Software Requirements Specification

Version # 1

Tourister

<u>Team # 8</u>

Member Name	Member Roll #	Primary Responsibility
Wisha Riaz	15L-4005	Functional Requirements (UC-7,UC-8,UC-9,UC-10, UC-11,UC-12) , Appendix B
Alina Sajid	14L-4379	Non-functional Requirements (4.1,4.2)
Ifrah Nadeem	15L-4224	Functional Requirements (UC-13,UC-14,UC-15,UC-16,UC-17), Appendix B
Aqsa Noor	15L-4049	Functional Requirements (UC-1,UC-2,UC-3,UC-4, UC-5, UC-6) , Appendix A
Bilal Iqbal	14L-4143	Introduction (1.1-1.2)
Dawood Umer	14L-4161	Introduction (1.3-1.5)

Table of Contents

Revision History

Name(s)	Date	Reason(s) For Change(s)	Version

1. Introduction

1.1 Product

The purpose of this document is to build an online system to manage tours. Ir offers tour and travel services to users including transport, meals and travel equipment required by user. It also provides customized travel packages to the users. The system handles online registration, bookings and online payment. The document gives the detailed description of both functional and non-functional requirements proposed by the client.

1.2 Scope

The scope of the online tour management system is to ease tour management and to create a convenient and easy-to-use application for customers, trying to book online-tours. This application will provide its user the facility to keeps their travel plans and calendars in synchronization and up-to-date.The system will use online database for faster access and will keep the data updated for all of its user. The product follows MVC architecture technique as it provides faster development and modification does not affect the entire model.This project describes the software and hardware interface requirements using UML diagrams.

1.3 Business Goals

Above all, we hope to provide a comfortable user experience along with the best pricing available. We want to create a user-friendly environment for the user so that user can book tours and avail our package offers with ease. Our goal is to deliver the product using scrum methodology i.e deliver new features in every sprint. We will ensure timely delivery of our product.

1.4 Document Conventions

- Entire document should be justified
- Convention for main title
    - Font Face: Times New Roman
    - Font Style: Bold
    - Font Size: 18
- Convention for sub-title
    - Font Face: Times New Roman
    - Font Style: Bold
    - Font Size: 14
- Convention for body
    - Font Face: Times New Roman
    - Font Size:12

1.5 References

➢ **Books**

- Software Engineering: A Practitioner's Approach, Roger S. Pressman, 7th Edition, McGraw-Hill, 2010
- Software Requirements and Specifications: A Lexicon of Practice, Principles and Prejudices (ACM Press) by Michael Jackson
- Software Requirements (Microsoft) Second Edition By Karl E. Wiegers

➢ **Websites**

- https://www.slideshare.net/
- https://en.wikipedia.org/wiki/Software_requirements_specification

2. Functional Requirements

2.1 Use-Case 1: Sign-Up

Identifier	UC-1
Name	Sign up
Summary	This panel will include fields for user to enter his/her data for the account registration. After entering the data and tapping the 'sign up' button, the data will be validated by the system. If every input is correct then after the validation, user will be registered to the app.
Priority	High
Actors	Traveler
Pre-condition(s)	The user will tap on the app's icon.
Post-condition(s)	User account will be made. He/she will then be moved to home screen.

Typical Course of Action		
S#	**Actor Action**	**System Response**
1.	User enters first name.	
2.	User enters last name.	
3.	User enters "User-Name"	
4.		Prompts for email.
5.	User enters email.	
6.		Email not verified.
7.		Errors is displayed "wrong email entered".
8.		Prompts for password.
9.	Users enters Password.	
10.	User Re-enters Password.	
11.		Password not verified.

12.		Errors is displayed “wrong password entered”.
13.	User press Sign-Up button.	
14.		Errors is displayed “Account is not created”.
Alternate Course of Action (Wrong input case)		
S#	**Actor Action**	**System Response**
4.		Prompts for email.
5.	User enters email.	
6.		Email Verified.
7.		Prompts for password.
8.	User enters password.	
7		Password verified.
9.	User press Sign-Up button.	
10.		User account gets created.
11.		A notification will appear, with text ‘Account Created Successfully’.
12.		User is moved to home screen.

Table 1: UC-1

2.2 Use-Case 2: Log In

Identifier	UC-2
Name	Log In
Summary	It allows user to login into the account.
Priority	High
Actors	Traveler, Admin
Pre-condition(s)	User is barred from performing any function
Post-condition(s)	User logs into their account where they can perform various functions
Typical Course of Action	

S#	Actor Action	System Response
1.	Enters username	
2.		Verify username
3.		Prompts for password
4.	Enters password	
5.		Verify password.
Alternate Course of Action (Invalid username)		
S#	**Actor Action**	**System Response**
2.		Shows error message
3.		Prompts for username
4.	Enters username	
Alternate Course of Action (Invalid password)		
S#	**Actor Action**	**System Response**
5.		Shows error message
6.		Prompts for password
7.	Enters password	
8.		Password verified.
9.		Takes to HomePage.

Table 2: UC-2

2.3 Use-Case 3: Log Out

Identifier	UC-3
Name	Log Out
Summary	It allows the user to sign out from his account. After clicking on the 'log out' button the user is taken to the starting page (where he must log in again to proceed to the homepage). He no longer has any access to the content of the application.
Priority	Normal
Actors	Traveler, Admin
Pre-condition(s)	The user has access to all the data of his account

Post-condition(s)	The user is taken to the initial (log In/Sign up) page	
Typical Course of Action		
S#	**Actor Action**	**System Response**
1	User clicks on the 'log out' button	
2		A message appears 'Are you sure you want to log out?'
3	User confirms by tapping the 'Yes' option	
4		User is successfully logged out from his account
5		A message appears 'Logged out successfully'
6		The user is taken to the initial log in/sign up page
Alternate Course of Action		
S#	**Actor Action**	**System Response**
3	User taps on the 'No' option	
4		'Unsuccessful log out' message is shown
5		User stays on the current window/page.

Table 3: UC-3

2.4 Use-Case 4: Manage Packages

Identifier	UC-4
Name	Manage Packages
Summary	It allows admin to keep the track of available packages and admin can update packages.
Priority	High
Actors	Admin

Pre-condition(s)	User is barred from performing any function	
Post-condition(s)	Packages are updated.	
Typical Course of Action		
S#	**Actor Action**	**System Response**
1.	Admin views package list.	
2.		Package list is displayed.
3.		Prompts for action.
4.	Admin adds new package.	
5.		Check in the database if it already exist.
6.	Admin deletes a package.	
7.		Check in the database if it does not exist.
8.	Updates a packages.	
9.		Check in the database if it already exist.
Alternate Course of Action (Invalid username)		
S#	**Actor Action**	**System Response**
5.		Shows error message.
3.		Prompts for re-add new Package.
4.	Enters new Package and its details.	
5.		new package is Added and list is updated.
Alternate Course of Action (Invalid password)		
S#	**Actor Action**	**System Response**
7.		Shows error message
6.		Prompts to delete again by typing correct package name and id.
7.	Enters Package to delete.	
8.		Package verified.
9.		Deletes the package and updates the list.

Table 4: UC-4

2.5 Use-Case 5: View Profile

Identifier	UC-5	
Name	View Profile	
Summary	It allows user to view his profile info and upcoming tours.	
Priority	High	
Actors	Traveller	
Pre-condition(s)	User has an account on the website and has successfully logged into his account.	
Post-condition(s)		
Typical Course of Action		
S#	**Actor Action**	**System Response**
1.	User views user profile.	
2.		Displays profile of particular user.

Table 5: UC-5

2.6 Use-Case 6: View Feedback

Identifier	UC-6	
Name	View feedback	
Summary	It allows user to view ratings and comments of other users on different trips.	
Priority	High	
Actors	Traveller	
Pre-condition(s)	User has an account on the website and has successfully logged into his account.	
Post-condition(s)		
Typical Course of Action		
S#	**Actor Action**	**System Response**
1.	User views feedback.	
2.		Prompts to select trip.

4.	User selects trip.	
5.		Displays ratings and comments for that particular trip.
Alternate Course of Action (Invalid trip)		
S#	**Actor Action**	**System Response**
5.		Shows error message.
6.		Prompts for re-enter trip.
7.	Selects trip again.	
8.		Displays ratings and comments for that particular trip.

Table 6: UC-6

2.7 Use-Case 7: Give Feedback

Identifier	UC-7	
Name	Give Feedback	
Summary	It allows user to give reviews on his trips.	
Priority	High	
Actors	Traveller	
Pre-condition(s)	User has an account on the website and has successfully logged into his account. User went on that particular trip.	
Post-condition(s)	User’s comment and rating gets displayed on the website.	
Typical Course of Action		
S#	**Actor Action**	**System Response**
1.	User clicks give feedback.	
2.		Prompts user to select trip.
4.	User selects trip.	
5.		Prompts to enter rating and comment.
6.	User enters rating and comment.	
7.	User clicks submit.	
8.		Displays user’s rating and comments on website.

Alternate Course of Action (Invalid trip)		
S#	**Actor Action**	**System Response**
5.		Shows error message.
6.		Prompts for re-enter trip.
7.	Selects trip again.	
8.		Prompts to enter rating and comment.
9	User enters rating and comment.	
10		Display user's rating and comment on website.
Alternate Course of Action (Trip not attended)		
S#	**Actor Action**	**System Response**
5.		Shows error message.
6.		Prompts to enter attended trip.
7.	Selects trip again.	
8.		Prompts to enter rating and comment
9.	User enters rating and comment.	
10.		Display user's rating and comment on website.

Table 7: UC-7

2.8 Use-Case 8: Update Feedback

Identifier	UC-8
Name	Update Feedback
Summary	It allows user to update his review on a trip.
Priority	High
Actors	Traveller
Pre-condition(s)	User has an account on the website and has successfully logged into his account. User went on that particular trip. User has given review before.
Post-condition(s)	User's comment and rating are updated on the website.
Typical Course of Action	

S#	Actor Action	System Response
1.	User clicks change rating and comment	
2.		Prompts user to enter new rating and comment.
4.	User enters new rating and comment.	
5.		Updates rating and comment on website
Alternate Course of Action (Not given review before)		
S#	**Actor Action**	**System Response**
2.		Shows error message.
3.		Prompts for select some other trip.
4.	Selects trip again.	
5.	User clicks change rating and comment	
6.		Prompts user to enter new rating and comment.
7.	User enters new rating and comment.	
8.		Display user's rating and comment on website.

Table 8: UC-8

2.9 Use-Case 9: Pay Bill

Identifier	UC-9
Name	Pay Bill
Summary	It allows user to pay his tour bill.
Priority	High
Actors	Traveller
Pre-condition(s)	User has an account on the website and has successfully logged into his account. User has selected a tour.
Post-condition(s)	Payment confirmed.

Typical Course of Action		
S#	**Actor Action**	**System Response**
1.	User clicks pay bill.	

2.		DIsplays user’s pending dues.
4.	User selects one to pay its bill.	
5.		Prompts to credit/debit card details
6.	User enters credit/debit card details.	
7.	User clicks confirm payment	
8.		Confirm payment.
Alternate Course of Action (Invalid credit/debit card info)		
S#	**Actor Action**	**System Response**
8.		Shows error message.
9.		Prompts for credit/debit card details.
10.	Enters credit/debit card details again.	
11.		Confirms payment.

Table 9: UC-9

2.10 Use-Case 10: Upgrade Membership

Identifier	UC-10	
Name	Upgrade membership	
Summary	It allows user to upgrade his website membership.	
Priority	High	
Actors	Traveller	
Pre-condition(s)	User has an account on the website and has successfully logged into his account.	
Post-condition(s)	Membership fee is added to user’s pending dues.	
Typical Course of Action		
S#	**Actor Action**	**System Response**

1.	User clicks membership.	
2.		Prompts user to select type of membership.
4.	User selects membership.	
5.		Generate bill for the membership.
6.		Displays request pending until bill paid.
Alternate Course of Action (Membership already assigned)		
S#	**Actor Action**	**System Response**
5.		Shows error message.
6.		Prompts for re-select membership.
7.	Selects membership again.	
8.		Generate bill for the membership
9		Displays request pending until bill paid.

Table 10: UC-10

2.11 Use-Case 11: Take Package

Identifier	UC-11
Name	Take package
Summary	It allows user to select a tour package.
Priority	High
Actors	Traveller

Pre-condition(s)	User has an account on the website and has successfully logged into his account.
Post-condition(s)	Package fee is added to user's pending dues.

Typical Course of Action

S#	Actor Action	System Response
1.	User selects book tour.	
2.		Prompts user to select a package from available packages.
3.	User selects package.	
4.		Generate bill for tour package.
5.		Displays request pending until bill paid.

Table 11: UC-11

2.12 Use-Case 12: Take Facilities

Identifier	UC-12
Name	Take facilities
Summary	It allows user to take several facilities with his selected package.
Priority	High
Actors	Traveller

Pre-condition(s)	User has an account on the website and has successfully logged into his account. User has selected a tour package.
Post-condition(s)	Facility fee is added to user's pending dues for the tour.

Typical Course of Action

S#	Actor Action	System Response
1.	User clicks add facility to tour.	
2.		Prompts user to select a facility.
4.	User selects facility.	
5.		Generate bill for that facility.
6.		Displays request pending until bill paid.

Alternate Course of Action (Facility not available for that package)

S#	Actor Action	System Response
5.		Shows error message.
6.		Prompts for choose a different facility.
7.	Selects a different facility.	
8.		Generate bill for the facility.
9		Displays request pending until bill paid.

Alternate Course of Action (Facility already added in the package)

S#	Actor Action	System Response

5.		Shows error message.
6.		Prompts for choose a different facility.
7.	Selects a different facility.	
8.		Generate bill for the facility.
9.		Displays request pending until bill paid.

Table 12: UC-12

2.13 Use-Case 13: Update Profile

Identifier	UC-13
Name	Update profile
Summary	It allows user to update his profile.
Priority	High
Actors	Traveler, Admin
Pre-condition(s)	User has an account on the website and has successfully logged into his account.
Post-condition(s)	Changes are saved.

Typical Course of Action

S #	**Actor Action**	**System Response**
1.	User clicks change password.	
2.		Prompts user to enter old and new password.

4.	User enters old and new password.	
5.		Update password of user.
6.	User clicks change phone number.	
7.		Prompts user to enter new phone number.
8.	User enters new password.	
9.		Update phone number of user.
Alternate Course of Action (Incorrect old password)		
S #	**Actor Action**	**System Response**
5.		Shows error message.
6.		Prompts to enter correct old password.
7.	Enters password again.	
8.		Update password of user.
Alternate Course of Action (Invalid phone number format)		
S #	**Actor Action**	**System Response**
9.		Shows error message.
10.		Prompts to enter correct phone number.
11.	Enters password again.	
12.		Update password of user.

Table 13: UC-13

2.14 Use-Case 14: Check History

Identifier	UC-14
Name	Check history
Summary	It allows user to check his previous tour history.
Priority	High
Actors	Traveler
Pre-condition(s)	User has an account on the website and has successfully logged into his account.
Post-condition(s)	

Typical Course of Action

S #	**Actor Action**	**System Response**
1.	User views history.	
2.		Displays history of tours taken by user previously

Table 14: UC-14

2.15 Use-Case 15: Manage Tours

Identifier	UC-15
Name	Manage Tours

Summary	It allows admin to keep the track of available Tours and admin can update Tours.
Priority	High
Actors	Admin
Pre-condition(s)	User is barred from performing any function
Post-condition(s)	Tours are updated.

Typical Course of Action

S#	Actor Action	System Response
1.	Admin views tours list.	
2.		Tours list is displayed.
3.		Prompts for action.
4.	Admin adds new tour.	
5.		Check in the database if it already exist.
6.	Admin deletes a tour.	
7.		Check in the database if it does not exist.
8.	Updates a tour.	
9.		Check in the database if it already exist.

Alternate Course of Action (Insert)

S#	Actor Action	System Response
5.		Shows error message.

3.		Prompts for re-add new tour.
4.	Enters new tour and its details.	
5.		New tour is Added and list is updated.
Alternate Course of Action (Delete)		
S#	**Actor Action**	**System Response**
7.		Shows error message
6.		Prompts to delete again by typing correct tour name and id.
7.	Enters tour to delete.	
8.		Tour verified.
9.		Deletes the tour and updates the list.

Alternate Course of Action (Update)		
S #	**Actor Action**	**System Response**
7.		Shows error message
6.		Prompts to update again by typing tour that already exist in Database.
7.	Enters tour to update.	
8.		tour verified.

9.		Updates the tour details in the database.

Table15: UC-15

2.16 Use-Case 16: Manage Facilities

Identifier	UC-16	
Name	Manage facilities	
Summary	It allows admin to keep the track of available facilities and admin can update facilities.	
Priority	High	
Actors	Admin	
Pre-condition(s)	User is barred from performing any function	
Post-condition(s)	Facilities are updated.	
Typical Course of Action		
S#	**Actor Action**	**System Response**
1.	Admin views facilities list.	
2.		facilities list is displayed.
3.		Prompts for action.
4.	Admin adds new facilities.	
5.		Check in the database if it already exist.
6.	Admin deletes a facilities.	

7.		Check in the database if it does not exist.
8.	Updates a facility.	
9.		Check in the database if it already exist.

Alternate Course of Action (Insert)		
S#	**Actor Action**	**System Response**
5.		Shows error message.
3.		Prompts for re-add new facility.
4.	Enters new facility and its details.	
5.		New facility is Added and list is updated.

Alternate Course of Action (Delete)		
S#	**Actor Action**	**System Response**
7.		Shows error message
6.		Prompts to delete again by typing correct facility name and id.
7.	Enters facility to delete.	
8.		facility verified.
9.		Deletes the facility and updates the list.

Alternate Course of Action (Update)		
S#	**Actor Action**	**System Response**

7.		Shows error message
6.		Prompts to update again by typing facility that already exist in Database.
7.	Enters facility to update.	
8.		facility verified.
9.		Updates the facility details in the database.

Table 16- UC16

2.17 Use-Case 17: Book Trips

Identifier	UC-17	
Name	Book trips	
Summary	It allows user to book a tour.	
Priority	High	
Actors	Traveler	
Pre-condition(s)	User has an account on the website and has successfully logged into his account.	
Post-condition(s)	Tour bill is added to user's pending dues.	
Typical Course of Action		
S#	**Actor Action**	**System Response**
1.	User clicks book tour	

2.		Prompts user to select package.
3.	User selects a package from available packages	
4.		Prompts user to enter destination and date
5.	User enters destination and date	
6.		Generate bill for that tour.
7.		Displays request pending until bill paid.
Alternate Course of Action(No Tours to Destination)		
S#	**Actor Action**	**System Response**
6.		Shows error message.
7.		Prompts to select different destination
8.	User enters destination	
9.		Generate bill for that tour.
10.		Displays request pending until bill paid.

Table 17- UC17

2.18 Use-Case 18: Manage Bills

Identifier	UC-18
Name	Manage Bills

Summary	It allows bank system to keep the record of bills paid by traveller.
Priority	High
Actors	Bank System
Pre-condition(s)	User requests to retrieve bill.
Post-condition(s)	Bill status changes and update database.

Typical Course of Action		
S#	**Actor Action**	**System Response**
1.	Request to get bill details	
2.		Accepts requests
3.		Prompts for Bill ID
4.	Enter Bill ID	
5.		Checks Bill ID from database
6.		If Bill ID validates, return bill details to admin.
Alternative Course of Action(Bill ID not validates)		
6.		If Bill ID not validates, return Error Message "Bill ID not valid!".

3. Nonfunctional Requirements

3.1 Usability Requirements

The interface of our system is easy to use and learn and gives proper error messages in case of wrong inputs. Error messages will explain how to recover from errors. It will display confirmation dialog-box while making bookings and transactions. It will show proper pop-up notifications. The UI will be user-friendly and will have proper buttons which will be easy to interpret. It will satisfy the user needs and will maintain performance of the application.

3.2 Reliability Requirements

Total cost of the tour would be calculated correctly on the basis of user membership, discount, package and added facilities price. It will ensure no loss of data. Furthermore, it will ensure data integrity.

3.3 Security Requirements

Payment services via PayPal or credit/debit card must be integrated securely using good encryption methods to store and manipulate information. The system should keep customer information, online bookings and transactions secure. The system will ensure the privacy of customer's personal information.

3.4 External Requirements

The system will not disclose any personal information about customers to any other company as it will contain sensitive data like payment methods, credit cards details etc. The system will provide facilities to customers 24/7.

3.5 Operational Requirements

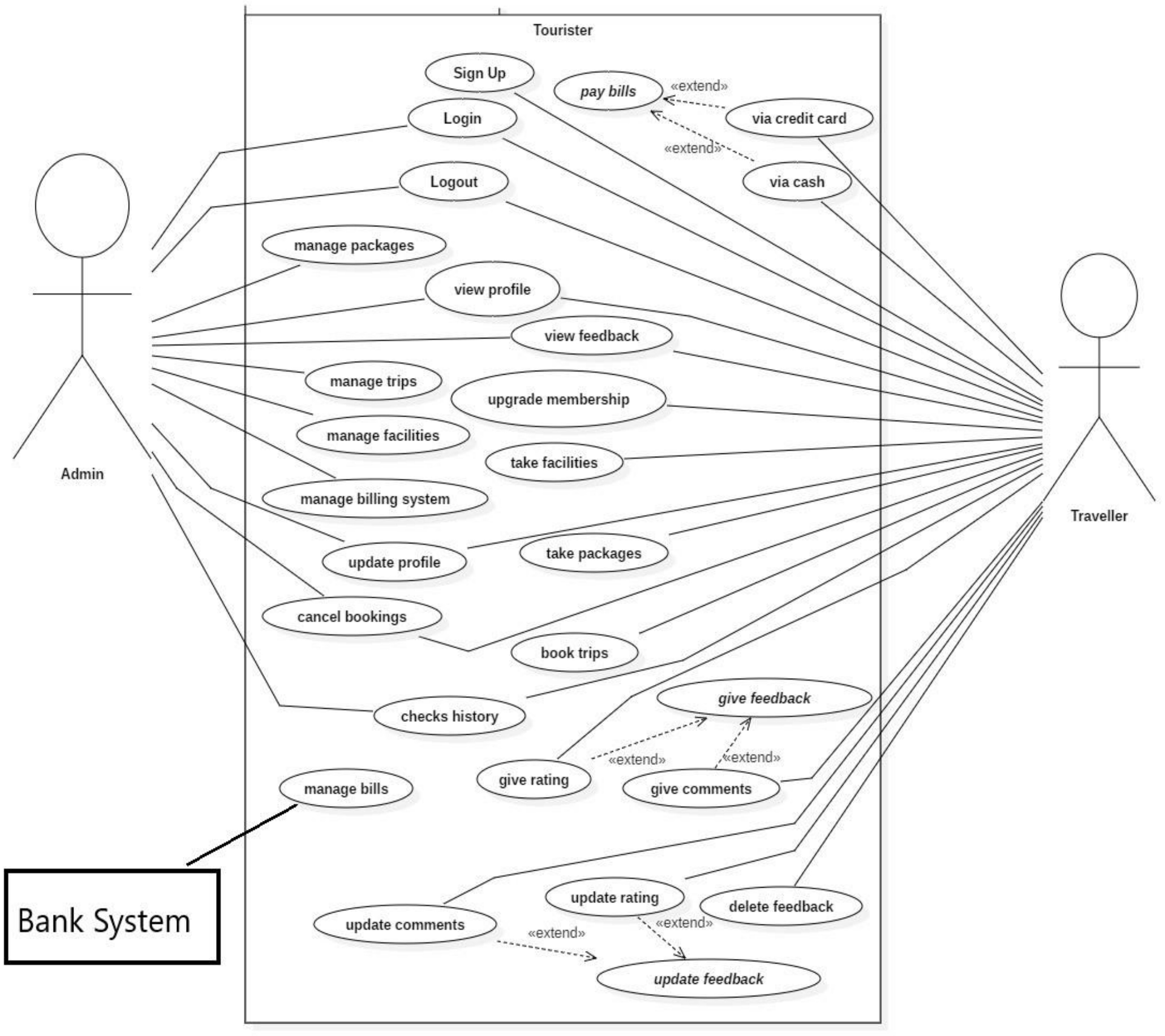
Website should be accessible from any platform having internet connectivity and browsing feature to make it available to users all the time.

Appendix A: Glossary

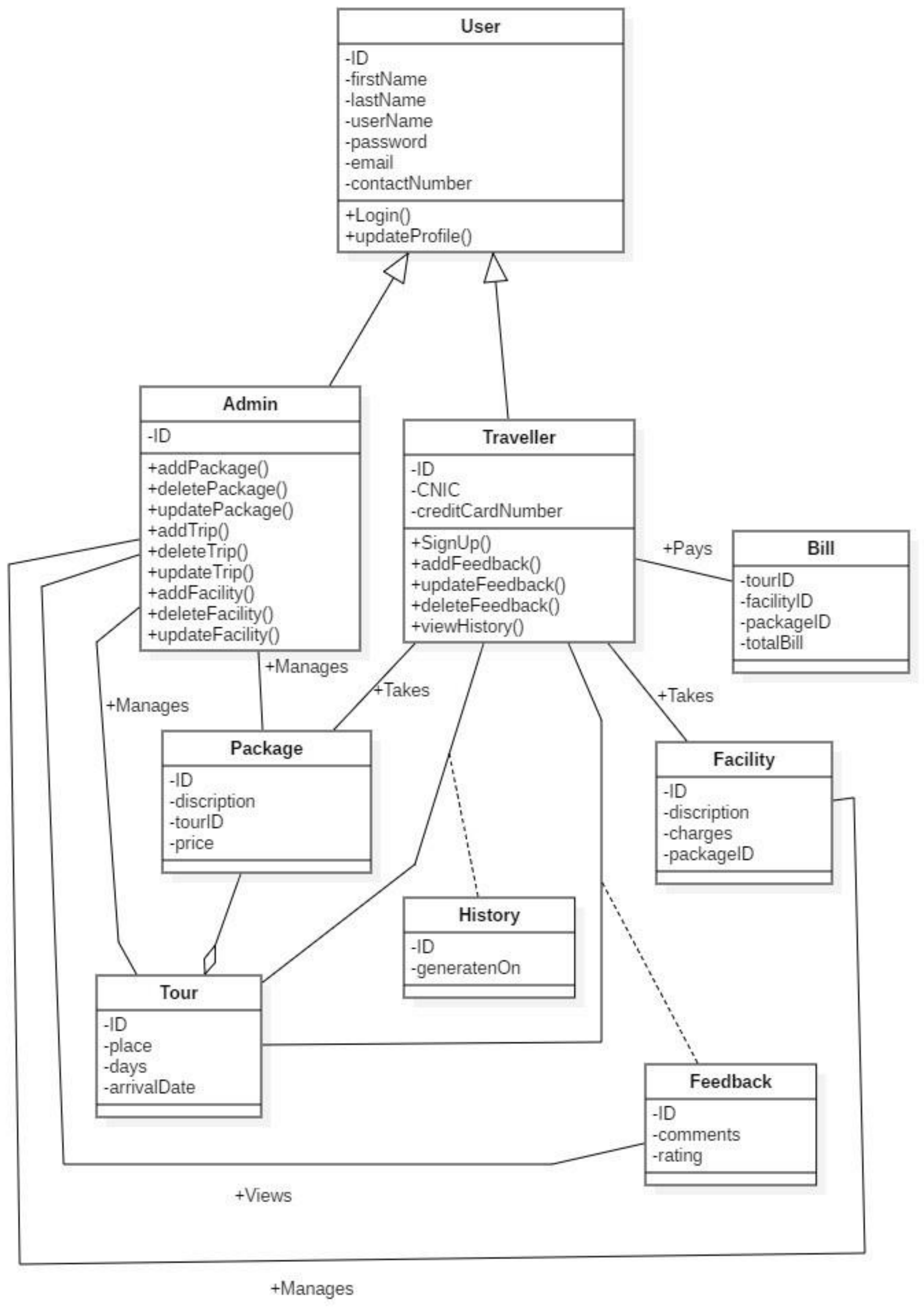
- **DFD** : Data Flow Diagram
- **UC** : Use-Case
- **CD**: Class Diagram
- **Data Reliability**: the extent to which data yields the same results on repeated trials.
- **Data Integrity**: It is the maintenance and the assurance of the accuracy and consistency of data.
- **MVC** : Model View Controller
- **Synchronization**: to cause to agree in time of occurrence; assign to the same time or period.
- **Sprint** : one phase

Appendix B: Analysis Models

Use Case Diagram

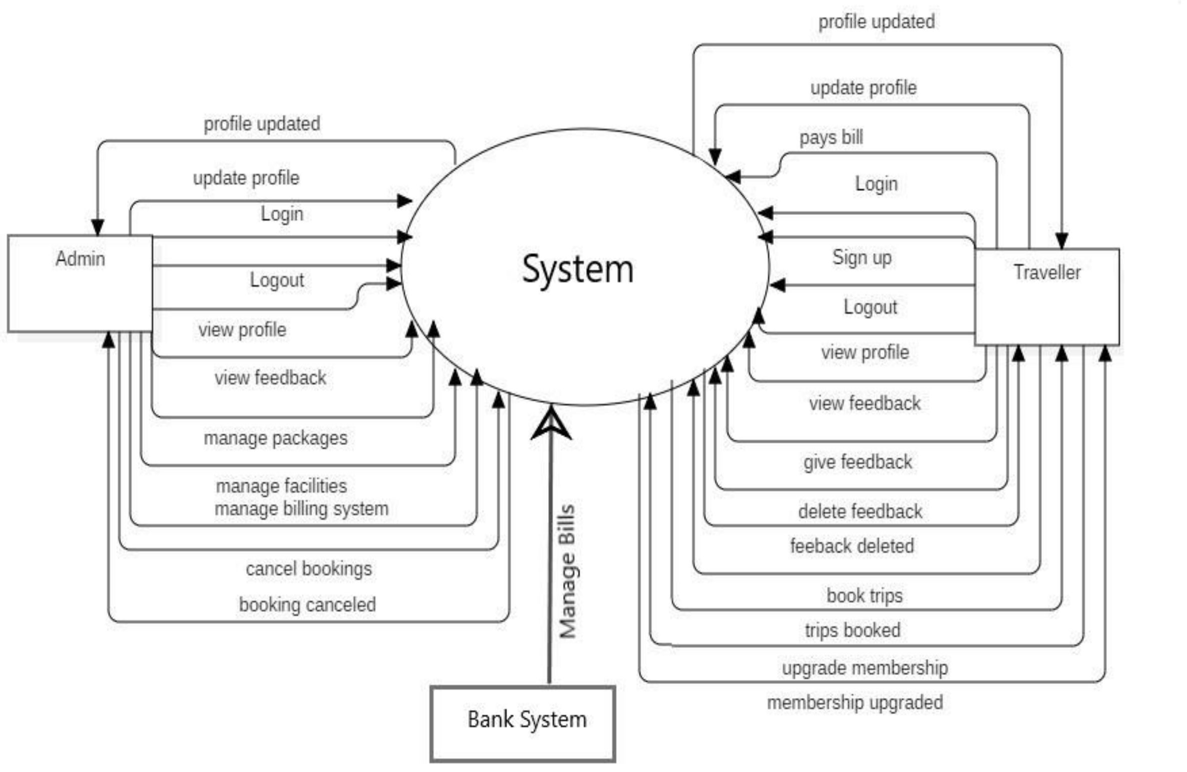


Class Diagram-Analysis Level

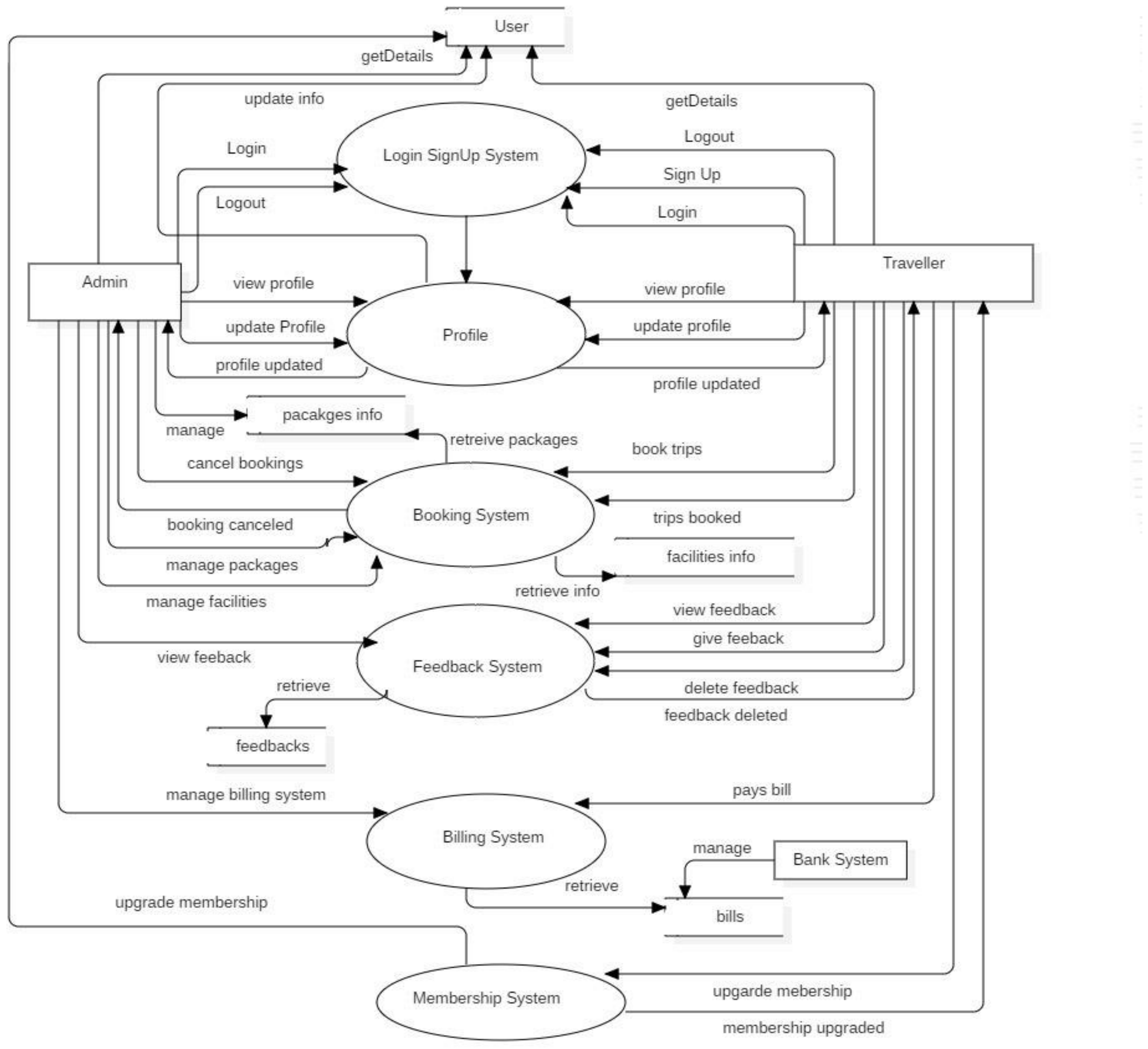


Data Flow Diagram

Level 0



Level 1


User
getDetails
update info
getDetails
Login SignUp System
Logout
Login
Sign Up
Logout
Login
Admin
Traveller
view profile
view profile
Profile
update Profile
update profile
profile updated
profile updated
pacakges info
manage
retreive packages
book trips
cancel bookings
Booking System
trips booked
booking canceled
facilities info
manage packages
retrieve info
manage facilities
view feedback
give feeback
view feeback
Feedback System
retrieve
delete feedback
feedback deleted
feedbacks
manage billing system
pays bill
Billing System
manage
Bank System
retrieve
upgrade membership
bills
Membership System
upgarde mebership
membership upgraded