1

PROJECT PROFILE

* 1. **Project Definition**
  2. **Project Scope**
  3. **Project Objective**

|  |  |  |
| --- | --- | --- |
| Project Profile | | |
|  | **Enrollment No.** | 03052001105  03052001105  03052001102 |
|  | **Student Name** | Timbaliya Rahul  Mihir Vaddoriya  Manish Parmar |
|  | **Project Title** | **India\_Tour** |
|  | **Project Type** | Web Application |
|  | **Tools & Technology** | HTML,CSS,  JAVA-SCRIPT  PYTHON-DJANGO,  MYSQL |
|  | **Front – End** | DJANGO HTML CSS  JAVA SCRIP |
|  | **Back – End** | MYSQL,SQLLITE |
|  | **Team Member** | 3 |

**1.1 Project Definition :-**

* India\_Tour System is a web-based technology that manages tour package and customer to easily book and find information about place
* India\_Tour system is a web-based system which help tourist to see available Package in company and book that package online to save time and money
* Administration can store the data of package . The System also keeps the user book info**.**
* This project uses Django and MYSQL,SQLite databases, and it has a module that is admin. All possible features such as validation, validation, security, etc., are considered.
* The proposed system influences or interacts with Tourist That can find any online service to book its package.
* This project also includes system testing and validation, which ensures that your System is error-free and working as expected.
  1. **Project Scope**
* Using This Web-Based Application user can find out Verity of Tour Packages and Book A Package.
* Using this application that can manage all the tour regarding questions can solve to contact-us page.
* User can register to the site and also unregister from the site.
* Verity of functionality and Validations are provide on this site.
* Best User experience on this site.
* Sort and simple navigate to the site.
  1. **Project Objective**
* Main objective of this web application to make the Tourism process online and save tourist time to offline process
* Help user to easily book an tour package without any physical process
* All the information about the tour package are easily available for end-user
* User can all the time updated about this site
* For user security site provide verity of validations to protect the user data and make site attack protected
* Site also provide Google captcha authentication to protect the site to abnormal activity to any form
* Many more functionality are provided to this sites

2

System AnaLYSis And Speciffication

* 1. **Existing System**
  2. **Limitations Of Existing System**
  3. **Feasibility Study**
  4. **Need Of New System**
  5. **Font End & Back End Tool**
  6. **Existing System**
* In the technical world all the work done by the web based system and internet
* Older System can also provide this type of services and functionality and facility but some enhancement are require
* Older system only provide package information not provide booking so user only gather information about the package and allocate sit by physical booking process and go to the office of particular travelling office
* Existing system are older by technology to not add some new feature to make advancement in the site

* 1. **Limitation Of Existing System**
* Some technology issue are present n the site to not and advancement are done.
* Limitation of technology.
* Some time run time error is occur on the application.
* UI is very older.
* Worst User experience.
* Booking is not Provided On the Existing System to user can not book any package at the site it require to goes to the office and book specific Tour Package
  1. **Feasibility Study**
* After implementing the Zoo Management System project and investigating and analyzing all existing or required features of the System, the next task is to carry out a feasibility study. All projects can be run with unlimited resources and infinite time, and the feasibility study involves considering all possible ways to provide a solution to a given problem. The proposed solution should meet all user requirements and be flexible enough to adapt to future changes based on future needs quickly.

### Economic Feasibility

* This is an essential aspect to consider when developing a project. We chose the technology based on the lowest possible cost factors.
* The organization must bear all hardware and software costs.
* Overall, it is estimated that the benefits that an organization will get from the proposed System will outweigh the initial and subsequent costs

### Technical Feasibility

* This included investigating features, performance, and limitations that could affect the ability to achieve a tolerable system. To demonstrate this proof of concept, we have explored all the features provided by the System as described in the System. Requirements Specification (SRS), and everything is possible with the different types of front-end and backend platforms. I confirmed that there is.
* Operational feasibility Undoubtedly, the proposed System is completely GUI-based, very user-friendly, and all inputs are self-explanatory to the layman. In addition, proper training is provided to convey the essence of the System to the user, and the user can become accustomed to the new System. Our customers are comfortable and happy with our research because the System has reduced stress and strain.
  1. **Need Of New System**
* To overcome older application /system problem
* To Provide New feature in a system
* To enhance user expriance
* To provide new UI to system
* User Easily interact with the system and find out what is want
* To overcome runtime error of a system
* Make the offline booking process online to save the user time and management the tour data
* Make the user experience better
* Enhance the user interaction
  1. **Font End & Back End Tool**
* **Front End Tools**
* **HTML**



* The **HyperText Markup Language** or **HTML** is the standard [markup language](https://en.wikipedia.org/wiki/Markup_language) for documents designed to be displayed in a [web browser](https://en.wikipedia.org/wiki/Web_browser).
* HTML (Hypertext Markup Language) is the only markup language for creating web pages. It provides some titles, headings, paragraphs, lists, tables, embedded images, etc., to describe the structure of text-based and multimedia information in HTML documents.
* HTML is a [markup language](https://en.wikipedia.org/wiki/Markup_language) that [web browsers](https://en.wikipedia.org/wiki/Web_browser) use to interpret and [compose](https://en.wikipedia.org/wiki/Typesetting) text, images, and other material into visual or audible web pages
* HTML is a [markup language](https://en.wikipedia.org/wiki/Markup_language) that [web browsers](https://en.wikipedia.org/wiki/Web_browser) use to interpret and [compose](https://en.wikipedia.org/wiki/Typesetting) text, images, and other material into visual or audible web pages.
* The First Version Of HTMLWas Written BY **Tim-Burners-Lee**
* **October 28, 2014** HTML5 was published as a W3C Recommendation
* **CSS**



* **Cascading Style Sheets** (**CSS**) is a [style sheet language](https://en.wikipedia.org/wiki/Style_sheet_language) used for describing the [presentation](https://en.wikipedia.org/wiki/Presentation_semantics) of a document written in a [markup language](https://en.wikipedia.org/wiki/Markup_language) such as [HTML](https://en.wikipedia.org/wiki/HTML) or [XML](https://en.wikipedia.org/wiki/XML) (including XML dialects such as [SVG](https://en.wikipedia.org/wiki/Scalable_Vector_Graphics), [MathML](https://en.wikipedia.org/wiki/MathML) or [XHTML](https://en.wikipedia.org/wiki/XHTML)) CSS is a cornerstone technology of the [World Wide Web](https://en.wikipedia.org/wiki/World_Wide_Web), alongside HTML and [JavaScript](https://en.wikipedia.org/wiki/JavaScript).
* CSS is said to as the cornerstone design tool of the World Wide Web along with HTML and JavaScript
* CSS level 1, published on 17 December 1996. [Håkon Wium Lie](https://en.wikipedia.org/wiki/H%C3%A5kon_Wium_Lie) and [Bert Bos](https://en.wikipedia.org/wiki/Bert_Bos) are credited as the original developers
* **Java-Script**

****

* JavaScript  often abbreviated to JS, is a [programming language](https://en.wikipedia.org/wiki/Programming_language) that is one of the core technologies of the [World Wide Web](https://en.wikipedia.org/wiki/World_Wide_Web), alongside [HTML](https://en.wikipedia.org/wiki/HTML) and [CSS](https://en.wikipedia.org/wiki/CSS).As of 2022, 98% of [websites](https://en.wikipedia.org/wiki/Website) use JavaScript on the [client](https://en.wikipedia.org/wiki/Client_(computing)) side for [webpage](https://en.wikipedia.org/wiki/Web_page) behavior, often incorporating third-party [libraries](https://en.wikipedia.org/wiki/Library_(computing)).[All major [web browsers](https://en.wikipedia.org/wiki/Web_browser) have a dedicated [JavaScript engine](https://en.wikipedia.org/wiki/JavaScript_engine) to execute the [code](https://en.wikipedia.org/wiki/Source_code) on [users](https://en.wikipedia.org/wiki/User_(computing))' devices..
* JavaScript Was Invented By Brendan Eich in 1995
* It Was Developed For Netscape 2, became the ECMA-262 Standard in 1997
* **Back End Tools**
* **Python-Django**

****

**What is Django?**

* Django is a Python-based web framework which allows you to quickly create web application without all of the installation or dependency problems that you normally will find with other frameworks.
* When you’re building a website, you always need a similar set of components: a way to handle user authentication (signing up, signing in, signing out), a management panel for your website, forms, a way to upload files, etc. Django gives you ready-made components to use**.**

**Why Django?**

* It’s very easy to switch database in Django framework

.

* It has built-in admin interface which makes easy to work with it.
* Django is fully functional framework that requires nothing else.
* It has thousands of additional packages available.
* It is very scalable**.**

**Features of Django**

Versatility of Django

* Django can build almost any type of website. It can also work with any client-side framework and can deliver content in any format such as HTML, JSON, XML etc. Some sites which can be built using Django are wikis, social networks, new sites etc.

Security

* Since Django framework is made for making web development easy, it has been engineered in such a way that it automatically do the right things to protect the website. For example, In the Django framework instead of putting a password in cookies, the hashed password is stored in it so that it can’t be fetched easily by hackers.

Scalability

* Django web nodes have no stored state, they scale horizontally – just fire up more of them when you need them. Being able to do this is the essence of good scalability. Instagram and Disqus are two Django based products that have millions of active users, this is taken as an example of the scalability of Django.

Portability

* All the codes of the Django framework are written in Python, which runs on many platforms. Which leads to run Django too in many platforms such as Linux, Windows and Mac OS.
* **Mysql (Database)**



* MySQL is currently the most popular database management system software used for managing the relational database. It is open-source database software, which is supported by Oracle Company. It is fast, scalable, and easy to use database management system in comparison with Microsoft SQL Server and Oracle Database. It is commonly used in conjunction with [PHP](https://www.javatpoint.com/php-tutorial) And Other scripts for creating powerful and dynamic server-side or web-based enterprise applications.
* It is developed, marketed, and supported by MySQL AB, a Swedish company, and written in [C programming language](https://www.javatpoint.com/c-programming-language-tutorial) and [C++ programming language](https://www.javatpoint.com/cpp-tutorial). The official pronunciation of MySQL is not the My Sequel; it is My Ess Que Ell. However, you can pronounce it in your way. Many small and big comanies use MySQL. MySQL supports many Operating Systems like [Windows](https://www.javatpoint.com/windows), [Linux](https://www.javatpoint.com/linux-tutorial), MacOS, etc. with C, C++, and [Java languages](https://www.javatpoint.com/java-tutorial).
* **Sqlite (Database)**

****

* SQLite is an in-process library that implements a self-contained, serverless, zero-configuration, transactional SQL database engine. It is a popular choice as an embedded database for local/client storage in application software such as web browsers. It is also used in many other applications that need a lightweight, embedded database.
* SQLite is ACID-compliant and implements most of the SQL standards, using a dynamically and weakly typed SQL syntax that does not guarantee domain integrity.
* To use SQLite in a C/C++ program, you can use the sqlite3 API, which provides a lightweight, simple, self-contained, high-reliability, full-featured, and SQL database engine. The API is implemented as a library of C functions that can be called from your program. One of the main benefits of using SQLite is that it is very easy to get started with. To create a new database in SQLite, you simply need to create a new file on your filesystem and connect to it using the sqlite3 API. For example, in C:

3

System Requirement Specification

**3.1 Proposed System And Advantage**

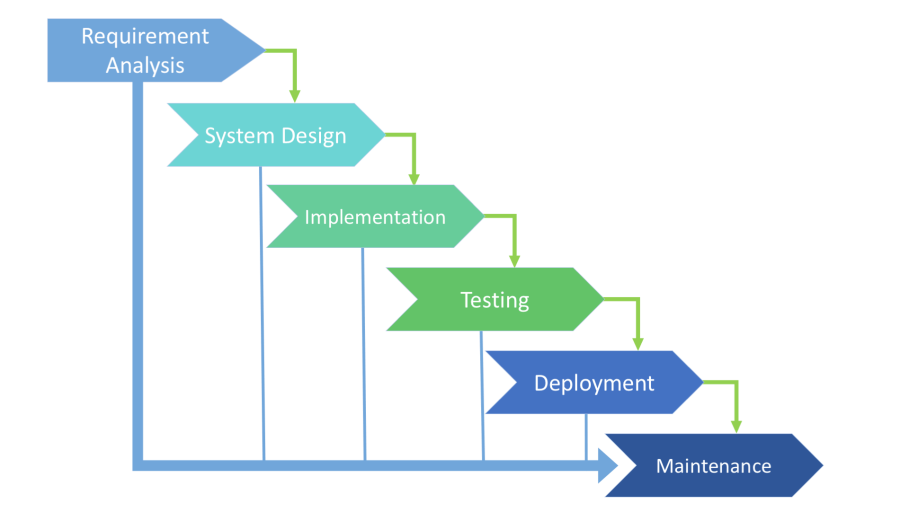
**3.2 Development Strategy Model**

* 1. **Proposed System And Advantage**

* Proposed System Is Very Simple and easy to use
* System also include security to authenticate the user
* New look and UI including with dark mode facility
* Google captcha and Email authentication is provide to user
* Reliable and less memory used
* Cross Platform independent

**Advantages**

* This system Very Used To find online Tour booking system user easily interact with system and collect all the information about its tour package or if any package is related to its choice it can book that package and save its time to offline process
* System Also some security feature like email authentication ,Google captcha authentication to user data will be secured from the hacker attack
* Best user interface to user easily interact with the application any easily to navigate the website
* Light mode and dark mode provide into the application to interact the user interaction with the system
* Not only booking user can generate its confirmed package booking invoice in PDF format
  1. **Development Strategy Model**



**Phase 1: Requirement collection and analysis**

* The requirement is the first stage in the SDLC process. It is conducted by the senior team members with inputs from all the stakeholders and domain experts in the industry. Planning for the quality assurance requirements and recognization of the risks involved is also done at this stage.

### Phase 2: Feasibility study

* Once the requirement analysis phase is completed the next sdlc step is to define and document software needs. This process conducted with the help of ‘Software Requirement Specification’ document also known as ‘SRS’ document. It includes everything which should be designed and developed during the project life cycle

### Phase 3: Design

* In this third phase, the system and software design documents are prepared as per the requirement specification document. This helps define overall system architecture.

### Phase 4: Coding

* Once the system design phase is over, the next phase is coding. In this phase, developers start build the entire system by writing code using the chosen programming language. In the coding phase, tasks are divided into units or modules and assigned to the various developers. It is the longest phase of the Software Development Life Cycle process.

### Phase 5: Testing

* Once the software is complete, and it is deployed in the testing environment. The testing team starts testing the functionality of the entire system. This is done to verify that the entire application works according to the customer requirement.

### Phase 6: Installation/Deployment

* Once the software testing phase is over and no bugs or errors left in the system then the final deployment process starts. Based on the feedback given by the project manager, the final software is released and checked for deployment issues if any.

### Phase 7: Maintenance

* Once the system is deployed, and customers start using the developed system, following 3 activities occur
* Bug fixing – bugs are reported because of some scenarios which are not tested at all
* Upgrade – Upgrading the application to the newer versions of the Software
* Enhancement – Adding some new features into the existing software

4

System DESIGN

**4.1 Data Flow Diagram**

**4.2 E-R Diagram**

**4.3 Class Diagram**

**4.4 Use Case Diagram**

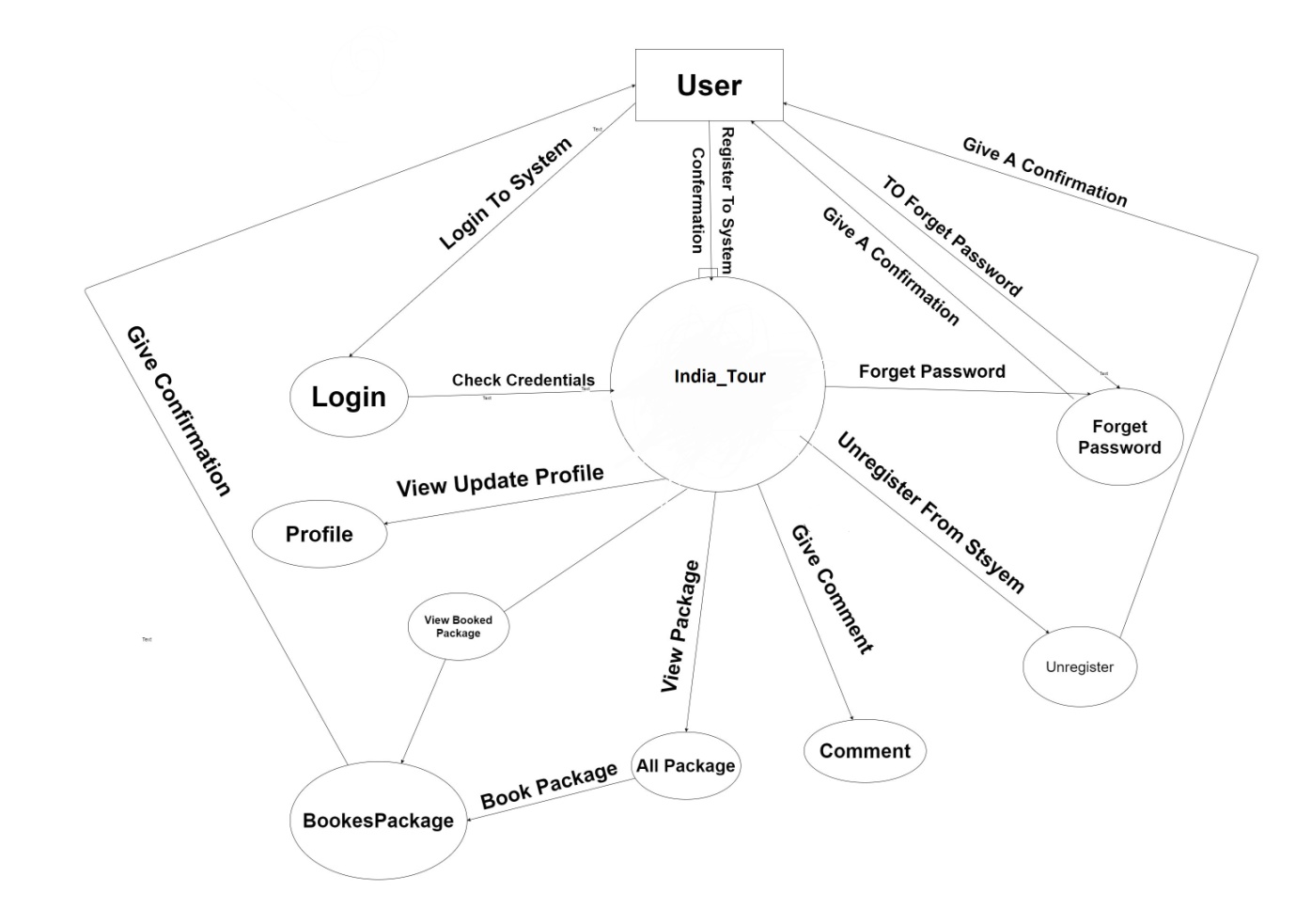
**4.5 Sequence Diagram**

**4.6 Activity Diagram**

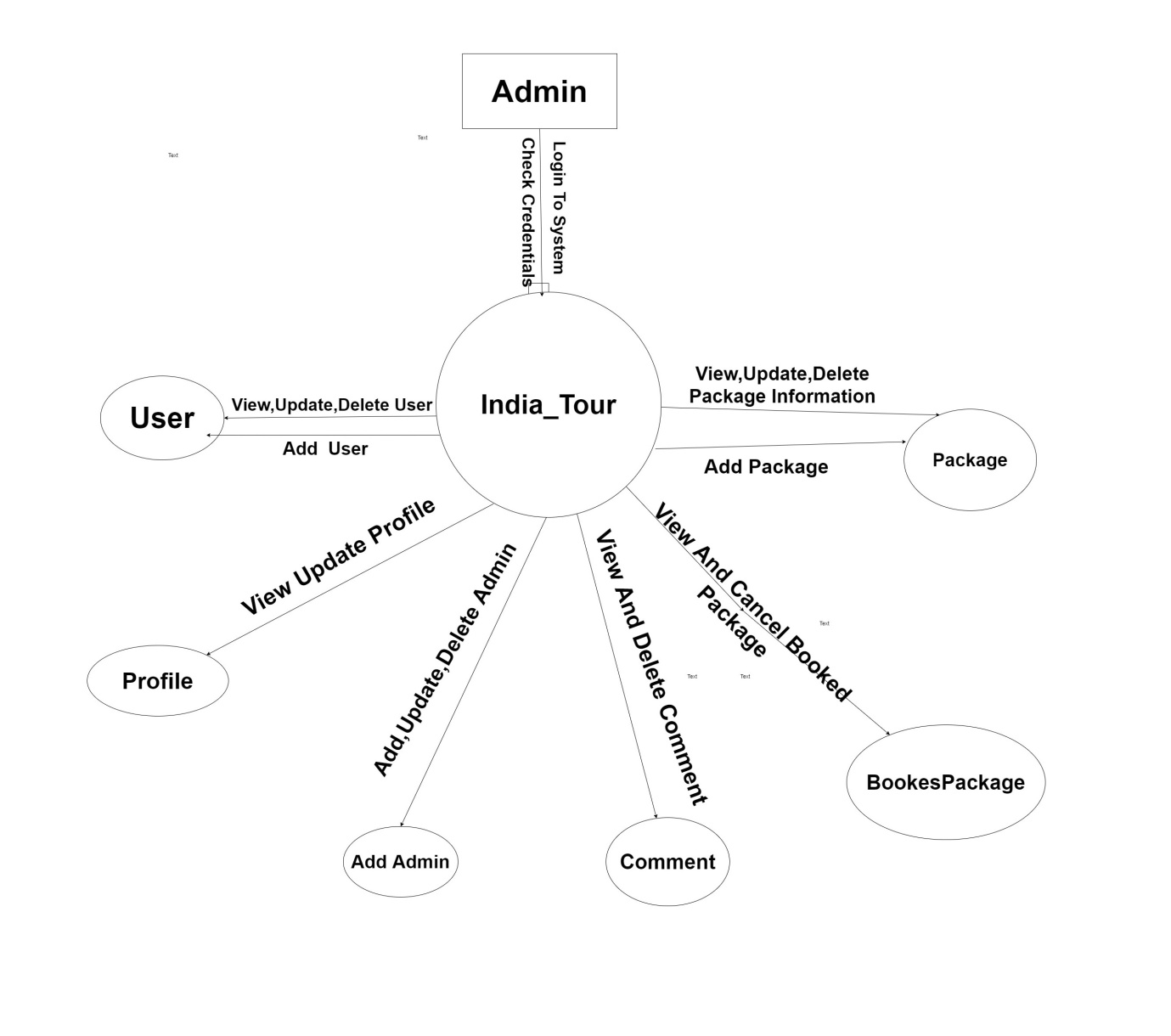
**4.7 Data Dictionary(Database)**

**4.1 Data Flow Diagram**

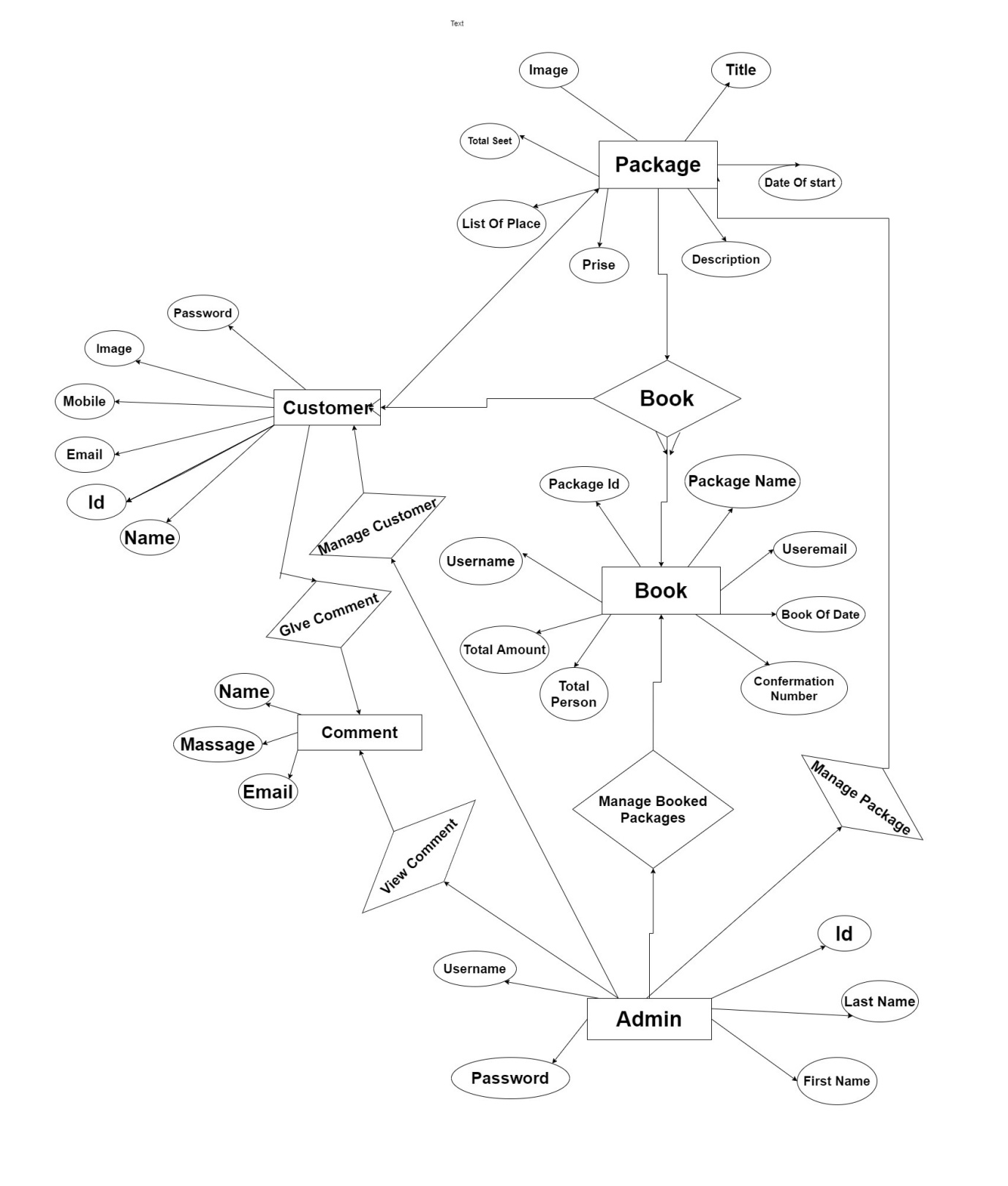
* **User Module DFD**

****

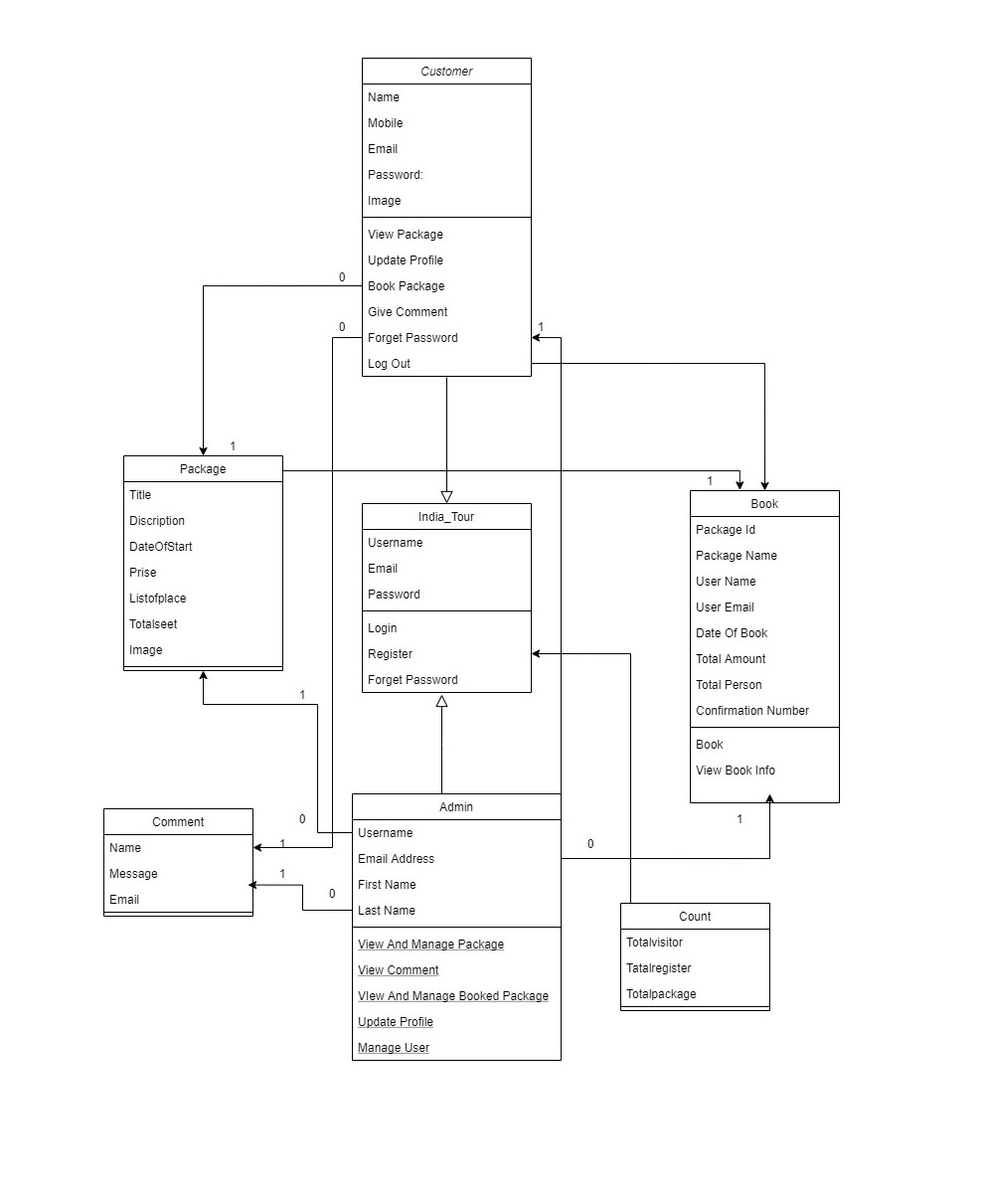
* **Admin Module DFD**

****

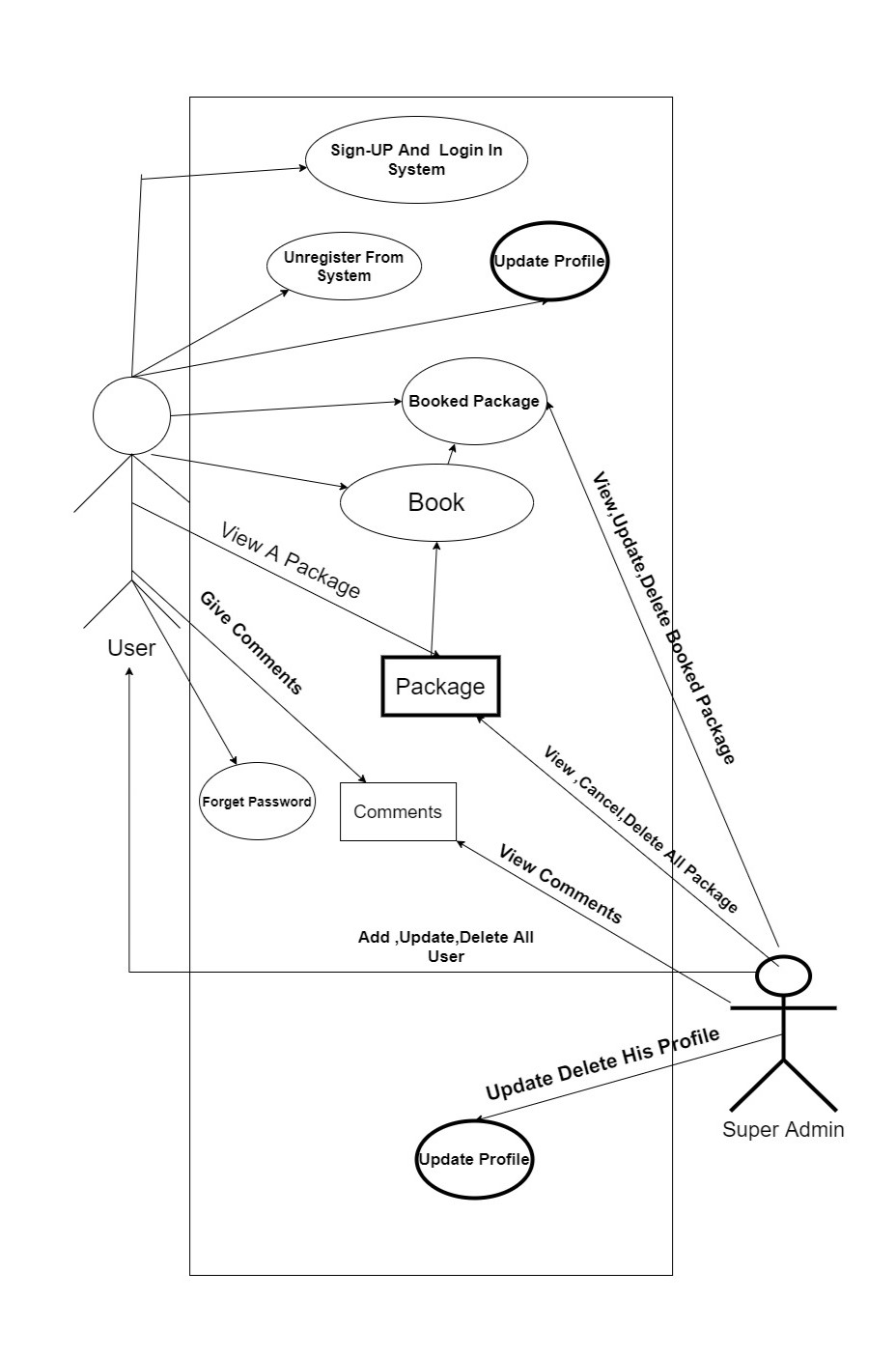
**4.2 E-R Diagram:**



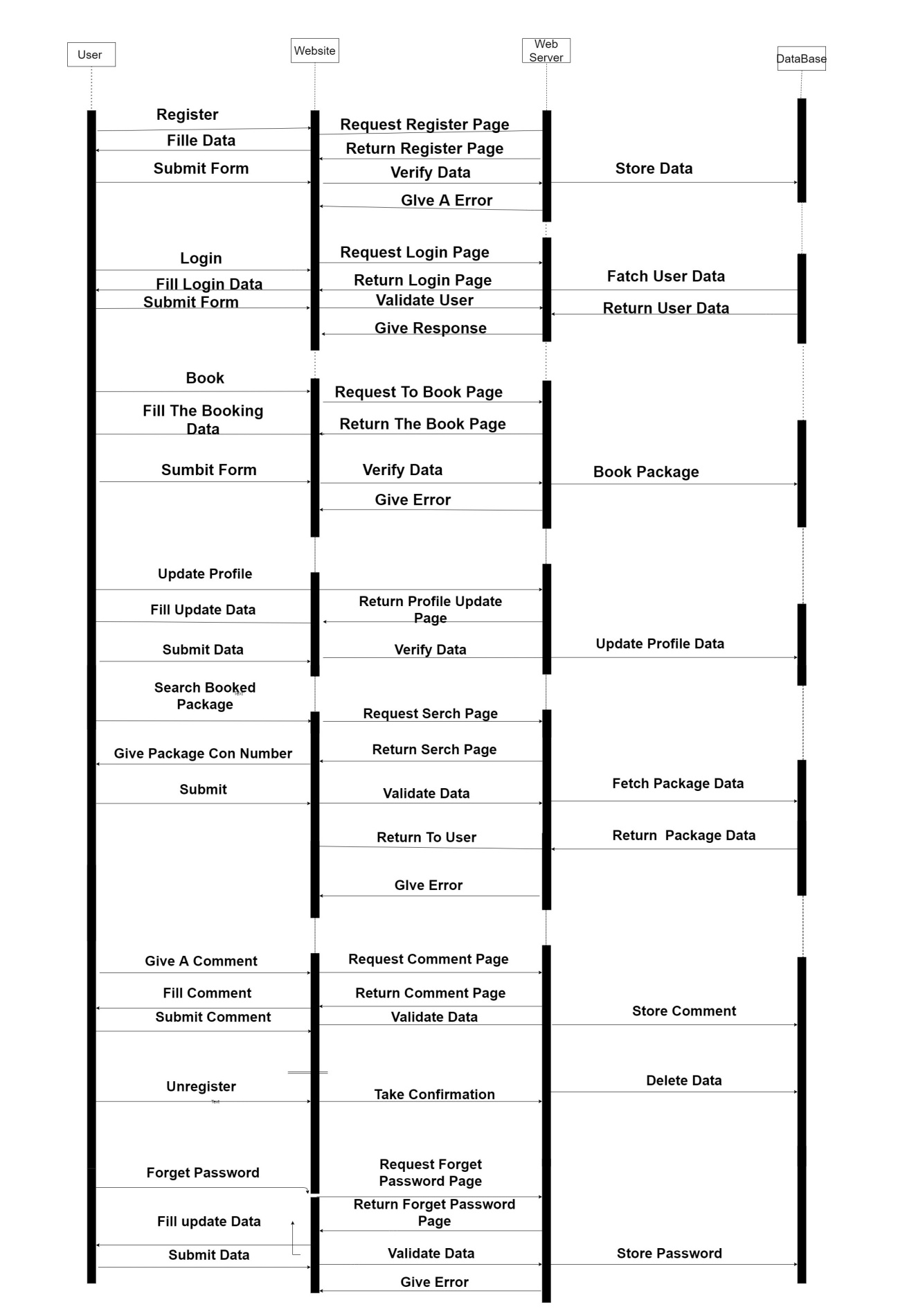
**4.3 Class Diagram**

****

**4.4 Use Case Diagram**

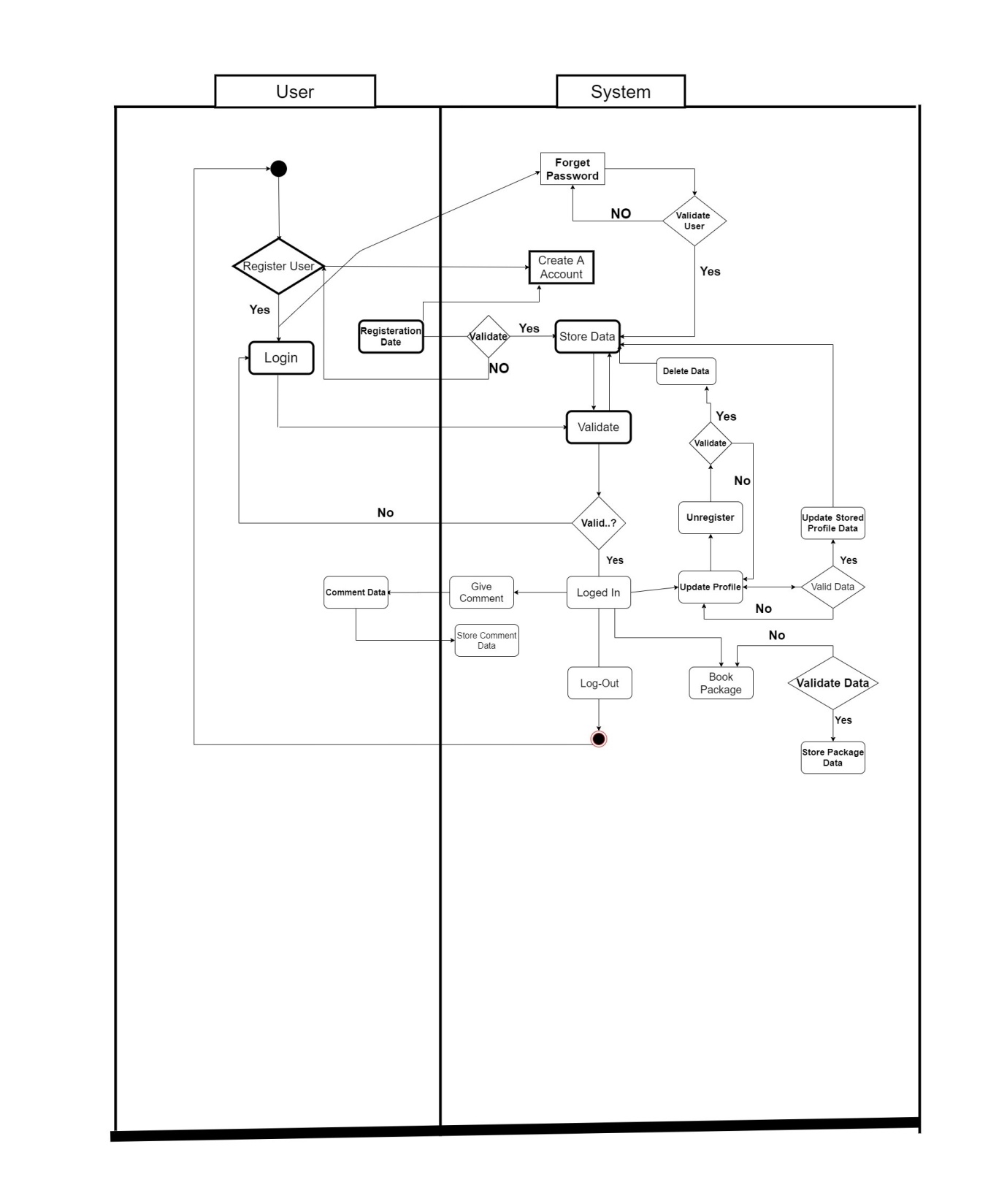
****

**4.5 Sequence Diagram**

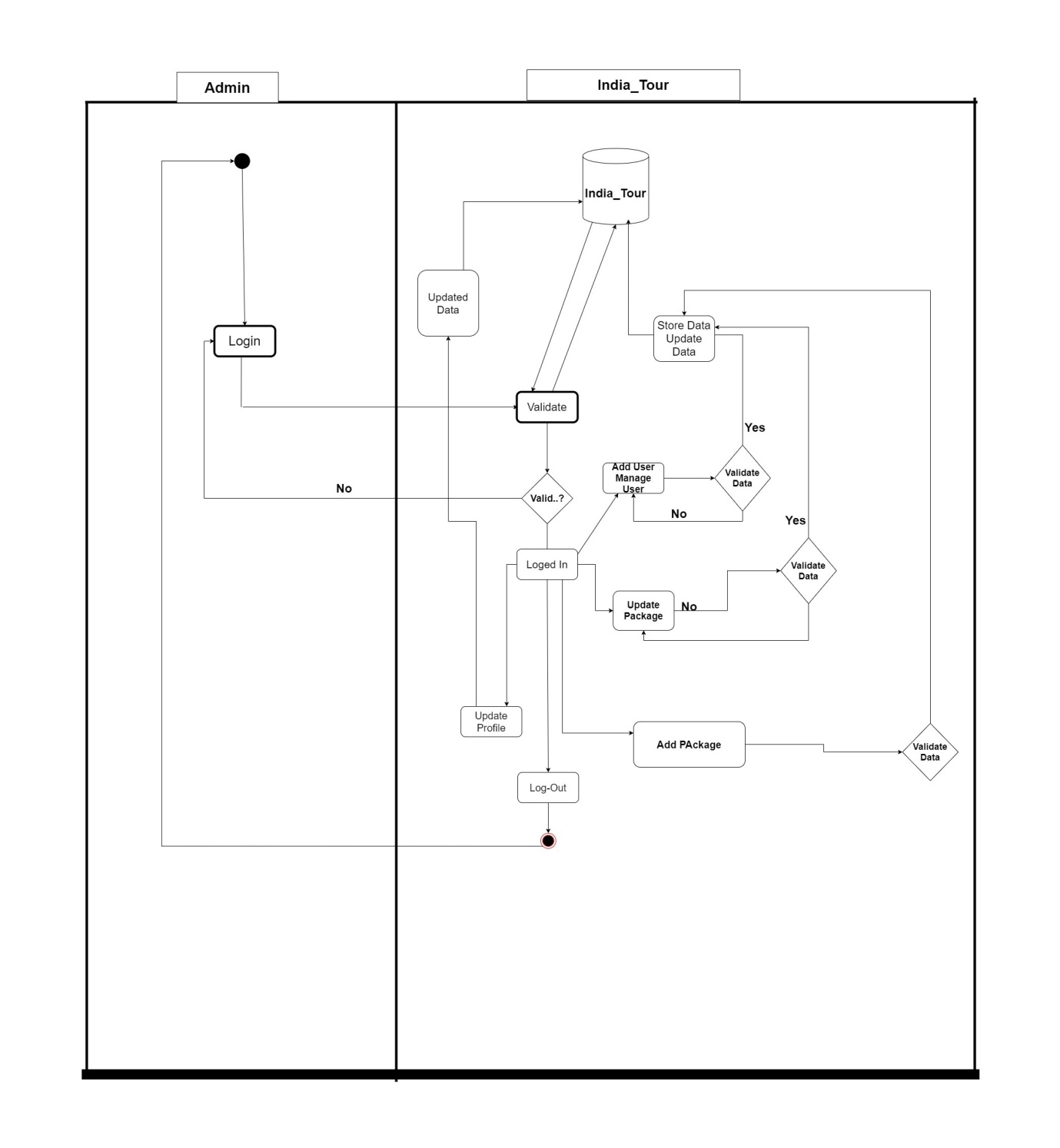
****

**4.6 Activity Diagram**

* **User Activity Diagram**

****

* **Admin Activity Diagram**

****

**4.7 Data Dictionary (Database)**

* **Customer:**

|  |  |  |  |
| --- | --- | --- | --- |
| No | Name | Type | Constrain |
| 1 | ID | Integer(100) | Not Null  Auto Increment |
| 2 | Name | Varchar(255) | - |
| 3 | Email | Varchar(225) | - |
| 4 | Mobile | Varchar(225) | - |
| 5 | Image | Varchar(255) | - |

* **Users:**

|  |  |  |  |
| --- | --- | --- | --- |
| No | Name | Type | Constrain |
| 1 | ID | Integer(50) | Not Null  Auto Increment |
| 2 | Username | Varchar (225) | - |
| 3 | Password | Varchar(255) | - |
| 4 | Email Address | Varchar(225) | - |
| 5 | First Name | Varchar(225) | - |
| 6 | Last Name | Varchar(20) | - |

* **Contact :**

|  |  |  |  |
| --- | --- | --- | --- |
| No | Name | Type | Constrain |
| 1 | Full Name | Varchar(50) | - |
| 2 | Email | Varchar(50) | - |
| 3 | Massage | Varchar(350) |  |

* **Package:**

|  |  |  |  |
| --- | --- | --- | --- |
| No | Name | Type | Not null |
| 1 | Id | Integer(255) | Not Null  Auto increment |
| 2 | Title | Varchar(255) | - |
| 3 | Description | Varchar (225) | - |
| 4 | Date Of Start | Varchar(255) | - |
| 5 | Prise | Varchar(255) |  |
| 6 | List Of Place | Varchar(255) | - |
| 7 | Total Seet | Varchar(225) |  |
| 8 | Image | Image(225) |  |

* **Book :**

|  |  |  |  |
| --- | --- | --- | --- |
| No | Name | Type | Constrain |
| 1 | Id | Integer (100) | Not Null  Auto Increment |
| 2 | PackageId | Varchar(225) | - |
| 3 | Package Name | Varchar(225) | - |
| 4 | User Name | Varchar (225) | - |
| 5 | User Email | Varchar (225) | - |
| 6 | Date Of Book | Date | - |
| 7 | Total Amount | Varchar (225) | - |
| 8 | Total Person | Varchar(225) | - |
| 9 | Confirmation Number | Varchar (225) | - |

* **Count:**

|  |  |  |  |
| --- | --- | --- | --- |
| No | Name | Type | Constrain |
| 1 | ID | Integer  (50) | Not Null  Auto Increment |
| 2 | Total Visitor | Varchar(225) | - |
| 3 | Total Regiaster | Varchar(225) | - |
| 4 | Total Package | Varchar(225) | - |

5

**Screen Design**

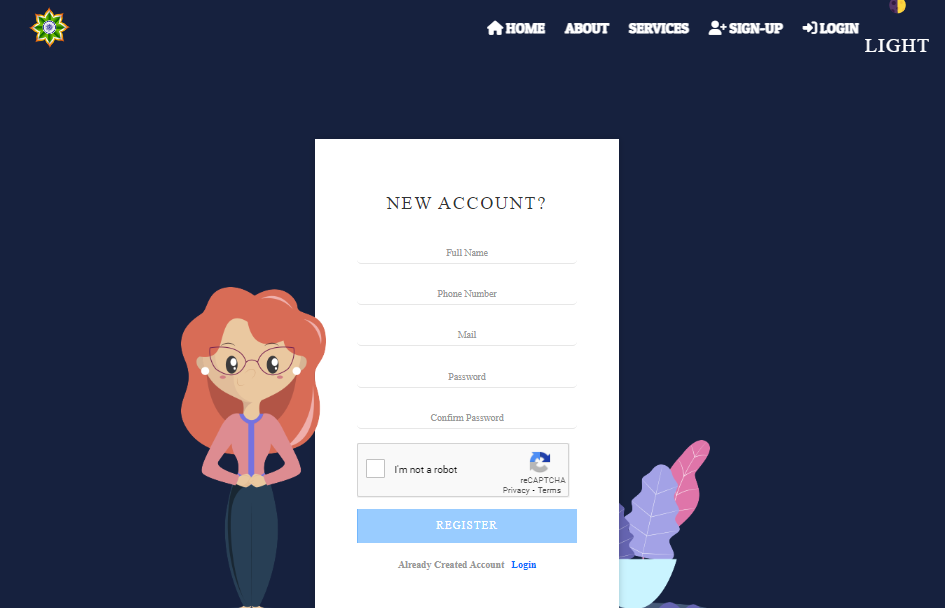
**5.1 Input Screen**

**5.2 Output Screen**

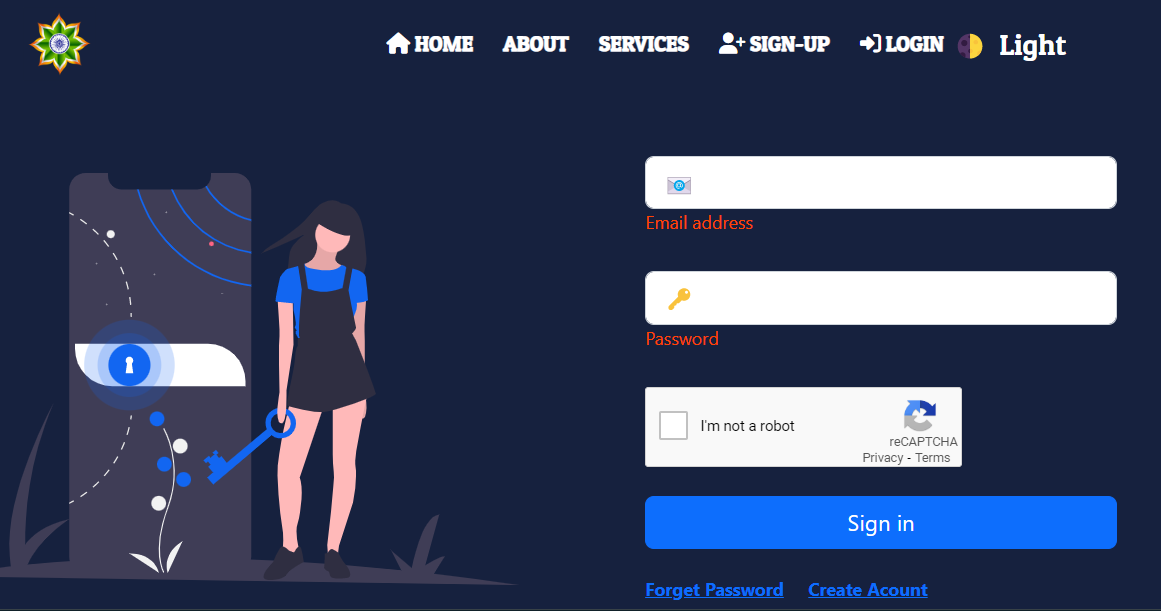
**5.3 Report Screen**

**5.1 Input Screen**

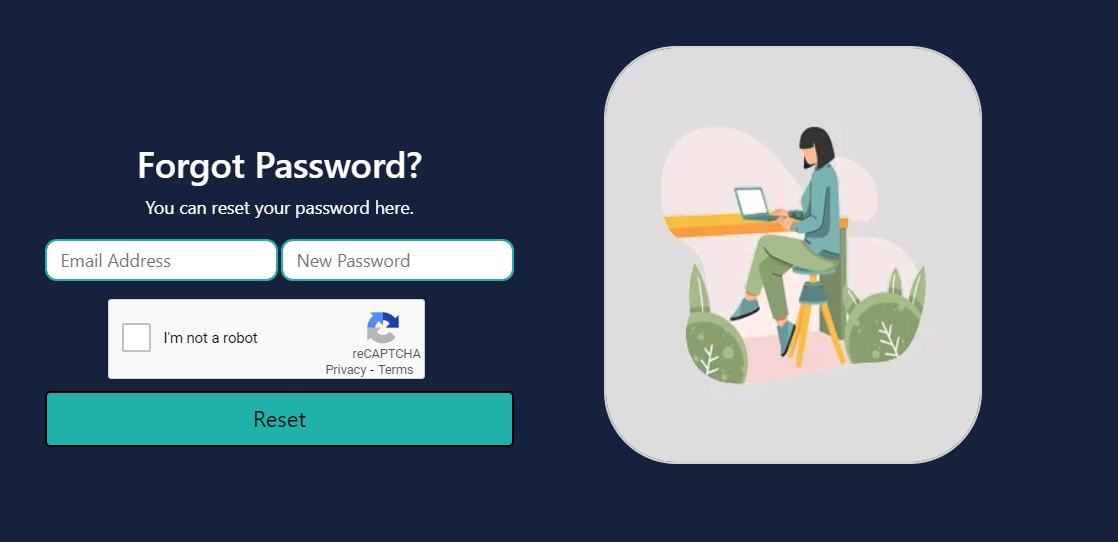
* **SignUp Page.**

****

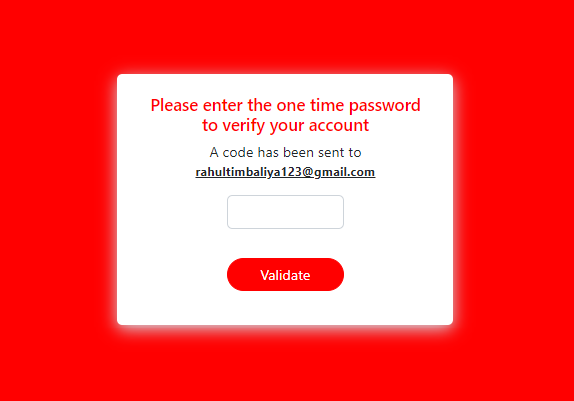
* **Login Page:**

****

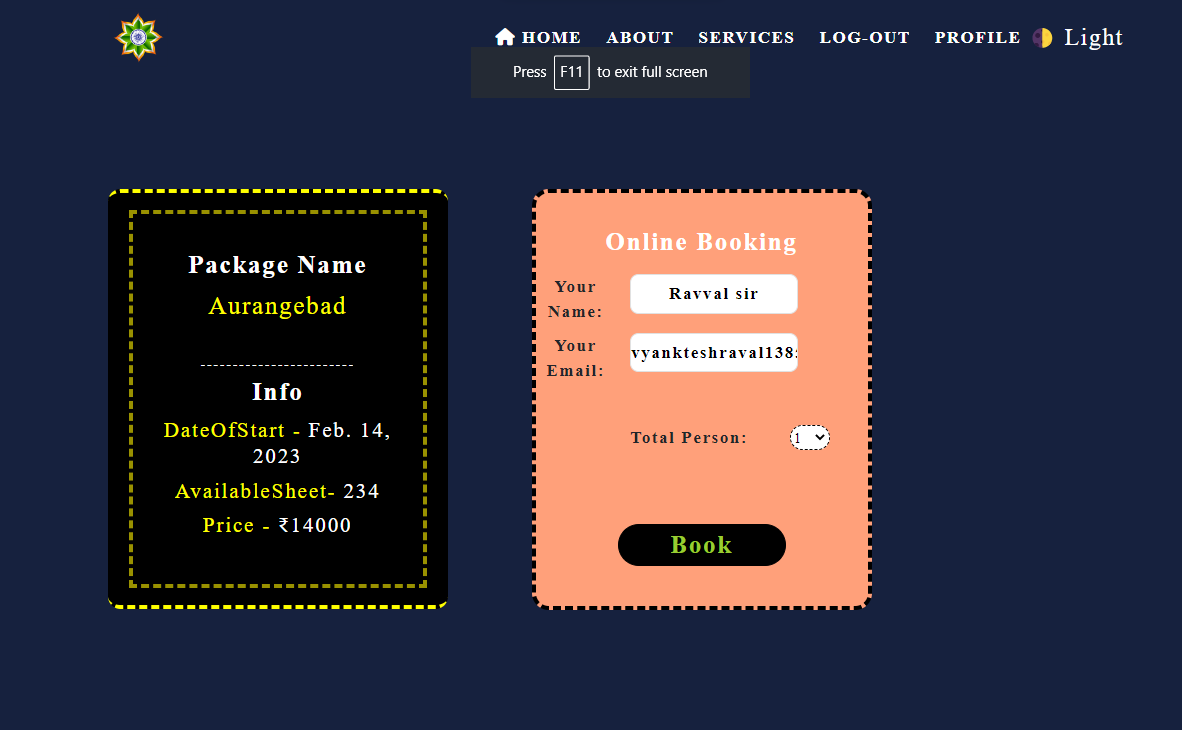
* **Forget Password Page:**



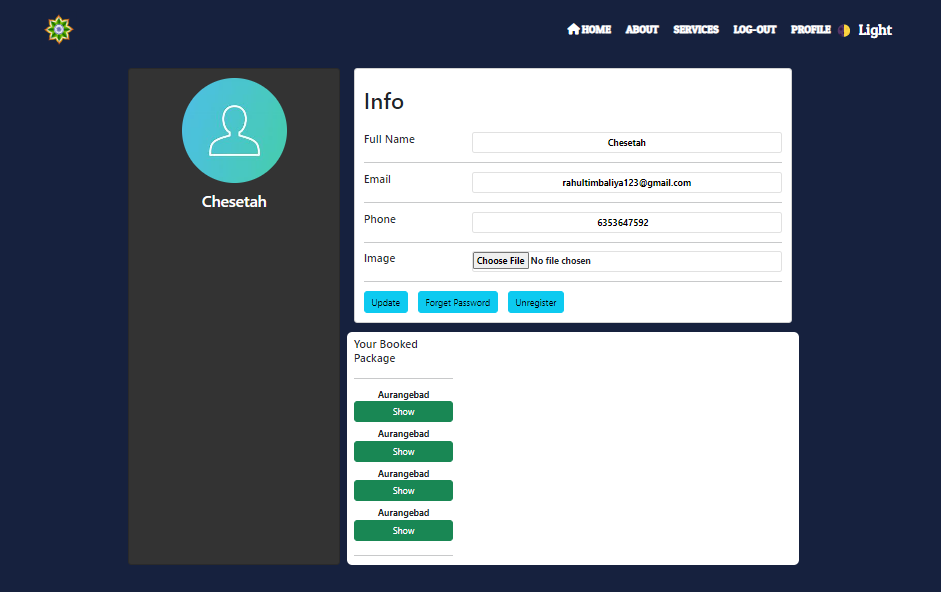
## Email Otp Page:

****

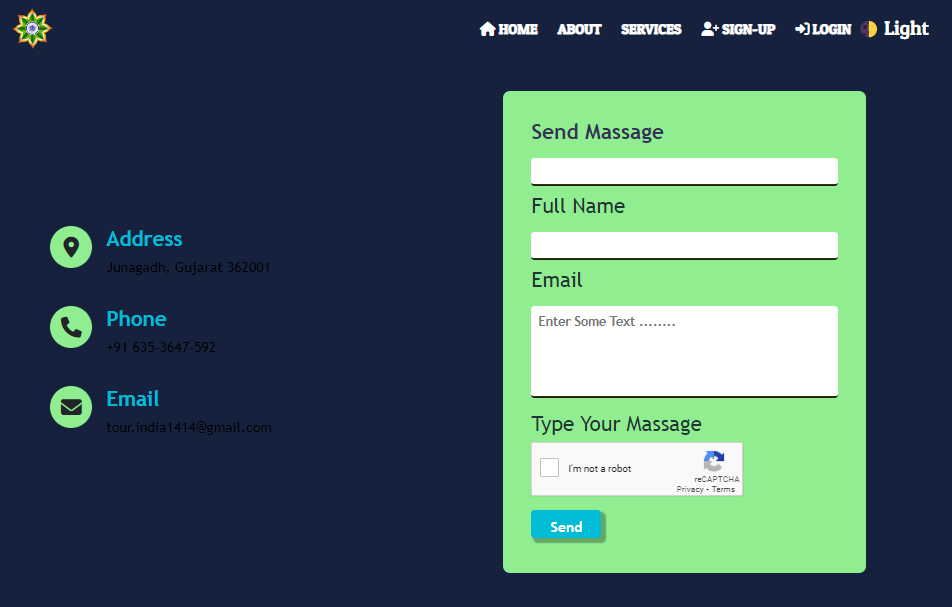
## Book Page:

****

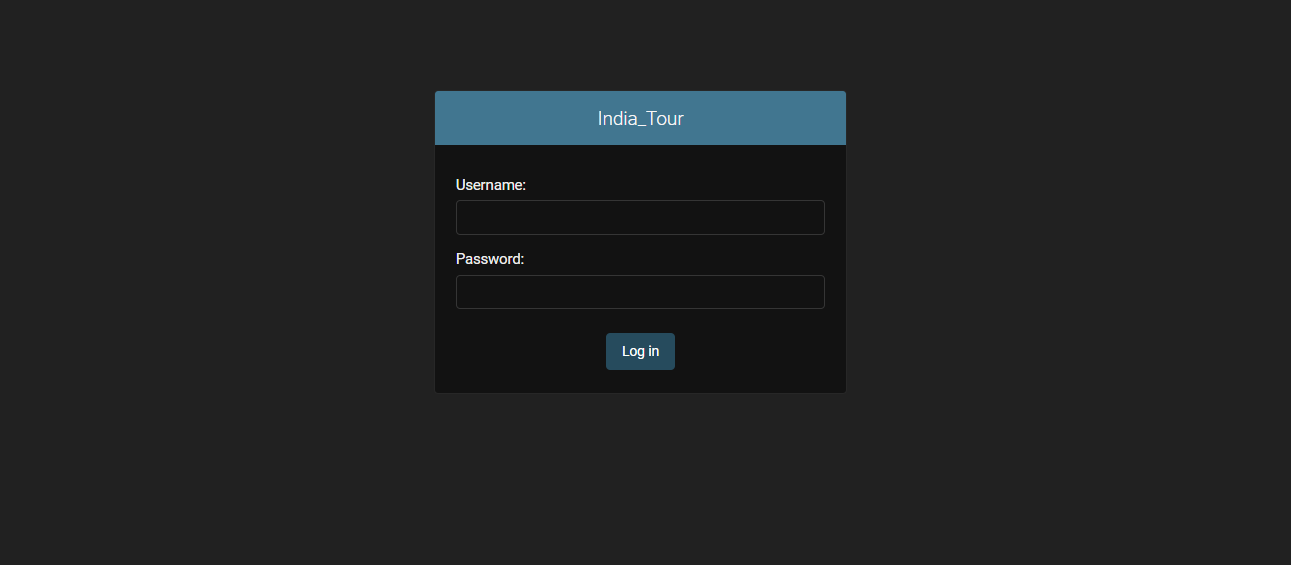
* + - **Profile Page:**

****

* + - **Contact Page:**

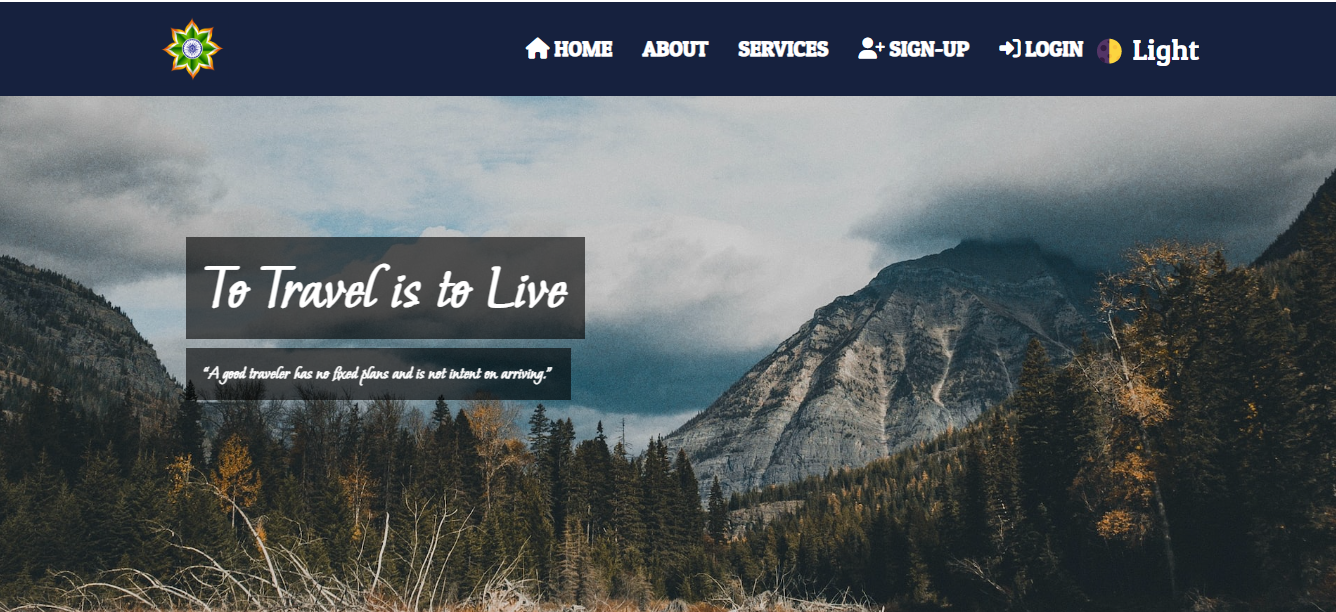
****

## Admin Login:

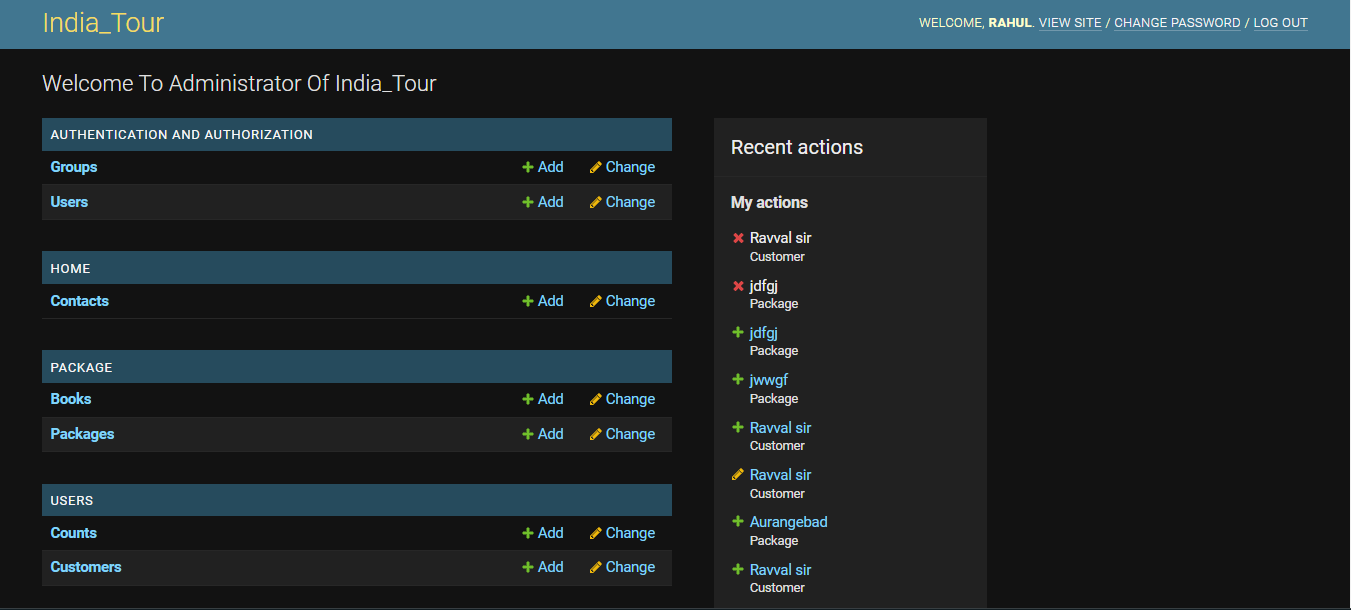
****

**5.1 Output Screen**

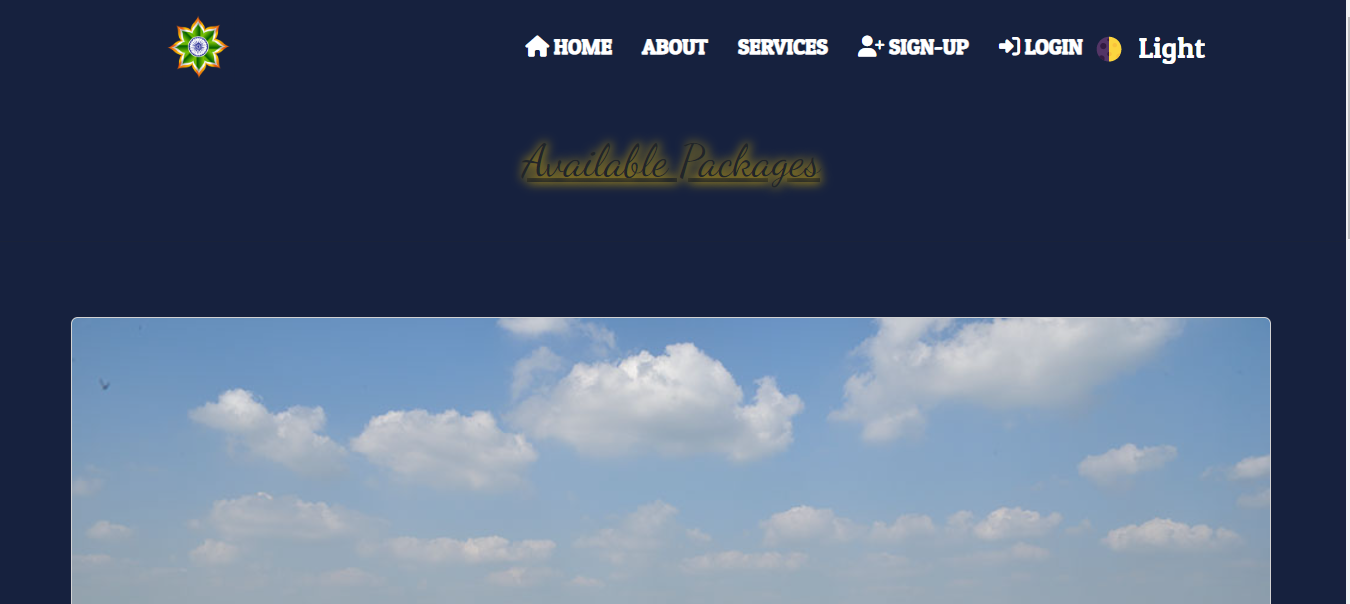
## IndexPage:

****

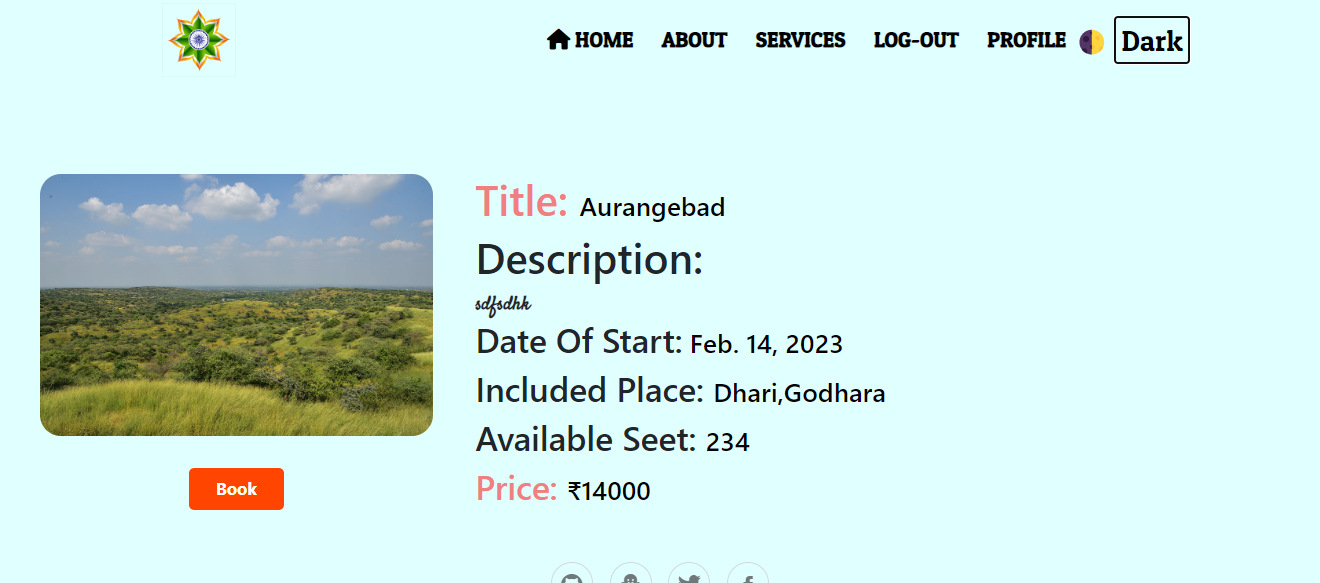
## Admin Dashboard:

****

## Service Page:

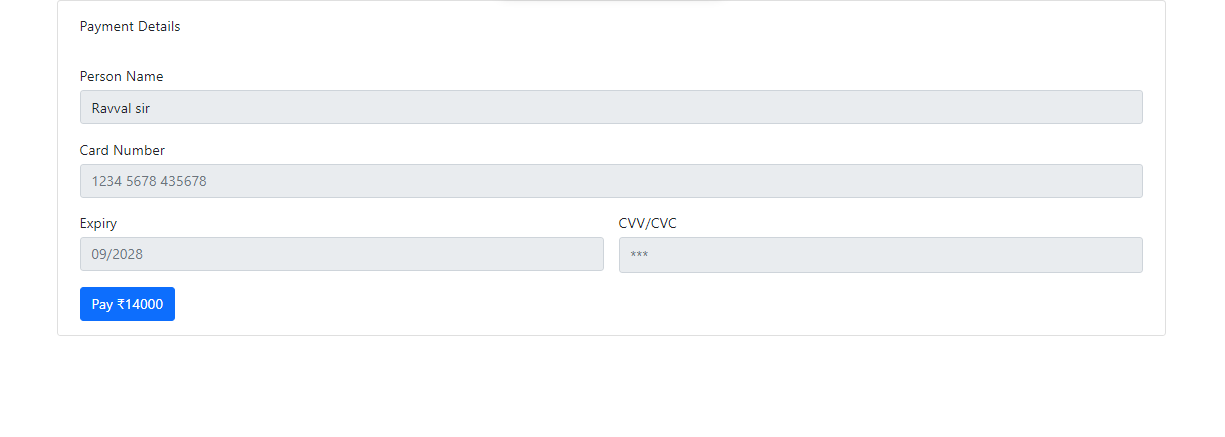
****

## Package Page:

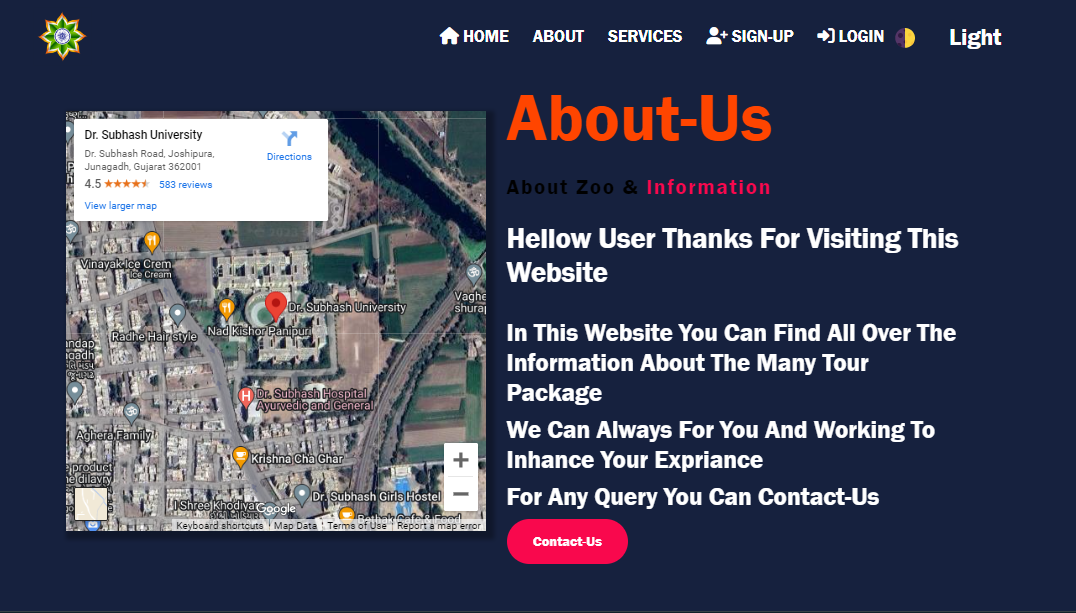


## S

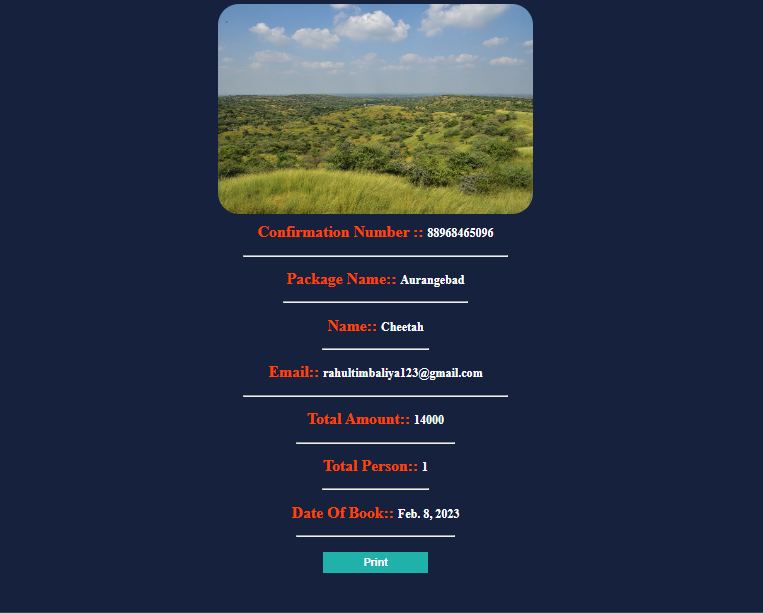
## Payment Page:



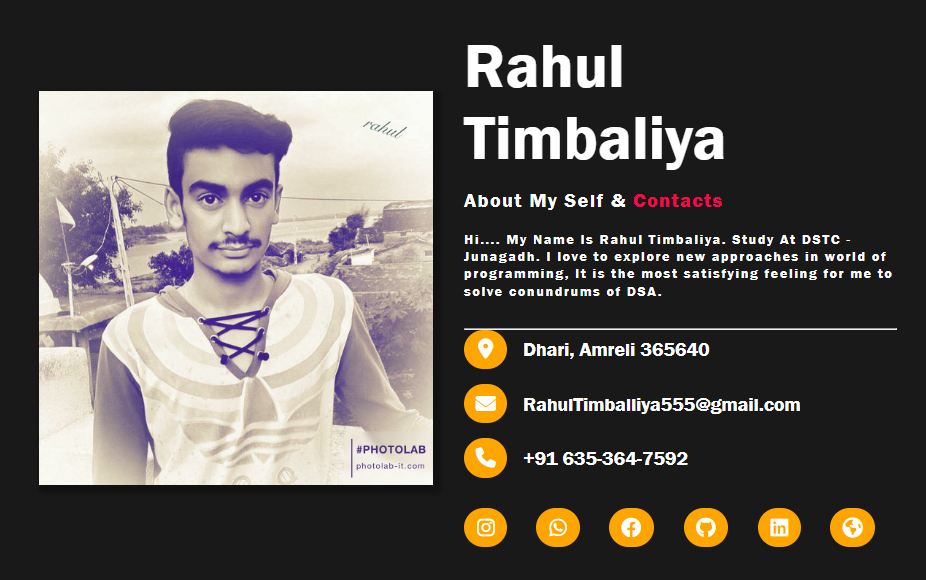
## About Page:



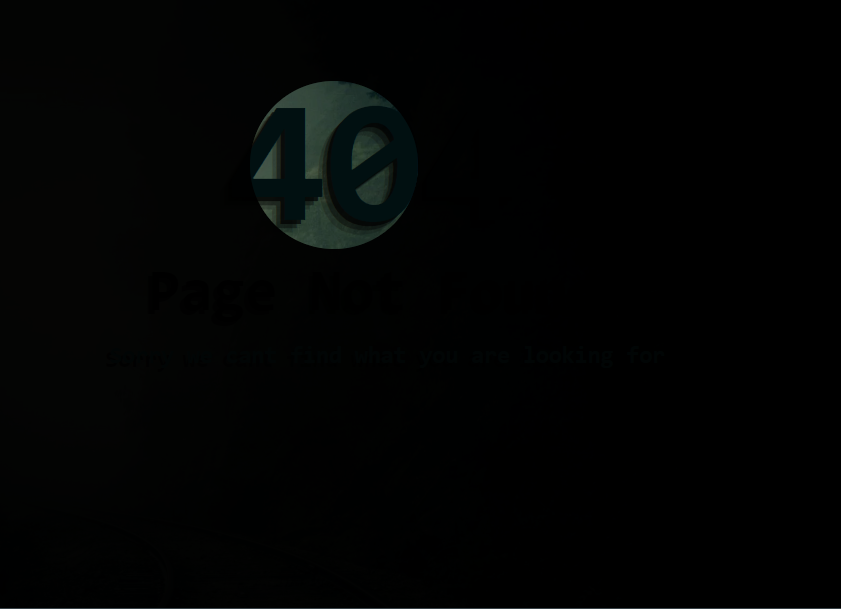
## Package Detail Page:

****

## Developer Info Page:



## 404 Error Page:



**5.3 Report Screen**

****

6

**Coding**

* **View.py**

from django.shortcuts import render, HttpResponse, redirect

from users.models import Customer,Count

from package.models import Book

from django.contrib import messages

import requests

import json

from django.core.mail import send\_mail,EmailMultiAlternatives

import random

name=""

mail=""

passw=""

image=""

phone=""

otp2=""

otpr2=""

emailr=""

rpwd=""

otpd2=""

def signup(request):

request.session['id']=""

request.session['emailotp']=0

if request.method == 'POST':

# Get The Request Data

global name

global mail

global passw

global image

global phone

global otp2

name=request.POST.get('name')

phone = request.POST.get('phone')

mail = request.POST.get('mail')

passw = request.POST.get('pass')

cpass = request.POST.get('cpass')

otp=random.randint(10000, 99999)

otp2 = str(otp)

print(otp2)

clientkey=request.POST['g-recaptcha-response']

secretkey='6Ldrj0MkAAAAABH016Obv\_PagpFzfP2HOgOQ9v3v'

cptchadata={

'secret':secretkey,

'response':clientkey

}

r=requests.post('https://www.google.com/recaptcha/api/siteverify',data=cptchadata)

response=json.loads(r.text)

verify=response['success']

print(verify)

#Fatch The File Name And It's Extension

# imagename = image.name

# ex = imagename.split('.')[1]

# ex = ex.lower()

# print(ex)

# Save The Data To The DataBase

if not Customer.objects.filter(email=mail):

if passw == cpass:

if len(passw) >= 8:

if len(phone) == 10:

if verify:

subject, from\_email, to = 'Create Account', 'zoomanagmentsystem@gmail.com',mail

text\_content = 'This is an important message.'

html\_content = '<img src="https://cdn.pixabay.com/photo/2015/02/27/22/28/india-652857\_960\_720.png" alt="img"> <br> Hi '+name+' <br> Is Your One Time Password(OTP)<strong style="color:red;">'+otp2+ '</strong><br>Use For Craete The Account India\_Tour'

msg = EmailMultiAlternatives(subject, text\_content, from\_email, [to])

msg.attach\_alternative(html\_content, "text/html")

msg.send()

request.session['name']="gh"

return redirect('email')

else:

messages.error(request, 'Captcha Is Not Verify')

else:

messages.error(

request, 'Mobile Number Should Only Contain 10 Number')

else:

messages.error(

request, 'Password Must 8 Character Long !!!!')

else:

messages.error(

request, 'Conferm Password And Password Is Not Match !!!!')

else:

messages.error(request, 'Email Is Already Exist !!!!')

return render(request, 'signup.html')

def login(request):

request.session['emailotp']=0

if request.method == 'POST':

email = request.POST.get('email')

passwd = request.POST.get('passwd')

data = Customer.objects.filter(email=email).values()

clientkey=request.POST['g-recaptcha-response']

secretkey='6Ldrj0MkAAAAABH016Obv\_PagpFzfP2HOgOQ9v3v'

cptchadata={

'secret':secretkey,

'response':clientkey

}

r=requests.post('https://www.google.com/recaptcha/api/siteverify',data=cptchadata)

response=json.loads(r.text)

verify=response['success']

if verify:

if data:

if data[0].get('password') == passwd:

request.session['id'] = data[0].get('id')

request.session['uname']=data[0].get('name')

request.session['uemail']=data[0].get('email')

return redirect('/')

else:

messages.error(request, 'Password Are Incorrect')

return redirect("login")

else:

messages.error(request, 'User Does Not Exist !!!!')

return redirect("login")

else:

messages.error(request, 'Captcha Does Not Verify!!!!')

if request.session['id']:

messages.error(request,"Alredy Loged In")

return redirect('/')

else:

return render(request, 'login.html')

def logout(request):

request.session['id'] = ""

request.session['logedin'] = ""

return redirect('/')

def profile(request):

if request.method == 'POST':

updateid = request.session['id']

name = request.POST.get('name')

mobile = request.POST.get('mobile')

if len(mobile)==10:

if request.FILES.get('image')==None:

member = Customer.objects.get(id=updateid)

member.name = name

member.mobile = mobile

member.save()

else:

image=request.FILES['image']

imagename = image.name

ex = imagename.split('.')[1]

ex = ex.lower()

# print(ex)

if ex=='jpg' or ex=='jpeg' or ex=='png':

member = Customer.objects.get(id=updateid)

member.name = name

member.mobile = mobile

member.image=image

member.save()

else:

messages.error(request, 'You Should Only Upload JPG,JPEG OR PNG File')

else:

messages.error(request, 'Mobile Number Should Only Contain 10 Number')

# print(member)

# print(name,mobile)

id = request.session['id']

if not id == "":

id = request.session['id']

data = Customer.objects.filter(id=id).values()

datapac=Book.objects.filter(Useremail=request.session['uemail'])

if datapac:

context = {

'image': data[0].get('image'),

'name': data[0].get('name'),

'email': data[0].get('email'),

'mobile': data[0].get('mobile'),

'packagedata':datapac

}

return render(request, 'profile.html', context)

else:

context = {

'image': data[0].get('image'),

'name': data[0].get('name'),

'email': data[0].get('email'),

'mobile': data[0].get('mobile'),

}

return render(request, 'profile.html', context)

else:

messages.error(request, 'Whithout Login Not Goes To This Page')

return redirect('/')

def email(request):

if not request.session['name'] =="ghelo":

if request.method=="POST":

rot=request.POST.get('enter')

rotp=str(rot)

print(otp2)

print(rotp)

if rotp == otp2:

new=Customer()

new.name=name

new.mobile=phone

new.email=mail

new.password=passw

new.save()

return redirect('login')

else:

messages.error(request, 'Otp Are Incorrect')

context={

'email': mail

}

return render(request,'emailr.html',context)

else:

request.session['emailotp'] = 1

return redirect('/')

def reemail(request):

if request.method=="POST":

rotp=request.POST.get('enter')

if rotp==otpr2:

fetch=Customer.objects.get(email=emailr)

print(rpwd)

fetch.password=rpwd

fetch.save()

messages.error(request,"Your Password Reset Successfully")

return redirect('login')

else:

messages.error(request,"Incorrect OTP Plz Try Again")

context={

'email': emailr

}

return render(request,'emailr.html',context)

def forget(request):

global rpwd

global emailr

global otpr2

if request.method=="POST":

email=request.POST.get('email')

pwd=request.POST.get('password')

uemail=Customer.objects.filter(email=email).values()

clientkey=request.POST['g-recaptcha-response']

secretkey='6Ldrj0MkAAAAABH016Obv\_PagpFzfP2HOgOQ9v3v'

cptchadata={

'secret':secretkey,

'response':clientkey

}

r=requests.post('https://www.google.com/recaptcha/api/siteverify',data=cptchadata)

response=json.loads(r.text)

verify=response['success']

if not uemail:

messages.error(request,"User Does Not Exist Please Enter Valid Email Address")

else:

if not len(pwd) < 8:

if verify:

emailr=email

rpwd=pwd

otpr=random.randint(10000, 99999)

otpr2 = str(otpr)

print(rpwd,emailr,otpr2)

subject, from\_email, to = 'Forget Password', 'zoomanagmentsystem@gmail.com',emailr

text\_content = 'This is an important message.'

html\_content = '<img src="https://cdn.pixabay.com/photo/2015/02/27/22/28/india-652857\_960\_720.png" alt="Img"> <br> Hi '+uemail[0].get('name')+' <br> Is Your One Time Password(OTP) <strong style="color:red;">'+otpr2+ ' </strong> Used To Reset The Password '

msg = EmailMultiAlternatives(subject, text\_content, from\_email, [to])

msg.attach\_alternative(html\_content, "text/html")

h=msg.send()

if h:

return redirect('emailr')

else:

messages.error(request,"Some Technical Issue On The Server To Send The Massage")

else:

messages.error(request,"Captcha Is Not Verify")

else:

print("hello")

messages.error(request,"Password Should Be 8 Character Long !!!!")

return render(request,'Forgate.html')

def demail(request):

global otpd2

id = request.session['id']

if not id == "":

name= request.session['uname']

semail=request.session['uemail']

otpd=random.randint(10000, 99999)

otpd2 = str(otpd)

print(rpwd,emailr,otpr2)

subject, from\_email, to = 'Unregister Account', 'zoomanagmentsystem@gmail.com',semail

text\_content = 'This is an important message.'

html\_content = '<img src="https://cdn.pixabay.com/photo/2015/02/27/22/28/india-652857\_960\_720.png" alt="Img"> <br> Hi '+name+' <br> Is Your One Time Password(OTP) <strong style="color:red;">'+otpd2+ ' </strong> Used To Unregistered The The Account '

msg = EmailMultiAlternatives(subject, text\_content, from\_email, [to])

msg.attach\_alternative(html\_content, "text/html")

h=msg.send()

if h:

return redirect('checkdr')

else:

messages.error(request,"Some Technical Issue On The Server To Send The Massage")

else:

messages.error(request, 'Ristricted')

return redirect('/')

def checkdr(request):

if request.method=="POST":

dotp=request.POST.get('enter')

if otpd2==dotp:

new=Customer.objects.get(id=request.session['id'])

new.delete()

messages.error(request,"Account Unregistered Successfully")

return redirect('logout')

else:

messages.error(request,"OTP Is Incorrect")

id = request.session['id']

if not id == "":

context={

'email': request.session['uemail']

}

return render(request,'emailr.html',context)

else:

messages.error(request, 'Ristricted')

return redirect('/')

7

**Testing**

**(Manual,Test Case And Test dAta)**

**7.1 Manual Testing**

**7.2 Test Case**

**7.1 Manual Testing**

* Manual testing is a software testing process in which test cases are executed manually without using any automated tool. All test cases executed by the tester manually according to the end user's perspective. It ensures whether the application is working, as mentioned in the requirement document or not. Test cases are planned and implemented to complete almost 100 percent of the software application. Test case reports are also generated manually.
* Manual Testing is one of the most fundamental testing processes as it can find both visible and hidden defects of the software. The difference between expected output and output, given by the software, is defined as a defect. The developer fixed the defects and handed it to the tester for retesting.
* Manual testing is mandatory for every newly developed software before automated testing. This testing requires great efforts and time, but it gives the surety of bug-free software. Manual Testing requires knowledge of manual testing techniques but not of any automated testing tool.
* Manual testing is essential because one of the [software testing](https://www.javatpoint.com/software-testing-tutorial) fundamentals is "100% automation is not possible."
* There are various methods used for manual testing. Each technique is used according to its testing criteria. Types of manual testing are given below:

1. White Box Testing
2. Black Box Testing
3. Gray Box Testing

**White-box testing**

* The white box testing is done by Developer, where they check every line of a code before giving it to the Test Engineer. Since the code is visible for the Developer during the testing, that's why it is also known as White box testing.

**Black box testing**

* The black box testing is done by the Test Engineer, where they can check the functionality of an application or the software according to the customer /client's needs. In this, the code is not visible while performing the testing; that's why it is known as black-box testing.

**Gray Box testing**

* Gray box testing is a combination of white box and Black box testing. It can be performed by a person who knew both coding and testing. And if the single person performs white box, as well as black-box testing for the application, is known as Gray box testing.

**7.2 Test Case**

* **Login Page Test Case**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Test case Id | Test case Description | Test Step | Test Data | Expected Output | Actual Output | Pass/Fail |
| 1 | **Check Customer Login with Valid data** | **Go to Login page Enter Username Password** | **Username**  **Rahultimbaliya555**  **@gmail.com**  **Password**  **123456789** | **User Should**  **Login To The An Application** | **As Expected** | **Pass** |
| 2 | **Check Customer**  **Login with invalid data** | **Go to Login page Enter Username Password** | **Username**  [**Abc@gmail.com**](mailto:Abc@gmail.com)  **Password**  **124** | **User Should**  **Not Login To The An Application** | **As Expected** | **Pass** |
| 3 | **Check Customer**  **Login without Check**  **Captcha** | **Go to Login page Enter Username Password** | **Username**  [**Abc@gmail.com**](mailto:Abc@gmail.com)  **Password**  **124** | **User Should**  **Not Login To The An Application** | **As Expected** | **Pass** |
|  |  |  |  |  |  |  |

* **Register Page Test case**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Test case Id | Test case Description | Test Step | Test Data | Expected Output | Actual Output | Pass/Fail |
| 1 | **Check Customer Sign-Up with Valid data** | **Go To Site and enter Name**  **Mobile**  **Email**  **Password**  **Confirm**  **Password**  **And**  **Check**  **Captcha** | **Name:**  **Rahul**  **Email:**  **Rahultimbaliya**  [**555@gmail.com**](mailto:555@gmail.com)  **Mobile:**  **6353647592**  **Password:**  **123456789**  **Confirm Password:**  **123456789** | **User**  **Should**  **Sign-up**  **Into the**  **application** | **As**  **Expected** | **Pass** |
| 2 | **Check Customer Sign-Up with Valid data** | **Go To Site and enter Name**  **Mobile**  **Email**  **Password**  **Confirm**  **Password**  **And**  **Check**  **Captcha** | **Name:**  **Rahul**  **Email:**  **Abc@gmail.com**  **Mobile:**  **6353647592**  **Password:**  **123456789**  **Confirm Password:**  **123456789** | **User**  **Should**  **Not**  **Sign-up**  **Into the**  **application** | **As**  **Expected** | **Pass** |
|  |  |  |  |  |  |  |

8

**Enhancement**

**8.1 Advantage Of Your Project**

**8.2 Limitation Of Your Project**

**8.3 Feature Scope**

**8.1 Advantage Of Your Project**

* This system is very used to find online Tour Booking system .User easily interact with system and collect all the information about its tour package or if any package is related to its choice it can book that package and save its time to offline process
* System also provide some security feature like Email Authentication ,Google Captcha Authentication to user data will be secured from the hacker attack
* Best user interface to user easily interact with the website and easily to navigate the website
* Light mode And Dark Mode provided into the website to increase the user interaction with the system
* Not only booking user can generate its confirmed package booking invoice in PDF format
* User Can manage its profile
* Booking confirmation email is received on the system

**8.2 Limitation Of Your Project**

* Some web page are very heavy it means some time its load time are increase
* Some design issue for the mobile phone it not compatible with the mobile phone and tablet (Not Responsible)
* Payment system is not available at a time to user can make online payment
* Application have some runtime bug and error
* Some page are not well designed
* Not any DSA technique are used in application
* Very simple and short application

**8.3 Feature Scope**

* In feature design is better and enhance the user experience
* Fix the webpage issue to overcome the load time
* Make application to mobile or tablet friendly means responsive
* Include payment system to user easily can make payment through the application
* Fix run time error and bug
* Provide advance functionality
* Make it larger and complex logic are apply in the application
* Well design all the page on the application

9

References

* **Reference Sites:**

<https://www.tutorialspoint.com/php/>

https://www. stackoverflow.com/

https://www. getbootstrap.com/

[htttp://www.youtube.com](http://www.youtube.com)/

[htttp://www.jquery.com](http://www.youtube.com)/

<https://www.Google.com/>

<https://www.github.com/>