3.Goal:** | The user login into system. |
| **4.Pre-condition:** | The user should be register in the system  Even can login. |
| **5.Post-condition:** | The user can finishing his services in the system. |
| **6. The Assumption:** | The user should be enter his/her information even can login. |
| **7-The Main success scenarios:** | 1. User enter to login page if he registered in the system.  2.enter his/her username and password  3. Then user clicks in login button.  4.then the system check if user is recorded in the system or not :  i. if user is recorded , system move user to next page.  ii. If user is not recorded system print "Login failed!!" |

**3- Use-case view places:**

|  |  |
| --- | --- |
| **1.Name and identifier:** | Use-case view place |
| **2.Initiator:**  **3.Goal:** | User and system.  The user is see the places in the system. |
| **4.Pre-condition:** | * The user must be registering in system. * The system will show the places. |
| **5.Post-condition:** | The user can be see the places and choose his/her place. |
| **6. The Assumption:** | The user should be after login in the system click the button to show the places. |
| **7-The Main success scenarios:** | System shall to show the places that are recorded in the system that admin added it done by :  1. User click in Places button.  2. Then system show places on the screen. |

**4- use-case view package:**

|  |  |
| --- | --- |
| **1.Name and identifier:** | Use-case view package. |
| **2.Initiator:** | User and system. |
| **3.Goal:** | The user is see the package in the system. |
| **4.Pre-condition:** | * The user must be registering in system. * The system will show the packages. |
| **5.Post-condition:** | The user can be see the packages and choose his/her packages. |
| **6. The Assumption:** | The user should be after login in the system click the button to show the packages. |
| **7-The Main success scenarios:** | System shall to show the packages that are recorded in the system that admin added, it done by :  1. User click in packages button.  2. Then system show packages on the screen. |

**5- Use-case Select package:**

|  |  |
| --- | --- |
| **1.Name and identifier:** | Use-case Select package: |
| **2.Initiator:** | User and system. |
| **3.Goal:** | The user will select the package when the packages shows. |
| **4.Pre-condition:** | * The user must be registering in system. * The packages must be show. |
| **5.Post-condition:** | The user will be select the package he\she needed. |
| **6. The Assumption:** | The user must be select package. |
| **7-The Main success scenarios:** | After System show packages on screen ,user can select a packages that he wants and reserve it ,this function done by :  1. System show packages on screen.  2. User select the package that he prefers by click on it.  3. System show the reservation form to the user.  4. User fill the form and click on next button to move to payment page to confirm the reservation. |

**6- Use-Case Payment online:**

|  |  |
| --- | --- |
| **1.Name and identifier:** | Use-Case Payment online |
| **2.Initiator:** | User and system. |
| **3.Goal:** | The user payment the packages after it  Shows. |
| **4.Pre-condition:** | * The user must be registering in system. * The packages must be show. * The user must be select package. |
| **5.Post-condition:** | The user will be pay for package by his/her card. |
| **6. The Assumption:** | The user should be pay package online. |
| **7-The Main success scenarios:** | After finishing of selecting package user should payment  online and confirm reservation this done by :  1. System shows payment Page.  2. User fill information about his/her card.  3.Then System revise this information:  If Info is right then system confirm reservation and record it and print on screen reservation details.  Else print on screen "wrong in info please fill again" and user shall fill Information again. |

**7- Use-case Email Notification:**

|  |  |
| --- | --- |
| **1.Name and identifier:** | Use-case Email Notification |
| **2.Initiator:** | User and system. |
| **3.Goal:** | The user receive notifications from Email. |
| **4.Pre-condition:** | * The user must be registering in system. * The packages must be show. * The user must be select package. |
| **5.Post-condition:** | The user will get the notification from the system. |
| **6. The Assumption:** | The user must be receive message from the system. |
| **7-The Main success scenarios:** | This feature is for user that make a reservation, the system send a notification before 24 hours to the user to remind him about the package. |

**8-Admin Login:**

|  |  |
| --- | --- |
| **1-Name and identifier** | Admin Login |
| **2-inititor** | Admin or the system |
| **3-Goal** | Admin log in the system. |
| **4-Pre-condation** | The user s name and password are recorded in the database. |
| **5-Post-condation** | Admin will register and access to the system. |
| **6-The Assumption** | The person who will try to log in will be already an admin. |
| **7-The Main success scenarios:** | 1-the person try to enter the username and password and he enter them incorrectly so log in rejected  2-the person try to enter the username and password and he enter them incorrectly so log in accepted |

**9-Add a places or subject:**

|  |  |
| --- | --- |
| **1-Name and identifier** | Add a place or subject |
| **2-inititor** | Admin |
| **3-Goal** | Admin wants to do some operation like add place or subject. |
| **4-pre-Condation** | 1-Admin must do log in the system  2-All operation which Admin want to do is failed. |
| **5-post-Condation** | Operation is done. |
| **6-Assumpation** | Admin log in correctly and admin know how the operation is being done. |
| **7-Scenireos** | 1-After login Admin done operation which he wants to do then system check if it failed so it is done.  2- After login Admin done operation which he wants to do then system check if it failed so it is rejected. |

**10-Add or Delete or Change Package:**

|  |  |
| --- | --- |
| **1-Name and identifier** | Add or Delete or Change Package |
| **2-Inititor** | Admin |
| **3-Goal** | Is give the capability the Admin to add or delete or change (manipulation) view the package details. |
| **4-Pre-condation** | 1-the system show to the user all operation  2-Admin do operations which he wants to do. |
| **5-post-condaiton** | The system record and confirm the operation in the database. |
| **6-Assumption** | Admin log in correctly and know how the operation is done. |
| **7-sceneros** | The system select the h who want to do it and he did it and the system  Confirm and record the operation. |

**11-View Reservation.**

|  |  |
| --- | --- |
| **1-Name and identifier** | View Reservation |
| **2-inititor** | The system and the Admin |
| **3-Goal** | Admin git the capability to see all details about reservation. |
| **4-Pre-condation** | The Admin click on the reservation button. |
| **5-Post-condation** | The system shows all reservation and details. |
| **6-Assumption** | The Admin login successfully and know how to do this operation. |
| **7-sceneros** | the Admin click in the button of reservation and the system show him all view and details. |

**12-View User Details:**

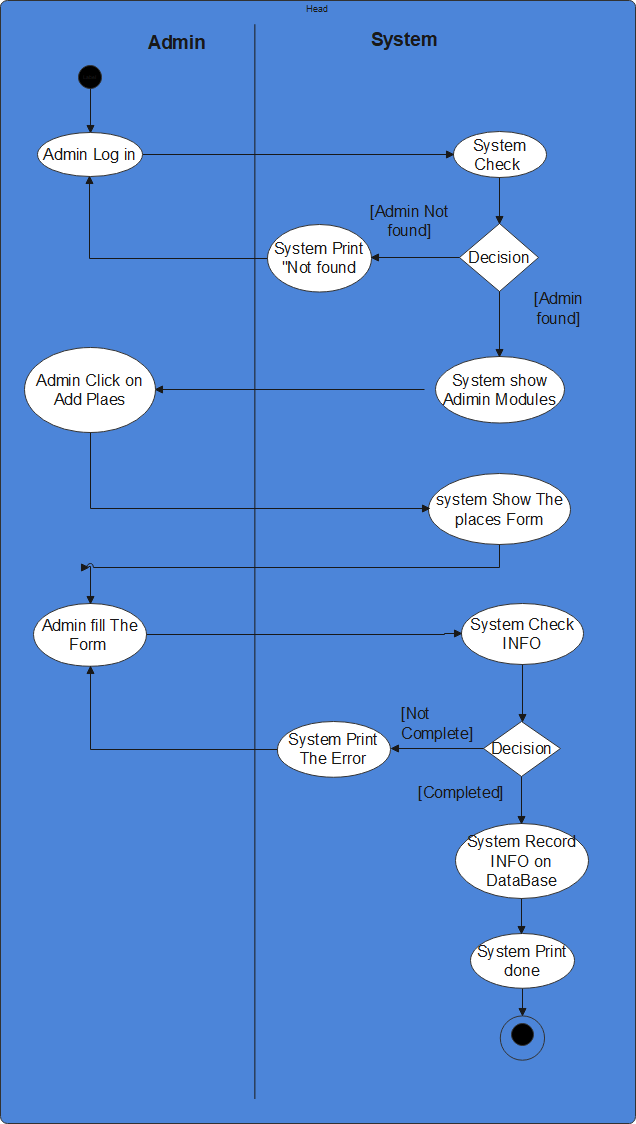
|  |  |
| --- | --- |
| **1-Name and Identifier** | View User details |
| **2-Inititor** | Admin |
| **3-Goal** | Admin see all register users and their details. |
| **4-Pre-condationus** | The Admin click in button which enable him to see all details about the user. |
| **5-Post-Condation** | System all users with their details. |
| **6-Assumpation** | The Admin login successfully and know how to do this operation. |
| **7-Scenreo** | After the login the user click on the button to see all users and their details and the system show them to him |

**13-Filter Tools**

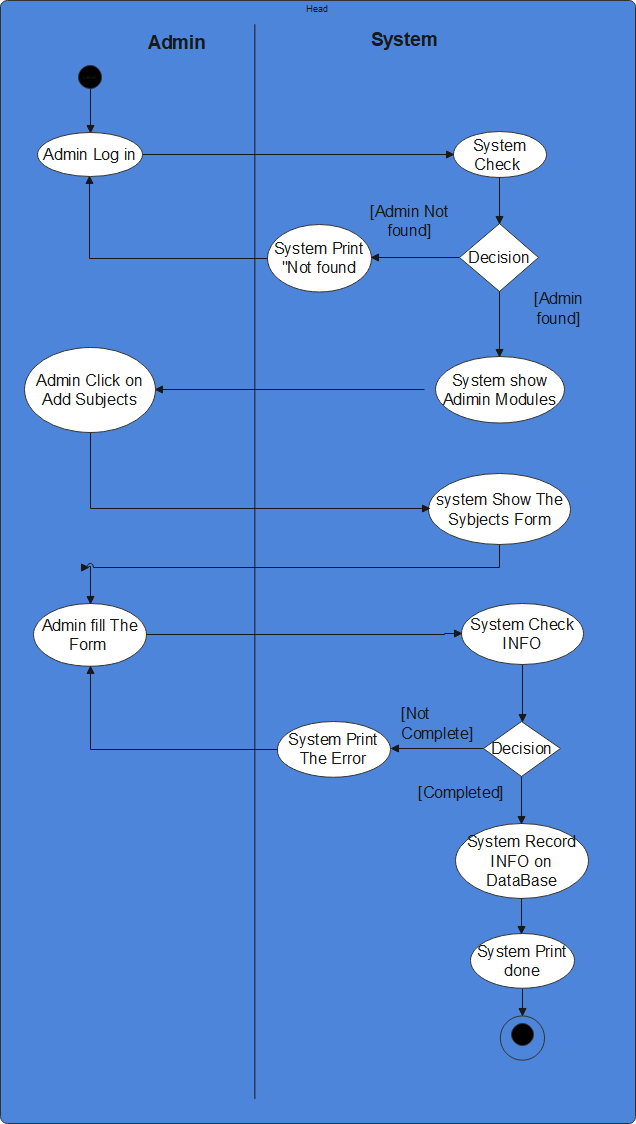
|  |  |
| --- | --- |
| 1-Name and Identifier | Filter Tools |
| 2-Inititor | The User |
| 3-Goal | The Admin login successfully and know how to do this operation. |
| 4-Pre-condation | 1-the user determine the element who want to determine  2-the user click on the buttons and  Change it to achieve her filtration. |
| 5-Post-Condation | The system will do filtration and show the user which he wants to filter after filtration. |
| 6-Assumpation | The system provides buttons to do the filtration and the user know how to do it. |
| 7-Scenreo | The user clicks on the button of the filtration and change the value of  Price to the range of prices which he want then he click on the button filter it. |

**8.2 Activity Diagrams:**

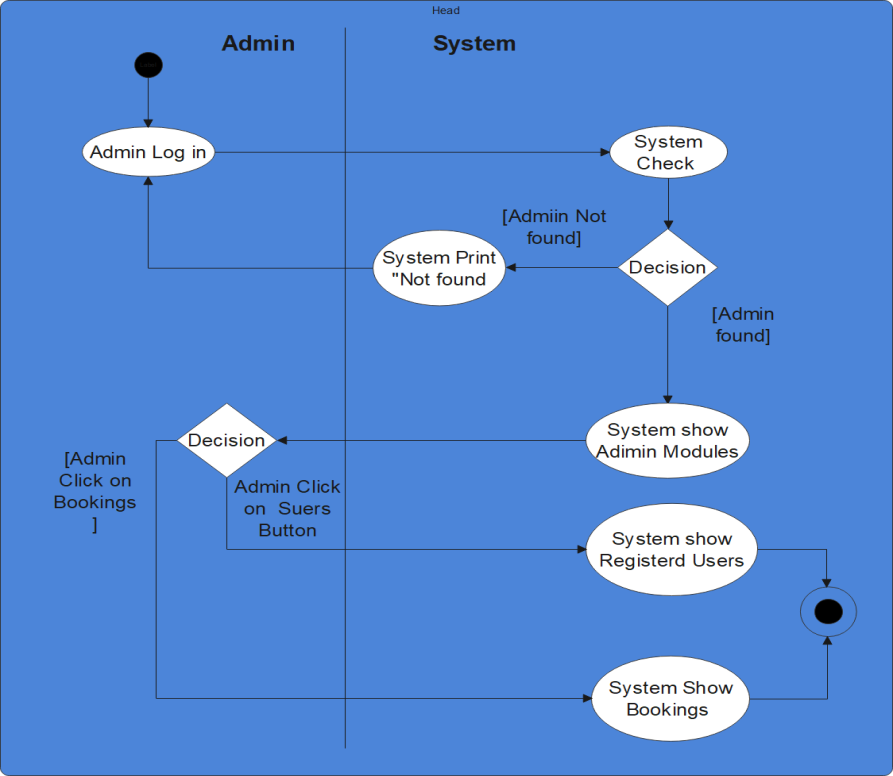
**[8.2.1]Admin Add places:**



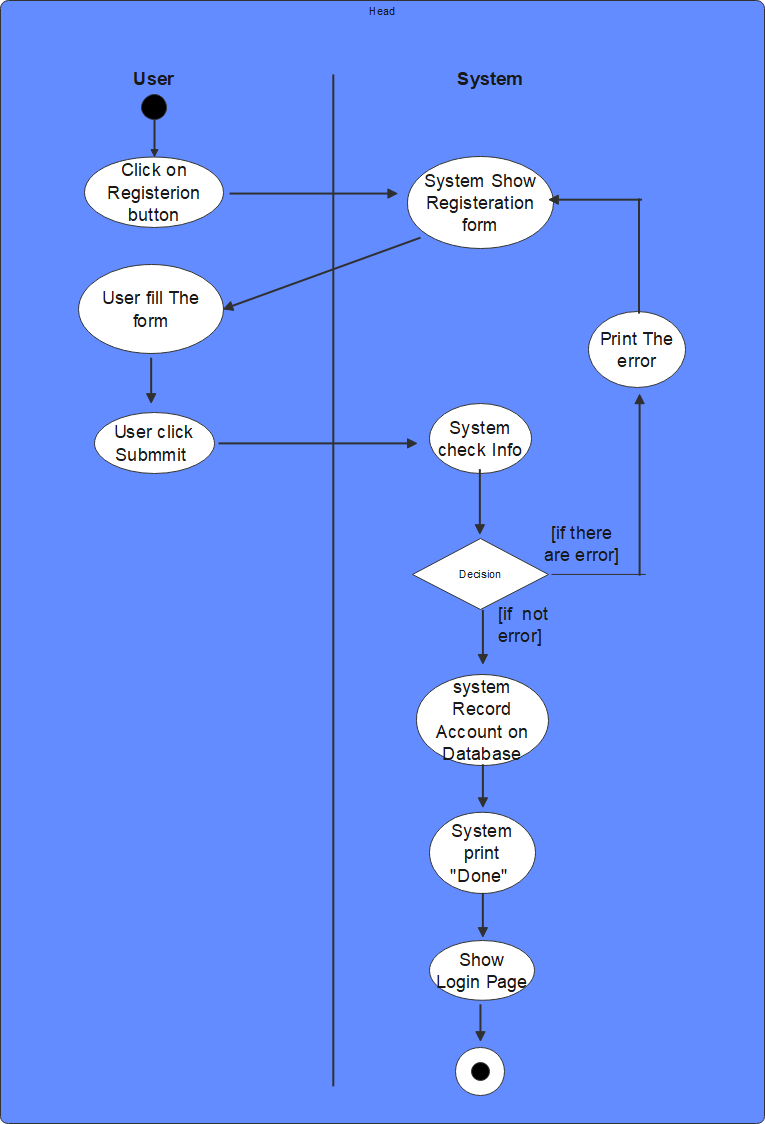
**[8.2.2]Add Subject:**



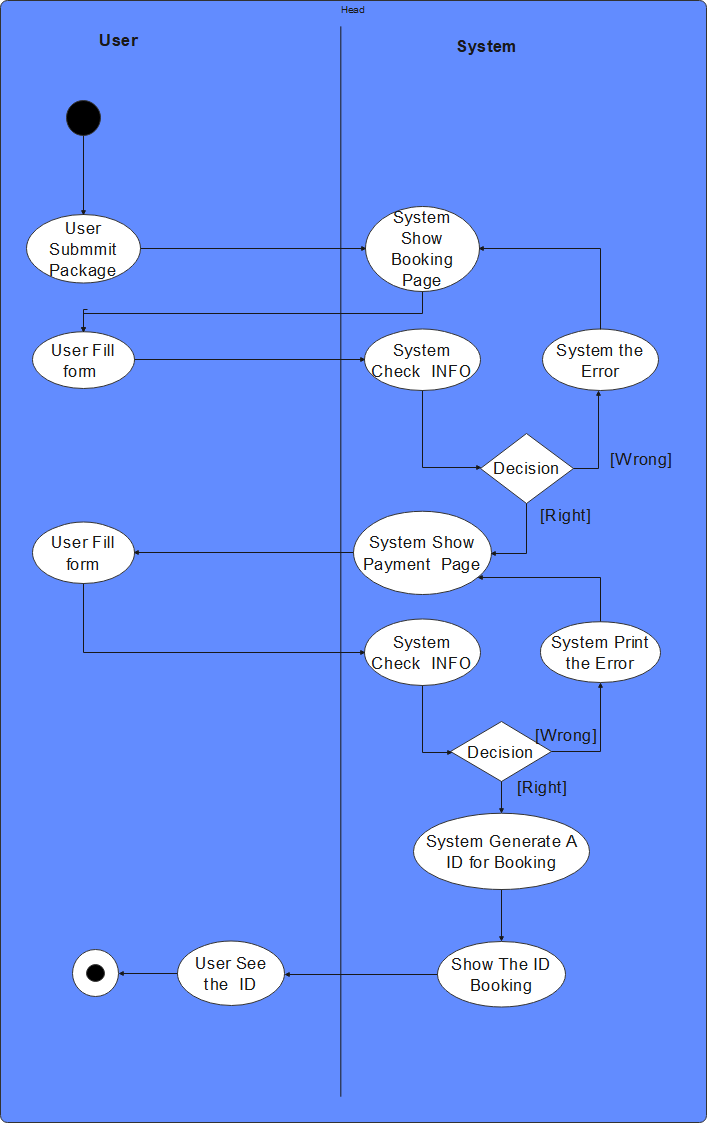
**[8.2.3]Admin See Bookings ,Users:**



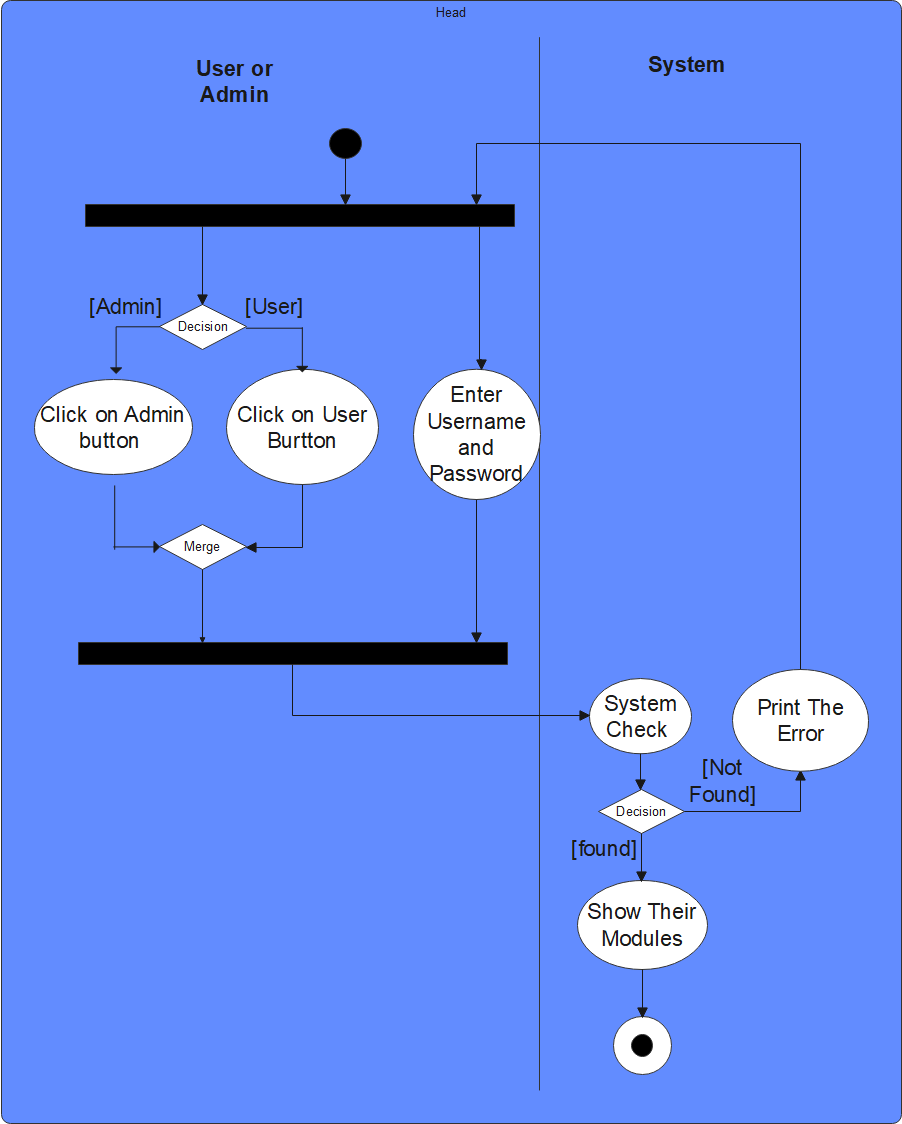
**[8.2.4]Registration:**



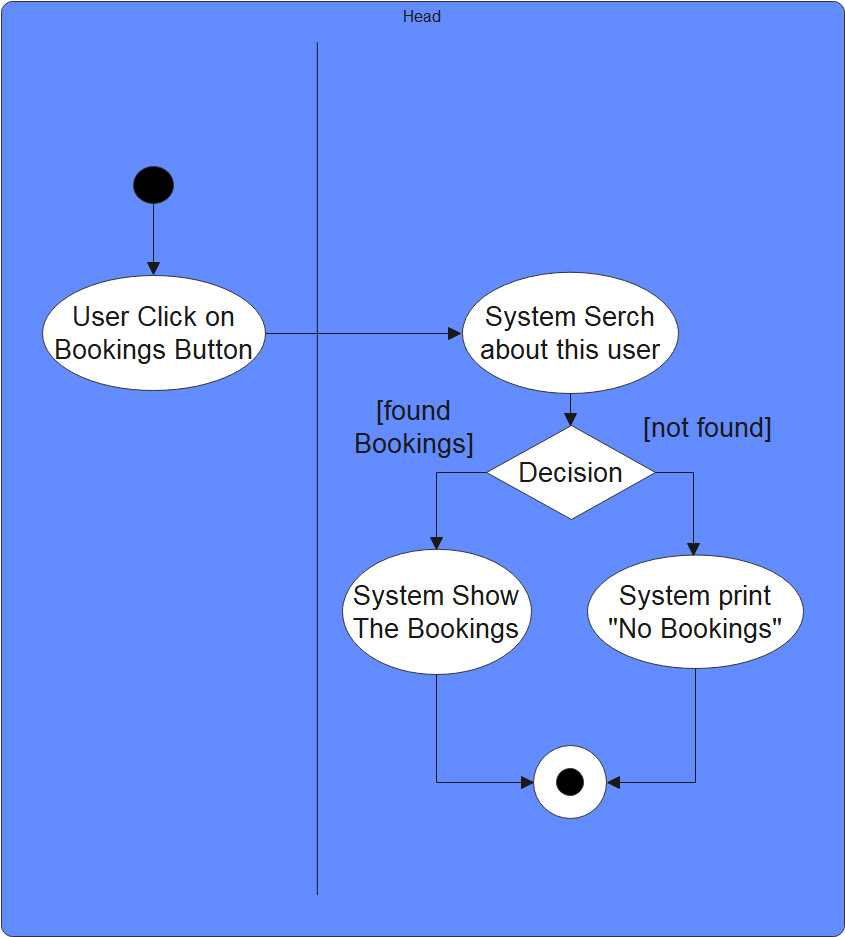
**[8.2.5]User book and pay online:**



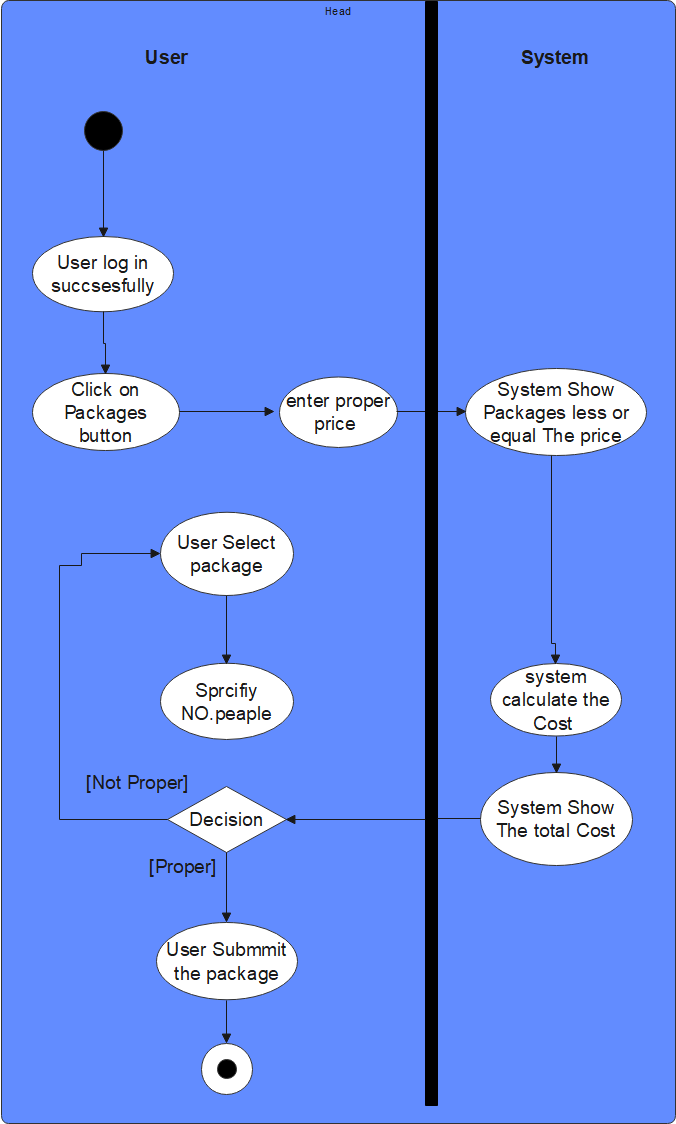
**[8.2.6]User and admin Login:**



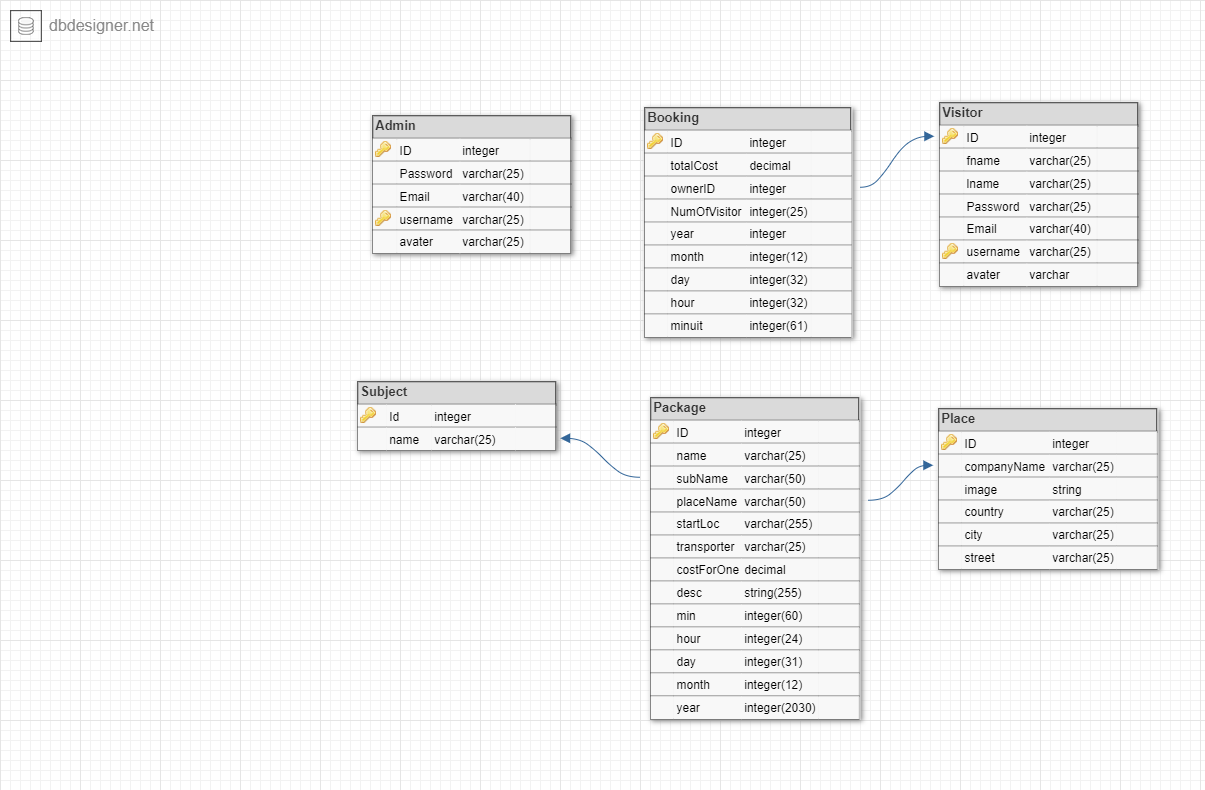
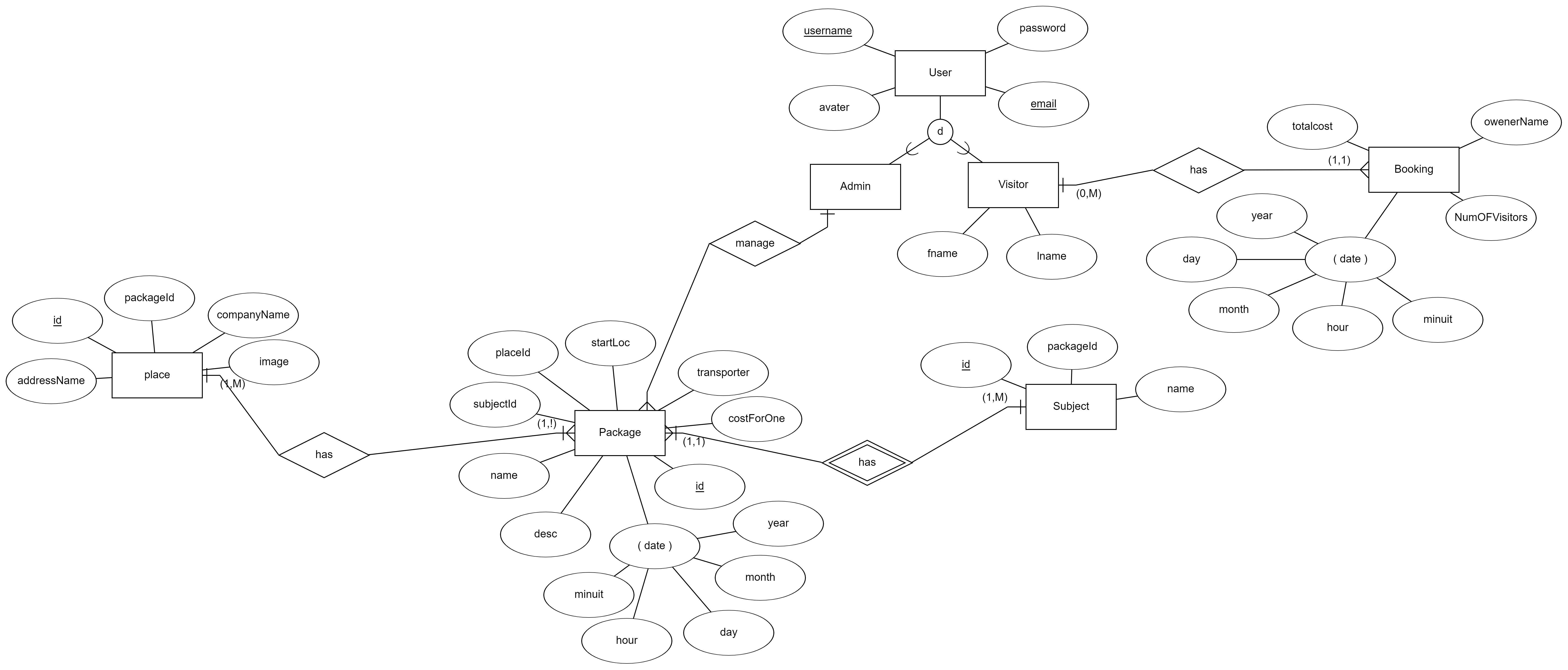
**[8.2.7]view Bookings for user:**



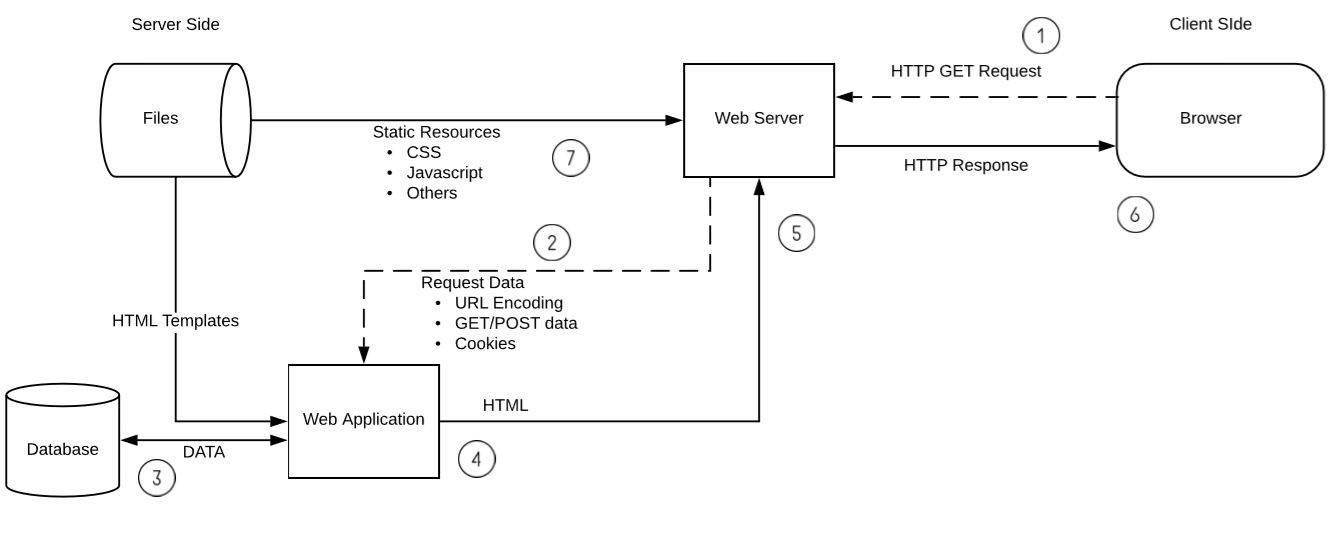
**[8.2.8]User select and view package:**



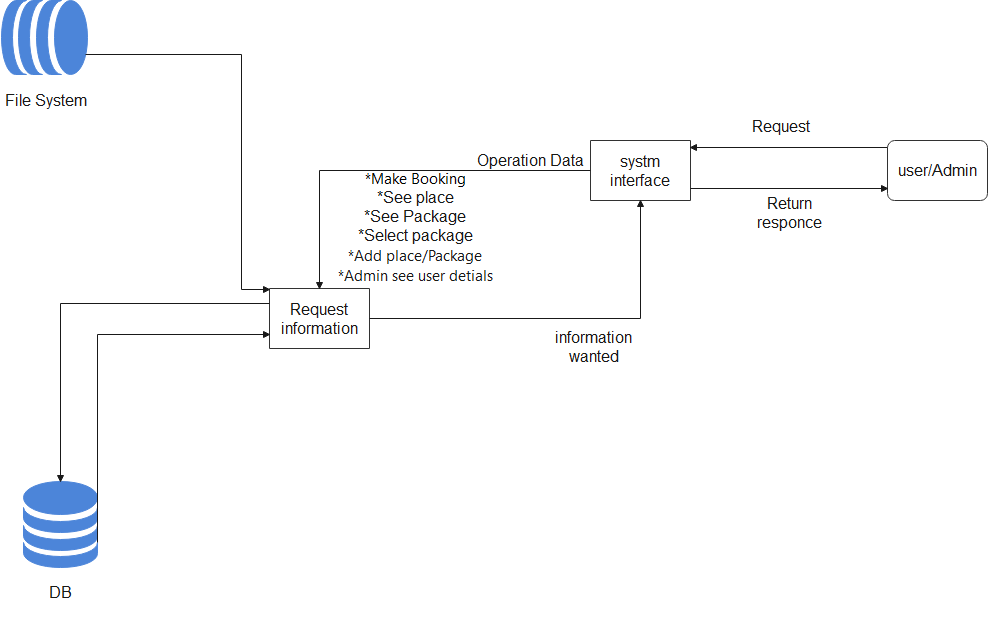
**8.3 Database Specifcation:**



**[8.4]System Architecture:**

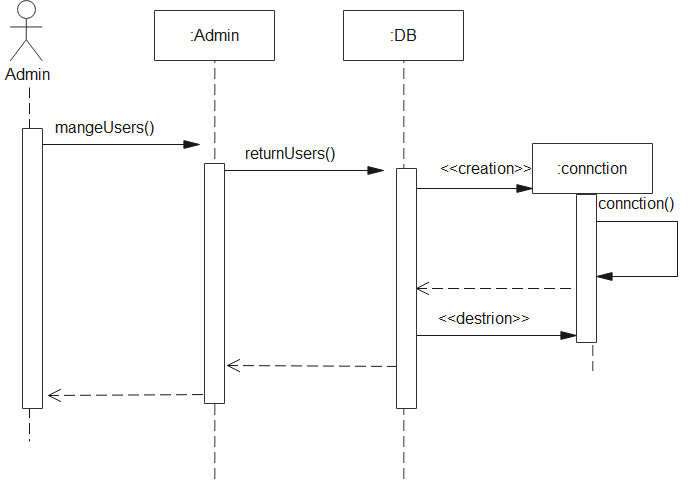


**Architecture Diagram:**

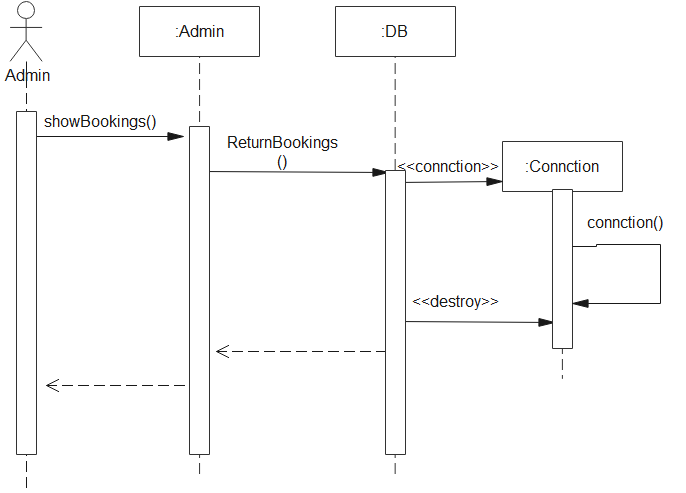


**[8.5]Sequence Diagrams:**

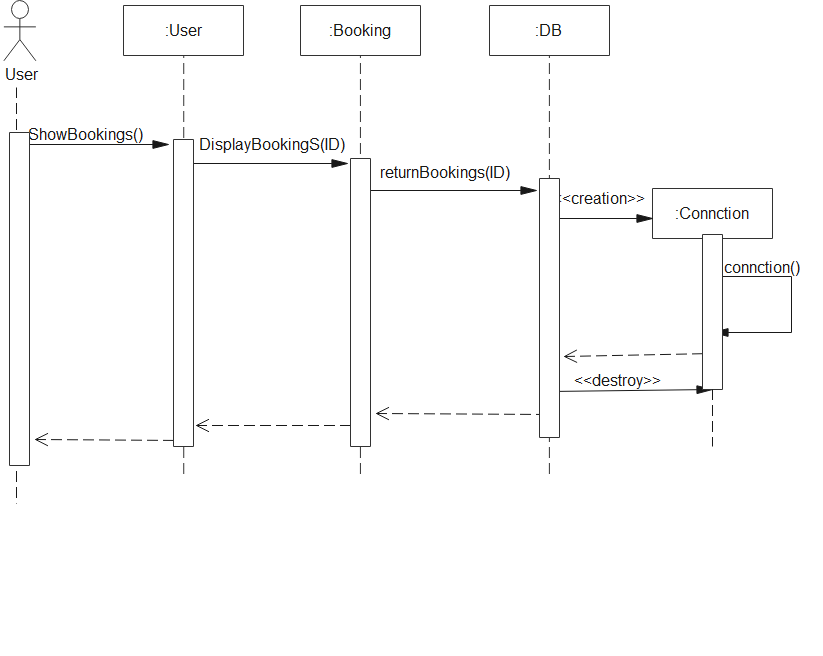
**[8.5.1]Admin mange users:**



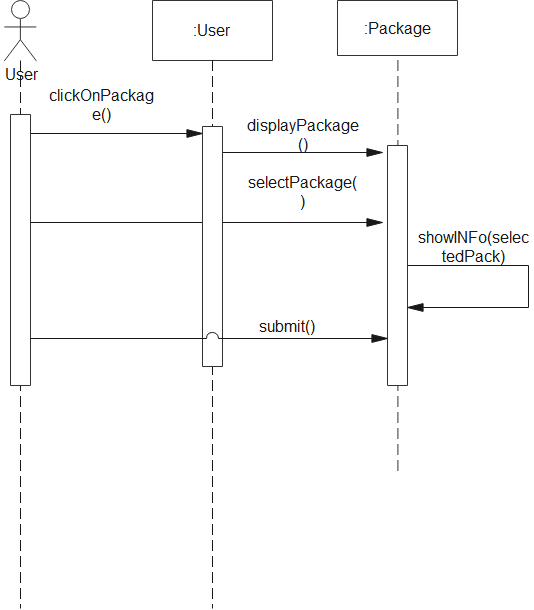
**[8.5.2]Admin mange Bookings:**



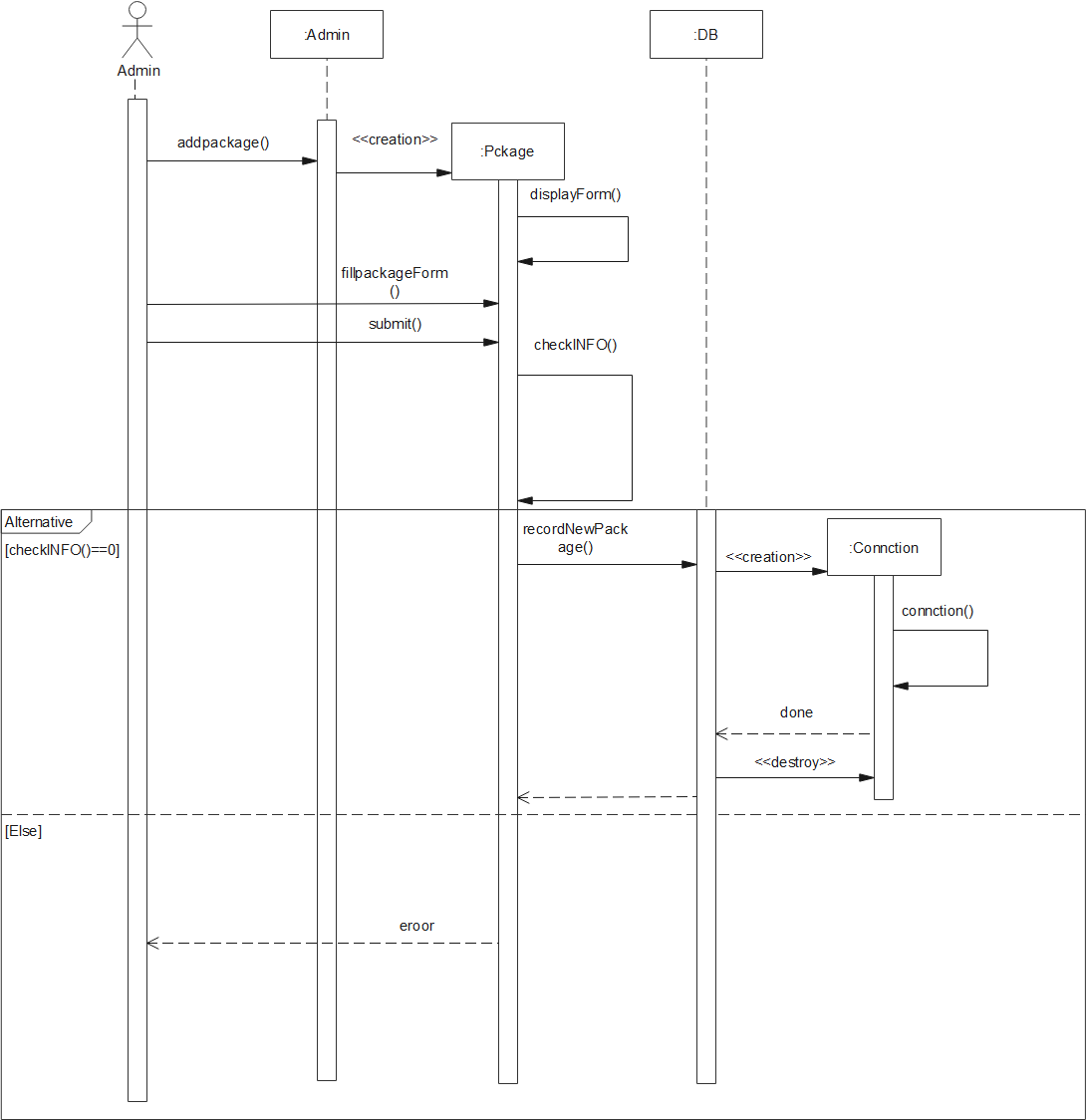
**[8.5.3]User see his bookings:**



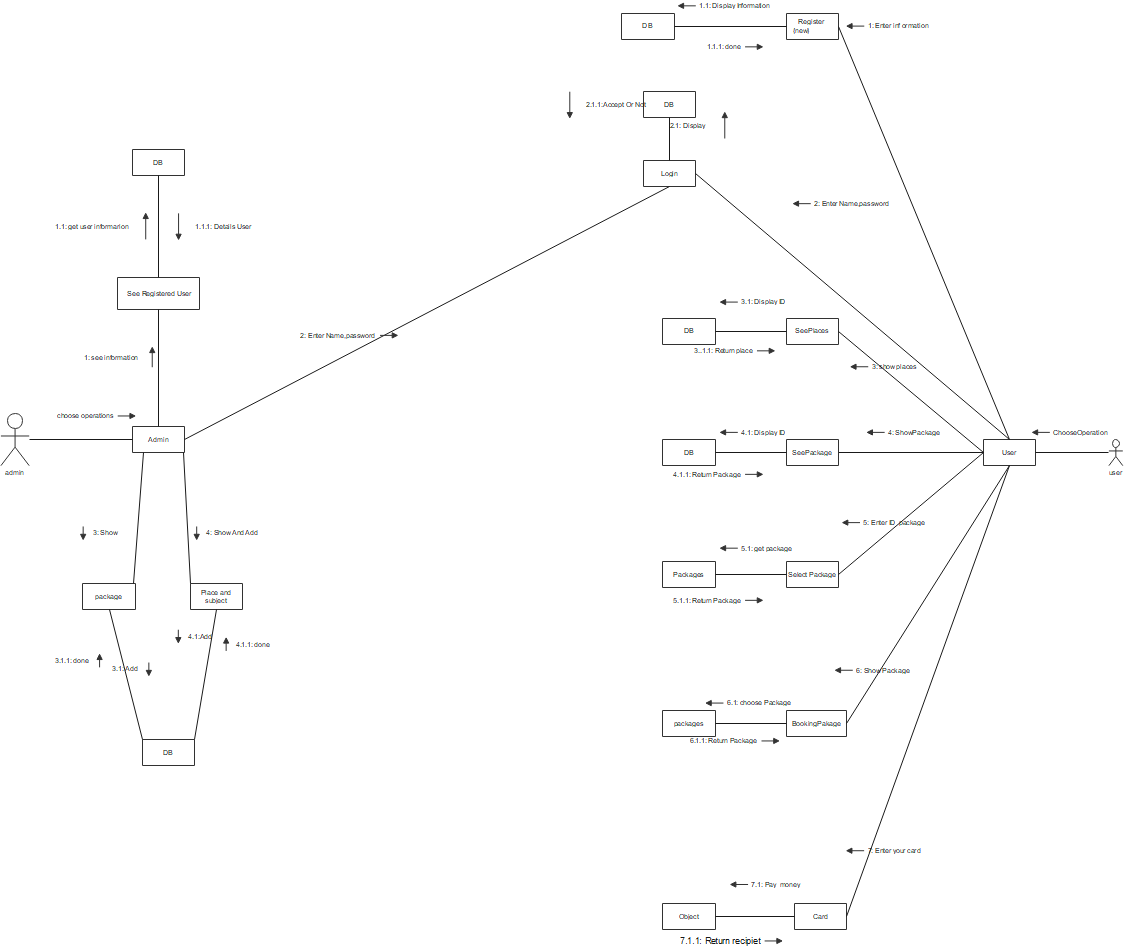
**[8.5.4]User Select package:**



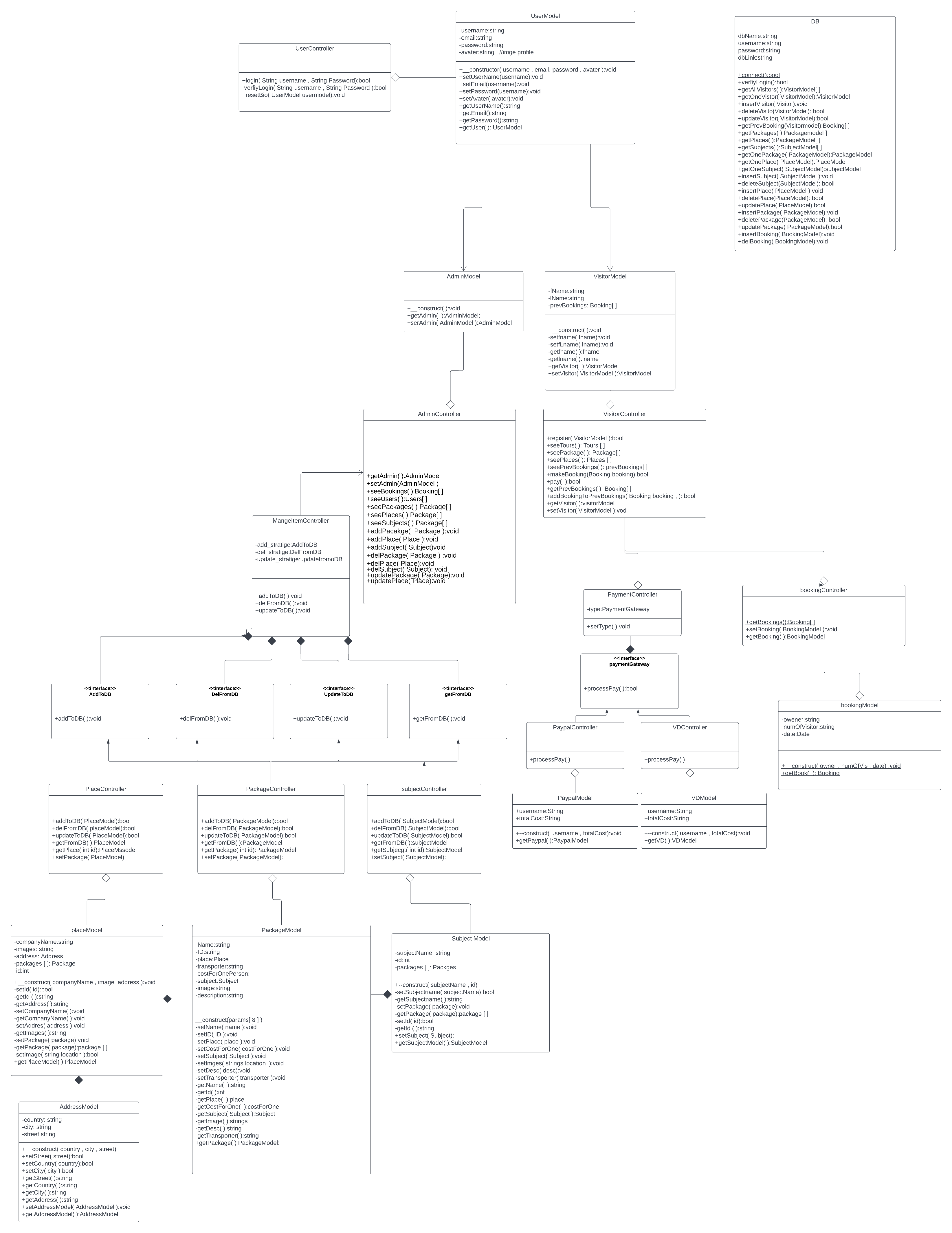
**[8.5.5] add package:**



[8.6]Collaboration Diagrams:



**[8.7]Class Diagrams:**



**[9.0]Design Patterns:**

[9.0.1]First Design pattern:

# Name: Singleton Design Pattern.

# Intent:

**Singleton** is a creational design pattern that lets you ensure that a class has only one instance, while providing a global access point to this instance.

# Problem:

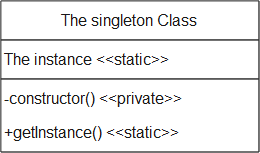
**Database, it is a global class and mustn't be creation more than one instance and any objects of the system can access it.**

# Solution:

* **Make the default constructor private**, to prevent other objects from using the new operator with the Singleton class.
* **Create a static creation method that acts as a constructor.** Under the hood, this method calls the private constructor to create an object and saves it in a static field. All following calls to this method return the cached object.

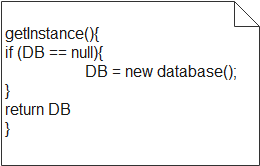
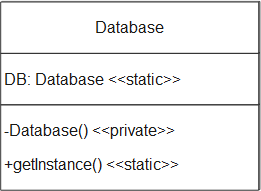
**\*\*If the code has access to the Singleton class(DB), then it’s able to call the Singleton’s static method. So whenever that method is called, the same object is always returned.**

# Structure:



The **Singleton** class declares the static method getInstance() that returns the same instance of its own class.

The Singleton’s constructor should be hidden from the client code. Calling the getInstance method should be the only way of getting the Singleton object.

# Participants:

-The Objects of the system and the Instance of Database class

# Consequences:

The system allows to make one instance of Database Class and share it by all objects of the system.

# Implementation:

**#pesucode**

// The Database class defines the `getInstance` method that lets

// clients access the same instance of a database connection

// throughout the program.

class Database is

// The field for storing the singleton instance should be

// declared static.

private static field instance: Database

// The singleton's constructor should always be private to

// prevent direct construction calls with the `new`

// operator.

private constructor Database() is

// Some initialization code, such as the actual

// connection to a database server.

// ...

// The static method that controls access to the singleton

// instance.

public static method getInstance() is

if (Database.instance == null) then

acquireThreadLock() and then

// Ensure that the instance hasn't yet been

// initialized by another thread while this one

// has been waiting for the lock's release.

if (Database.instance == null) then

Database.instance = new Database()

return Database.instance

// Finally, any singleton should define some business logic

// which can be executed on its instance.

public method query(sql) is

// For instance, all database queries of an app go

// through this method. Therefore, you can place

// throttling or caching logic here.

// ...

class Application is

method main() is

Database foo = Database.getInstance()

foo.query("SELECT ...")

// ...

Database bar = Database.getInstance()

bar.query("SELECT ...")

// The variable `bar` will contain the same object as

// the variable `foo`.

**[9.0.2] second Design pattern:**

# Name: Immutable Design Pattern.

# Intent:

This pattern (Immutable)use to save state of objects as creation and not

change by time.

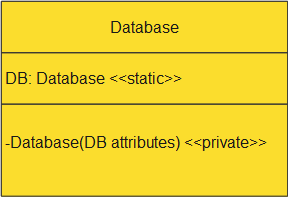
# Problem:

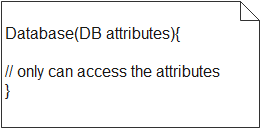
We have a Database Class and his configuration mustn't change as creation instance not modify by the time.

# Solution:

1. Ensures that the constructor of Database Class is only the place where the values of instance variables are set or modified.
2. DB the object of Database class which has methods like access some attributes must not change the DB variables.

# Structure:





# Participants:

The object of Database class and objects that on want to change the state of the object.

# Consequences:

Any objects of the system can change the state of the Database object as creation by the time.

# Implementation:

**trait WithMutable**

**{**

**private $mutable = false;**

**protected function isMutable()**

**{**

**return $this->mutable;**

**}**

**public function withMutable(callable $fn)**

**{**

**$x = clone $this;**

**$x->mutable = true;**

**$x = call\_user\_func($fn, $x);**

**$x->mutable = false;**

**return $x;**

**}**