

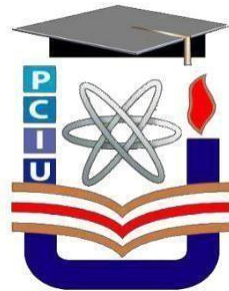
Ticket Booking Management System

by

Md Riyajul Jannat

Id: CSE 01205995

**This Project Report Presented in Partial Fulfillment of the Requirements for the
Degree of Bachelor of Science in Computer Science & Engineering**



**Department of Computer Science & Engineering
Port City International University
Chattogram, Bangladesh**

March, 2021

Ticket Booking Management System

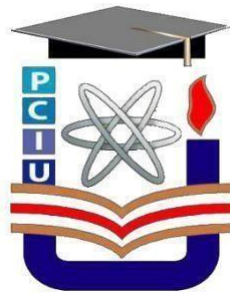
**This Project Report Presented in Partial Fulfillment of the Requirements for the
Degree of Bachelor of Science in Computer Science & Engineering**

Supervised by

Mohammed Armanuzzaman Chowdhury

Lecturer, Department of CSE

Port City International University



Department of Computer Science & Engineering

Port City International University

Chattogram, Bangladesh

March, 2021

Approval for Submission

This project “**Ticket Booking Management System**” by Md Riyajul Jannat, Student ID: CSE 01205995, Batch: CSE 012 has been approved for submission to the Department of Computer Science & Engineering, Port City International University in partial fulfillment of the requirements for the degree of Bachelor of Science (Engineering).

.....

Signature of the Supervisor

Mohammed Armanuzzaman Chowdhury

Lecturer,

Department of Computer Science & Engineering,

Port City International University

S Khulshi Rd, Chattogram 4202

Chattogram, Bangladesh.

Email: Arman.cu.cse@gmail.com

Telephone: 01925-568358

DECLARATION

I hereby declare that the whole project work has been done by me and not any portion of the work contained in support of any other application for any other qualification or degree of this or any other university or institution.

.....

Signature of the candidate

Md Riyajul Jannat

ID: CSE 01205995

DEDICATION

This project is dedicated to

My beloved parents

&

Honorable teachers

ACKNOWLEDGEMENT

At first, I would like to praise and thanks to almighty Allah, the most merciful and most beneficent for giving patience, courage, and great opportunity to develop the project within the meantime.

I am grateful to my project supervisor **Mohammed Armanuzzaman Chowdhury** for providing me his valuable time, motivation, guidance & support to complete this project perfectly.

I'm also grateful to our faculty members for their support and inspiration all through long.

ABSTRACT

Ticket is certification or token which we need when we travel from one place to another place through any vehicle, Launch, Airplane or for any event. This system allows us to book ticket online for bus, launch, air, events, movie. We can check upcoming events also with this system. It provides all the facility to user to book and manage ticket online easily. User can book their ticket anytime online with this system. User can compare ticket price because a lot of agencies are available here. User can search for events with location. User can search for movies with location and date they don't have to input movie name. They can see all movie list of that location on that date. In summary, this system will fulfil all user requirements to book ticket online compare to other system.

Table of Contents

List of Tables.....	viii
List of Figures	ix

Chapter 1	Problem Definition	1 - 4
	1. Problem Statement	1
	2. Project Objectives	1
	3. Preliminary Solution	2
	4. Project Scope	2
	5. Estimated Cost and Time for Feasibility Study	3
	6. Organizations of the Report	3
Chapter 2	Feasibility Study	5-10
	1. Background	5
	2. System Outline	5
	3. Methodology	6
	4. Overview of Alternatives	7
	5. Recommendation	9
	6. Conclusion	10
Chapter 3	Project Planning	11-17
	1. Project Organization	11
	2. Risk Analysis	11
	3. Hardware and Software Resource Requirement	15
	4. Work Breakdown	15
	5. Project Schedule	16
	6. Monitoring and Reporting Mechanism	17
	7. Conclusion	17
Chapter 4	Software Requirements Specification	18-24
	1. Preface	18
	2. Introduction	18
	3. Glossary	19
	4. User Requirements Definition	19
	5. System Architecture	20
	6. User Requirements Specification	20
	7. System Evolution	21
	8. Appendices	21
	9. Index	23

Chapter 5	System Designing	25-28
	1. Architectural Pattern	25
	2. System Diagram	26
	3. ER Diagram	27
	4. Conclusion	28
Chapter 6	Implementation	29-36
	1. System Demonstration	29
Chapter 7	Conclusion	37-37
	1. System Limitation	37
	2. Future work	37
	3. Conclusion	37
	Appendices	38-49

List of Tables

Table Name	Description of Tables
------------	-----------------------

Chapter 1	Problem Definition
------------------	---------------------------

Table 1.1	Project Scope
-----------	---------------

Chapter 2	Feasibility Study
------------------	--------------------------

Table 2.1	System Outline
-----------	----------------

Table 2.2	Overview of Alternatives
-----------	--------------------------

Table 2.3	Summary of cost for alternative 1
-----------	-----------------------------------

Table 2.4	Summary of cost for alternative 2
-----------	-----------------------------------

Table 2.5	Benefits and Cost for alternative 1
-----------	-------------------------------------

Table 2.6	Investment analysis for alternative 1
-----------	---------------------------------------

Table 2.7	Benefits and Cost for alternative 2
-----------	-------------------------------------

Table 2.8	Investment analysis for alternative 2
-----------	---------------------------------------

Table 2.9	Comparison between alternatives
-----------	---------------------------------

Chapter 3	Project Planning
------------------	-------------------------

Table 3.3	Role of Team Member
-----------	---------------------

Table 3.2	List of possible risks
-----------	------------------------

Table 3.3	Possibility and Effect of the risks
-----------	-------------------------------------

Table 3.4	Risk Reduction Strategies
-----------	---------------------------

Table 3.5	Hardware and Software Resource Requirement
-----------	--

Table 3.6	Task Summary
-----------	--------------

Table 3.7	Milestone and Deliverables
-----------	----------------------------

Table 3.8	Schedule Representation
-----------	-------------------------

Table 3.9	Task Distribution
-----------	-------------------

Chapter 4	Software Requirements Specification
------------------	--

Table 4.1	Glossary
-----------	----------

Table 4.2	Definition of User Requirements
-----------	---------------------------------

Table 4.3	Specification of User Requirements
-----------	------------------------------------

Table 4.4	List of Use Cases
-----------	-------------------

Table 4.5	Server Specification
-----------	----------------------

Table 4.6	Database Specification
-----------	------------------------

Table 4.7	List of Figures
-----------	-----------------

List of Figures

Figure Name	Description of Figure
Chapter 4	Software Requirements Specification
Figure 4.1	System Architecture
Figure 4.2	Use Case Diagram
Chapter 5	System Designing
Chapter 5.1	Architectural Pattern of Ticket Booking

Chapter 1

Problem Definition

The “**Ticket Booking Management System**” is an internet-based software which helps to book ticket online and passenger can see available ticket and can choose bus ticket from the system.

It will be designed to manage a Ticket Booking all functions like choose bus seat by check available seat, book launch ticket, air ticket, events ticket, movie ticket by check available ticket, print booked ticket from system. Agency could be adding bus info, launch info, air info, event info, movie info and can manage added bus, launch, air, event, movie list, agency can see booked ticket list and can manage those ticket if they have to cancel it. The system will ensure manage the Ticket Booking very easily. At first, passenger can see the ticket information without log in to system if they need to book ticket they have to sign up to the system, passenger can see available seat before booking. Passenger can see bus seat availability before booking and can choose seat for book. Passenger have to bKash their ticket advance to confirm ticket. Agency can manage vehicle or ticket information and can see booked list individually. Agency can cancel ticket booking if they have to but they have to back ticket price directly to passenger within 72 hours.

The project is summarized through six sections which are briefly described as following:

Section 1: describes the problems with the current Ticket Booking management system.

Section 2: The purpose of this project is described briefly.

Section 3: Provides and discusses preliminary solutions to this problem.

Section 4: Scope and limitations of the project are described.

Section 5: Present the estimate cost and required time for the feasibility studies.

Section 6: Present the organizations of the report.

1.1 Problem Statement

Many agencies have its own ticket booking software but they can't meet the expectation of passengers. The present system has the following problems:

- Passengers have no option if they don't get seat.
- Passengers can't check other companies ticket price or facility.
- Passengers can't book all type of system in one system.
- Passengers have to give full price of ticket as advance & sometimes they cheated by agencies.
- Passengers not friendly with agencies software environment because there is no instruction for passengers how to use it.

1.2 Project Objectives

The project objectives are described below:

- To help passengers to check all agencies ticket and compare price.
- To ensure availability of seat and choose bus seat from system.
- To book all type of tickets like bus, launch, air, event, movie in one place.
- To ensure ticket advance system easily and passenger friendly.
- To ensure the system is trusted and make the system environment passenger friendly so that passenger can use the system easily.

1.3 Preliminary Solution

The problem can be solved by replacing the current Ticket Booking management system with an Online Ticket Booking management system where passenger can book all type of ticket online. Passenger can check upcoming events ticket in this system, Passengers can compare price with other agencies and can get ticket in reasonable price. Agencies can add & manage their vehicle and ticket information and can also check booked ticket list & can cancel any ticket if they have to and many more.

1.4 Project Scope

The project aims to provide an effective solution to the problem. The scope of this project is described in Table 1.1:

Table 1.1: Project Scope

Serial No.	Item	Details
1.	Functions	1. Network Credential 2. Session Control
2.	Features	1. View upcoming events 2. Book bus ticket 3. Book launch ticket 4. Book air ticket 5. Book events ticket 6. Book movie ticket 7. Print booked tickets 8. View how to buy tickets 9. Add & manage bus info 10. Add & manage launch info 11. Add & manage air info 12. Add & manage event info 13. Add & manage movie info 14. View & manage booked tickets 15. Manage user info 16. Manage admin info

3.	Facilities	<ol style="list-style-type: none"> 1. Passengers can see upcoming events online 2. Passengers can book bus ticket by comparing price with other company and choose seat 3. Passengers can book launch ticket by check available seat and compare price 4. Passenger can book air ticket by check availability of seat and compare price 5. Passengers can book movie ticket by search with movie name or district and seat availability 6. Passengers can book events ticket by checking availability 7. Agencies can manage their bus, launch, air, events, movie ticket. 8. Agencies can manage their booked ticket list and can cancel ticket if they have to.
4.	Required Time	1. Around 6 months is needed to complete the project.
5.	Estimated Cost	1. About 6 lack taka will be needed to complete the project.

1.5 Estimated Cost and Time for Feasibility Study

Determination and organization of the necessary details to make the project work best for a system needs a feasibility study. It will help to propose the best solution to the problem according to the resource available. It will also determine if the project is technically, legally, and financially possible with the resource available for the system.

So, we would like to propose to have **1-week** time and a cost of 10000 **BDT** to do the feasibility study.

1.6 Organization of the Report

Chapter 1 Problem Definition:

The chapter is summarized through six sections which are briefly described as following: Section 1 describes the problems with the current online shopping system. The purpose of this project is described briefly in Section 2. Section 3 provides and discusses preliminary solutions to this problem. The scope and limitations of this project are described in Section 4. Section 5 presents the estimated cost and the required time for the feasibility studies. Section 6 presents the organizations of the report.

Chapter 2 Feasibility Study:

The chapter is summarized in six sections. Section 1 describes the background of the study. The outline of the **Ticket Booking management system** is discussed in section 2. Section 3 describes the

methodology used for the study. The complete overview of the alternatives is described in section 4. Section 5 gives a conclusion to the study. The recommendation based on the study is given in Section 6.

Chapter 3 Project Planning:

The chapter covers eight sections. Section 2 discusses how the development team is organized and the team member's role in the team. Section 3 illustrates possible project risks, the likelihood of these risks, strategies for risk reduction. Hardware and software resource requirements are discussed in section 4. In section 5, the breakdown of the project into activities, milestones, and deliverables are discussed. Section 6 is on a project schedule that shows dependencies between activities, estimated time to reach each milestone, and allocation of people to activities. Section 7 discusses monitoring and reporting mechanisms. The concluding remarks are given in Section 8.

Chapter 4 Software Requirement Specification:

The chapter contains ten sections. Section 1 gives a preface to this chapter while section 2 introduces the readers with the system as well as with this chapter. Section 3 familiarize the readers with the technical terms used in this chapter. Section 4 narrates the user requirements briefly while section 5 gives a brief description of the system architecture. The specific description of the user requirement is described in section 6. Use case scenarios of the system are given in section 7 and section 8 bears the anticipated change or evolution of the system due to hardware changes. Section 9 and section 10 contains the appendices and index for this chapter consecutively.

Chapter 5 System Designing:

The chapter is composed of five sections. Section 1 describes the architectural pattern of the system, class diagram of the system is shown in section 2, Section 3 shows the code map of the system, Section 4 shows the ER-Diagram of the system and the conclusion for this chapter is given in section 5.

Chapter 6 Implementation:

The chapter consists of three sections. Section 1 shows the detailed class diagram for the system whereas section 2 includes the code map. The brief graphical demonstration of the system is shown in section 3.

Chapter 7 Conclusion:

The chapter consists of three sections. Section 1 includes the limitations of the system and the future works for the next versions are described in Section 2. The concluding remarks for the report are given in section 3.

Chapter 2

Feasibility Study

The “**Ticket Booking Management System**” is an internet-based software which helps to manage a Ticket Booking curriculum and passengers can book ticket online by comparing other companies ticket price.

It will be designed to manage a Ticket Booking all functions like manage see upcoming events in one place, book bus ticket by choose seat, book launch ticket by check available ticket, book air ticket by comparing price and schedule of air departure, book events ticket by check event schedule based on district, book movie ticket by movie name or check movie list by district and premier date, agencies can manage their bus, launch, air, events, movie ticket easily and can see booked ticket list and manage those ticket based on purpose. The system will also be designed to make the ticket booking system easily and user friendly.

The chapter is summarized in six sections. Section 1 describes the background of the study. The outline of the **Ticket Booking management system** is discussed in section 2. Section 3 describes the methodology used for the study. The complete overview of the alternatives is described in section 4. Section 5 gives a conclusion to the study. The recommendation based on the study is given in Section 6.

2.1 Background

Present Ticket Bookings have their management software. But has some limitations on their system. Passengers can't check other companies ticket price, passenger have limitation to book ticket, user interface isn't user friendly, passenger have only limited option they can't compare ticket price with others, agencies can't sell all type of tickets in one place.

Bangladesh is now growing day by day and peoples are now online dependent. So that they want to do their maximum work in online.

The proposed system Ticket Booking management system will be better to help to book tickets online.

2.2 System Outline

The **Ticket Booking management system** is a system to make the ticket booking process robust, reliable, efficient, and cost-effective. The complete outline of the project is described in Table 2.1.

Table 2.1: System Outline.

Serial	Item	Description
1	Users	1. Admin 2. Passenger 3. Agency
2	Existing problems	1. Passenger can't book events ticket online 2. Passengers can't check other company's price 3. Passengers can't book all type of tickets in one system 4. Passengers can't check departure time of all company that's why he/she is unable to get his desired schedule

		6. Agencies can't sell all type of ticket in one place 7. User interface is not user friendly 8. There is no online movie or event booking system
3	Reason of problems	1. Not having user friendly system 2. Not having an online platform for selling all type of ticket 3. Not having system where all agencies is available with their service
4.	Performed tasks	1. Register online. 2. Book ticket for desired place by comparing price. 3. Print booked ticket online. 4. Network credential. 5. Session control.
5	Required data	1. Agency information. 2. Bus, launch, air, event, movie ticket information. 3. Passenger information. 4. Admin information.

2.3 Methodology

Feasibility study will help to determine whether to proceed with the project or not. It will also determine if the problem is worth solving by considering the economical, technical and operational aspects. It also helps to determine which alternatives should be taken to produce the most benefit using the least expense. It will also make recommendations among proposed alternatives. There are three methods of analysis for feasibility study namely - Economically feasible study, Technically feasible study, and operationally feasible study.

By studying technical feasibility, we were able to know that if the required technology to implement the alternatives is available or not. We were able to determine if the required human resource is available to operate the system, by studying operational feasibility. The economical feasible study we were able to predict the profit based on the present investment. The technical feasibility study and operational feasibility study were conducted in the following manner:

- i. We sent a team of expert to observe the current system site and talked with the teachers.
- ii. We interviewed the chairman of the department.
- iii. We observed the workstations and servers available.
- iv. We gathered knowledge about the culture, union agreement, and rules of the department.
- v. We also talked with the users and held a survey with the help of questionnaires.

For the economic feasibility study, we do a financial analysis to compute the future benefit for the investment of the project by using the following formula:

$$P = F / (1 + I)^n \quad (1)$$

Here P, F, n, I am present value, future benefit, year of benefit & expected rate of return, respectively.

Based on clients' requirements, possible alternatives are proposed and then three types of feasibility study are analyzed in each alternative way.

2.4 Overview of Alternatives

A problem can be solved in many ways. Similarly, the problem of the existing **Ticket Booking Management System** can be solved variously. By studying the current system, we propose two alternatives to the current system to make the current Ticket Booking Management System more reliable, efficient, robust, and cost-effective. They are- (i) Ticket Booking Management System, (ii) Ticket Booking Management Web & Mobile Application.

A brief description of alternative systems is shown in Table 2.2:

Table 2.2: Overview of alternatives

Serial no.	Alternative 1	Alternative 2
1	It needs a visual studio to develop the website	It needs visual studio and android studio to develop the app
2	Data is accessed through an android set or pc with a web browser	Data is accessible through the android set
3	Slightly cost to maintain	Very costly to maintain

2.4.1 Economic Feasibility study for the alternatives

Initial investment needed for alternative 1 and alternative 2 is shown in table 2.3 and 2.4 respectively:

Table 2.3: Summary of cost for alternative 1

Serial No.	Item	Amount (BDT)
1	Application Software Development	3,00,000
2	Workstation PC	60,000
3	Web Server	20,000
4	Database Server	15,000
5	Initial Data Entry	15,000
Total		4,60,000

Table 2.4: Summary of cost for alternative 2

Serial No.	Item	Amount (BDT)
1	Application Software Development	7,00,000
2	Workstation PC	60,000
3	Web Server	20,000
4	Database Server	15,000
5	Initial Data Entry	15,000
Total		8,10,000

Financial Analysis of alternative 1:

The investment of Alternative 1 is BDT 4,60,000 BDT which is a one-time initial cost. Benefits and costs yearly are described in Table 2.5:

Table 2.5: Benefits & Cost for alternative 1

Benefits			Cost		
Serial No.	Particulars	Amount (BDT)	Serial No.	Particular	Amount (BDT)
1	Better Service	6,20,000	1	Maintenance & Stationary	2,25,000
			2	Data Entry	5000
Net return per year (6,20,000-(2,25,000+5000))					3,45,000

Based on table 2.5 investment analysis for alternative 1 is shown table 2.6

Table 2.6: Investment analysis for alternative 1

Year	Savings (BDT in lakh)	Present Value (10%)	Cumulative Value
1	3.45	3.1	3
2	3.45	2.8	5.8
3	3.45	2.5	8.32
4	3.45	2.25	10

Financial Analysis of Alternative 02:

The investment of Alternative 2 is BDT 9,50,000 BDT which is a one-time initial cost. Benefits and costs yearly are described in Table 2.7:

Table 2.7: Benefits and Cost for Alternative 2

Benefits		Cost			
Sl. No.	Particulars	Amount (BDT)	No. Sl.	Particular	Amount (BDT)
1	Better Service	5,00,000	1	Maintenance and Stationary	2,75,000
2	Data Entry	5000			
Net return per year (500000 – 275000 – 5000)					2,20,000

Based on table 2.7 investment analysis for Alternative 2 is shown in table 2.8.

Table 2.8: Investment analysis for Alternative 2

Year	Saving (Lakhs BDT)	Present Value (at 12%)	Cumulative Value
1	2.2	2	2
2	2.2	1.82	3.82
3	2.2	1.65	4.85
4	2.2	1.50	6.32
5	2.2	1.36	7.69
6	2.2	1.24	8.93
7	2.2	1.12	10

Benefits		Cost			
Sl. No.	Particulars	Amount (BDT)	No. Sl.	Particular	Amount (BDT)
1	Better Service	5,00,000	1	Maintenance and Stationary	2.75,000
2	Data Entry	5000			
Net return per year (500000 – 275000 – 5000)					2,20,000

Based on table 2.7 investment analysis for Alternative 2 is shown in table 2.8.

Table 2.8: Investment analysis for Alternative 2

Year	Saving (Lakhs BDT)	Present Value (at 12%)	Cumulative Value
1	2.2	2	2
2	2.2	1.82	3.82
3	2.2	1.65	4.85
4	2.2	1.50	6.32
5	2.2	1.36	7.69
6	2.2	1.24	8.93
7	2.2	1.12	10

2.4.2 Technical Feasibility study for the alternatives:

Our alternative systems need only a computer which is already available to everyone. So, all of the alternative systems are technically feasible.

2.4.3 Operational Feasibility study for the alternatives:

All of the alternatives are easy to use and user friendly. Also, it is a robust system that protects the security of the data. So, all of the alternatives are operationally feasible.

2.5 Recommendation

After analyzing the alternatives in different sectors like economical, operational, and technical according to money and time constraints, we recommend the best one of them based on different features. The comparison between alternatives to develop **Ticket Booking Management System** is illustrated in table 2.7:

Table 2.9: Comparison between alternative

Serial No.	Feature	Alternative 1	Alternative 2
1	Investment	4,60,000	8,10,000
2	System Life Cycle	4 year	6 year
3	Return Value	10,000,00	10,00,000
4	Payback Period	2 year	6year

From Table XII we can say that alternative 1 is more cost-effective than other alternatives. Also, Alternative 1 is more preferable than all other alternatives as it returns the investment in the least amount of time. So, Alternative 1 can be taken up to continue the project

2.6 Conclusion

Here we have proposed two alternatives-

Alternative 01: Ticket Booking Management web application

Alternative 02: Ticket Booking Management web and mobile application

All of the alternatives are both technical and operational feasible. But by Economical analysis, we have found that Alternative 1 is more beneficial than all other alternatives according to time and money. Also, it returns with a 5,40,000 BDT profit within the system life cycle. So finally, we have preferred Alternative 1 for our project.

Chapter 3

Project Planning

French writer Antoine de Saint-Exupéry quoted that, “A goal without a plan is just a wish”. The main objective of the project is to develop Ticket Booking Management System to make the current system easy, robust, reliable, efficient, and cost-effective as well as helps to make the online ticket booking system easier and manage all works of Ticket Booking easy. But without proper planning, the project is not possible.

The main purpose of this chapter is to plan, estimate cost, break the whole project tasks into smaller activities, schedule the project development and analyze different types of risks and prepare preliminary solutions to the problem. The budget needed for developing the system is about 4,60,000 BDT and the time needed for completing the project is approximately six months.

This chapter covers eight sections. Section 2 discusses how the development team is organized and the team member's role in the team. Section 3 illustrates possible project risks, the likelihood of these risks, strategies for risk reduction. Hardware and software resource requirements are discussed in section 4. In section 5, the breakdown of the project into activities, milestones, and deliverables are discussed. Section 6 is on a project schedule that shows dependencies between activities, estimated time to reach each milestone, and allocation of people to activities. Section 7 discusses monitoring and reporting mechanisms. The concluding remarks are given in Section 8.

3.1 Project Organization

Organizational Structure determines the relationship between functions and positions as well as subdivides and assigns roles, responsibilities, and authorities to carry out different tasks. A proper organizational structure is needed to balance many tasks efficiently and effectively. The project tasks are divided into some activities and the project team consists of some specific groups. The role of the team members in the development process of the system is given in Table 3.1

3.2 Risk Analysis

An uncertain event that harms project development is called a risk. It may affect the project schedule or affect the quality of the software. That's why, risk management is important to ensure the project's efficiency, counter any problems that may affect the creation or development of the required system. The process of risk management involves identification of the risks that may affect the project, analyzing the likelihood of the risk, making plans to counter identified risks, monitor to take action when problems arise.

Table 3.1: Role of Team Member

Serial No	Name	Roles	Responsibilities
01	Mr. Armanuzzaman Chowdhury	Project Manager	<ol style="list-style-type: none"> 1. Lead and manage and coordinate the project team. 2. Recruit project staffs and consultants. 3. Develop and maintain a detailed project plan. 4. Manage project deliveries within constraints. 5. Monitor project progress and performance.
02	Md Riyajul Jannat	Analyst	<ol style="list-style-type: none"> 1. Analyze the requirement of the system and system design. 2. Create and deliver Software Requirement Specification Document (SRS).
03	Md Riyajul Jannat	Designer	<ol style="list-style-type: none"> 1. Design the system as well as the database structure. 2. Create Design Document (DD).
04	Md Riyajul Jannat	Coder	<ol style="list-style-type: none"> 1. Allocate necessary resource needed. 2. Implement the system according to the design document
05	Md Riyajul Jannat	Tester	<ol style="list-style-type: none"> 1. Check the system for errors. 2. Suggests improvement for the system.

3.2.1 Risk Identification

In this step, the possible risks the project may encounter are identified. This process also includes analyzing the possibility of the risk and also the effect of the risk if occurred. Table 3.2 lists such risks and their likelihood and effects are described in table 3.3.

Table 3.2: List of Possible risks

Serial No.	Risk Name	Risk Type	Affects	Description
1	Staff Turnover	People	Project	Experienced staff leaves the project before it is finished.
2	Defective Component	Tool	Project	The components used in the project turns out to be defective.
3	Hardware Unavailability	Technology	Project	Hardware essential for the project not delivered on schedule.
4	Requirements Change	Requirement	Project and Product	Changes to requirements that require major design rework is proposed.
5	Product Competition	Organizational	Business	A competitive product is marketed before the system is completed.
6	Staff Unavailable	People	Project	Key staff falling ill on critical time.
7	Software Integration Problem	Tool	Product	Software tools not working together in an integrated way.
8	Database Failure	Technology	Product	The database used in the system cannot process as many transactions per second as expected.
9	Financial Crisis	Organizational	Project	Organizational financial problems force reductions in the project budget.
10	Time Estimation	Estimation	Project	The time required to develop the software is underestimated.

Table 3.3: Possibility and Effects of the risks

Serial No.	Risk Name	Possibility	Affects
1	Staff Turnover	Low	Catastrophic
2	Defective Component	Low	Tolerable
3	Hardware Unavailability	Moderate	Serious
4	Requirements Change	Moderate	Serious
5	Product Competition	Low	Serious
6	Staff Unavailable	Moderate	Serious
7	Software Integration Problem	High	Tolerable
8	Database Failure	Moderate	Serious
9	Financial Crisis	Low	Catastrophic
10	Time Estimation	High	Serious

3.2.2 Risk Reduction Strategies

The analysis of the risks is not sufficient. The strategies have to be devised for the time the risk arises to minimize its effect. The risk reduction strategies are described in Table 3.4

Table 3.4: Risk Reduction Strategies

Serial No.	Risk Name	Reduction Strategies
1	Staff Turnover	Keeping staff motivated, giving job security, and providing a supportive environment.
2	Defective Component	Buying components from well-known sources and changing the component as soon as possible.
3	Hardware Unavailability	Required hardware is purchased beforehand at the starting of the project.
4	Requirements Change	Discuss the impact of the requirement change with the customer.
5	Product Competition	Increase publicity and adding a new feature to make the product better than the competitor.
6	Staff Unavailable	Reorganizing team having more overlap of work and people, therefore, understand each other's jobs
7	Software Integration Problem	Changing the software with more compatible ones.
8	Database Failure	Investigate the possibility of buying a higher-performance database.
9	Financial Crisis	Prepare a briefing document for senior management showing how the project is making a very important contribution to the goals of the business and presenting reasons why cuts to the project budget would not be cost-effective.
10	Time Estimation	Overlap work and people, distribute work to outsource developers.

3.3 Hardware and Software Resource Requirement

For developing an effective system some hardware and software are needed. Without appropriate hardware and software, a project continuation may be endangered. Good quality hardware ensures environment stability as well as system performance and good software helps rapid development process and reduces the possibility of system crashes. Table 3.5 lists the hardware and software resources needed for the project.

Table 3.5: Hardware & Software Resource Requirement

Resource Type	Serial No	Resource Name	Quantity
Hardware	1	Personal Computer (Intel Core i5 8600K 3.6 GHz, 8GB DDR4 RAM)	05
	2	LAN Connection (10 MPs)	01
	3	Hard Drives (250 GB SSD)	03
	4	Keyboard and Mouse	05
Software	5	Operating System	05
	6	Database Management System	01
	7	Visual Studio 2019	01

3.4 Work Breakdown

A project is a huge collection of work. So, the works have to be broken down into some tasks to make the execution of the project hassle-free and easy. It is also important to set milestones to track the progress of the project. Also, some deliverables are produced after completing tasks to deliver to either project staffs or to the client. Tasks of the project are summarized in Table 3.6 and Table VII outlines the milestones and deliverables.

Table 3.6: Task Summary

Task	Name of tasks	Achievement
T1	Interviewing Staffs	
T2	Arranging Meeting with Stakeholders	
T3	Defining Function and Facilities	
T4	Preparing System Requirement Specification Document	Milestone (M1)
Deliverable(D1)		Deliverable(D1)
T5	Designing System Structure	
T6	Designing Database Structure	
T7	Preparing Detailed Design Document	Milestone (M2)
Deliverable(D2)		Deliverable(D2)
T8	Allocate Resources	
T9	Database Specification	
T10	Implementation of the System	Milestone (M3)
T11	Testing of the System	Milestone(M4)
T12	Delivering of the System	Deliverable(D3)

Table 3.7: Milestone & Deliverables

Type	Name	Description
Milestones	M1	Analysis of the System Completed
M2	Designing of the System Completed	
M3	Implementation of the System Completed	
M4	Complete Testing of the System Completed.	
Deliverables	D1	System Requirement Specification Document (SRS)
D2	Design Document (DD)	
D3	Complete System	

3.5 Project Schedule

Project scheduling is the process of deciding how the works in a project will be organized in as separate tasks and when and how these tasks will be executed. It is done by estimating calendar time and effort required to complete each task. It also identifies who will work on these tasks. To organize and complete a project in a timely, quality and financially responsible manner a proper scheduling of the project is very important. The number of persons assigned to the task, the duration of that task, and dependencies of the tasks is included in schedule representation which is shown in table VIII.

Table 3.8 Schedule Representation

Tasks	Effort (Person – Days)	Start Date	Duration (Days)	Dependencies
T1	07	November 01, 2020	05	
T2	03	November 6, 2020	03	
T3	05	November 10, 2020	08	T1, T2
T4	05	November 18, 2020	10	T3
T5	10	November 18, 2020	10	T4 (M1)
T6	05	November 28, 2020	20	T4 (M1)
T7	05	November 28, 2020	5	T5, T6
T8	05	December 2, 2020	5	T7 (M2)
T9	03	December 7, 2020	20	T7 (M2)
T10	10	December 28, 2020	60	T7 (M2)
T11	05	February 28, 2021	10	T8, T9, T10 (M3)
T12	05	March 10, 2021	10	T11 (M4)

The tasks are distributed among project members to make the work done efficiently and effectively. The Task distribution is shown in Table IX and staff allocation is shown graphically in Figure 6.2

Table 3.9 Task Distribution

Serial No	Task	Assigned Member
1	T1, T2, T3, T4	Mr. Armanuzzaman Chowdhury
2	T5, T6, T7	Md Riyajul Jannat
3	T8, T9, T10	Md Riyajul Jannat
4	T11	Md Riyajul Jannat
5	T12	Md Riyajul Jannat

3.6 Monitoring and Reporting Mechanism

Monitoring is the regular observation and recording of activities taking place in a project. It is a process of routinely gathering information on all aspects of the project. To monitor is to check on how project activities are progressing. It is a systematic and purposeful observation. Monitoring also involves giving feedback about the progress of the project to the donors, implementers, and beneficiaries of the project. Monitoring is very important in project planning and implementation as well as risk management. Reporting enables the gathered information to be used in making decisions for improving project performance. A project manager can monitor the project in various ways. The project can be monitored in the following way:

- Holding meetings weekly and monthly with the project staff where everyone will show their progress.
- Monitoring the progress of the project daily by collecting information about progress, problems, and difficulties.
- Comparing the project environment with potential risk indicator situations.
- Comparing the performance of the staff with a standard and take necessary action to correct it.

The project staff can report to the project manager as mentioned below:

- The project team will notify the manager of the completion of the task.
- The project staff will generate and deliver important documents to the manager.
- The staff will inform and generate a report about the milestones completed.

3.7 Conclusion

A plan the first step in completing a project successfully. This chapter has provided all the information about project organization, risk analysis, hardware and software requirements, work breakdown, scheduling, monitoring, and report mechanism of the project. All those tasks are so important for the completion of the project. This will help the project staff to complete the project on time. Finally, we expect this project to be completed successfully.

Chapter – 4

Software Requirements Specification

4.1 Preface

This is „System Requirements Specification’ document (generally known as „SRS’ document) of a software system named „**Ticket Booking Management System**. It is primarily intended to be proposed to a customer for its approval and a reference for developing the first release of the system for the development team. With version 1.0, the users will experience a completely stable release that includes high authentication of the users like passenger or. So, there is no worry about unauthenticated users. Version 1.0 will better help to book ticket online easily by check available bus and compare price with other ticket and confirm ticket by payment through bKash and submit it into system. Passengers will get better experience for book ticket.

4.2 Introduction

A software requirements specification document (SRS) is a description of a software system to be developed. It lays out functional and non-functional requirements and includes a set of use cases that describe user interactions that the software must provide. Software requirements specification document establishes the basis for an agreement between customers and contractors or suppliers on what the software product is to do as well as what it is not expected to do.

The purpose of this SRS document is to give a detailed description of the requirements for the “**Ticket Booking Management System**”. This document will completely illustrate the purpose and features of the system. It will provide a complete declaration for the development of the system in a precise and explicit manner. It will also explain system constraints, what system will do, and how the system will react to external interactions.

It'll be designed to make the Ticket Booking work easier and faster. At first, Passenger will search for ticket by placing location and journey date after get available ticket list they will choose ticket by comparing price and book ticket. Passengers can see upcoming events and book ticket in advance. Agency can add their bus, launch, air, events, movie information, agency can manage booked ticket list. Admin can manage passengers, agencies information, admin can manage admin information.

The chapter contains ten sections. Section 1 gives a preface to this document while section 2 introduces the readers with the system as well as with this document. Section 3 familiarize the readers with the technical terms used in this document. Section 4 narrates the user requirements briefly, while section 5 gives a brief description of the system architecture. The specific description of the user requirement is described in section 6. Use case scenarios of the system are given in section 7 and section 8 bears the anticipated change or evolution of the system due to hardware changes. Section 9 and section 10 contains the appendices and index for this document consecutively.

4.3 Glossary

Glossary is an alphabetical list of terms in a particular domain of knowledge with the definitions for those terms. It lists the technical terms used in the document. The glossary for this document is given in table 4.1.

Table 4.1: Glossary

Technical Term	Description
Authentication	The process or action of verifying the identity of a user or process.
Backup	A copy of a file or other item of data made in case the original is lost or damaged.
Constraints	The limiting barrier of an action or a system.
Credentials	A group of information proving a user's identity or qualifications.
Database	A collection of information organized into rows, columns, and tables, such a way that a computer program can quickly access, manage or update desired pieces of data.
Encryption	The process of converting information or data into a code, especially to prevent unauthorized access.
Login	The process by which an individual gains access to a computer system by identifying themselves.
Online	Operating being connected to a computer or telecommunication system such as the internet.
Response Time	The length of time taken for a system to react to a given event.
Server	A computer or computer program which manages access to a centralized resource or service in a network.

4.4 User Requirements Definition

Requirements are physical or functional need that a particular design, product or process aims to satisfy. After meeting with the client and properly discussing with them, some requirements are discovered. The requirements are divided into two categories such as functional requirements, which defines the functions of the system required by the client, and, nonfunctional requirements, which defines the characteristics as well as constraints of the system. The user requirements are defined in table 4.2.

Table 4.2: Definition of User Requirements

Requirement Type	Definition of Requirements
Functional Requirement	1. Passenger & agency will registration with valid information.
	2. Agency will add & can manage bus, launch, air, movie, event information.
	3. Passengers will book ticket by comparing price.
	4. Admin can manage passenger, agency & admin information,
	5. Agency can manage booked ticket list
	6. Passenger can print booked ticket.
Non-functional Requirement	7. The system must be highly secure and reliable.
	8. Every operation must be protected by authentication.
	9. The system must respond quickly.

4.5 System Architecture

A system architecture is the conceptual model that defines the structure of a system. A system architecture description could be a formal description and illustration of a system, organized during a manner that supports reasoning regarding the structures and behaviors of the system. The system architecture of the **Ticket Booking Management System** in Figure 4.1.

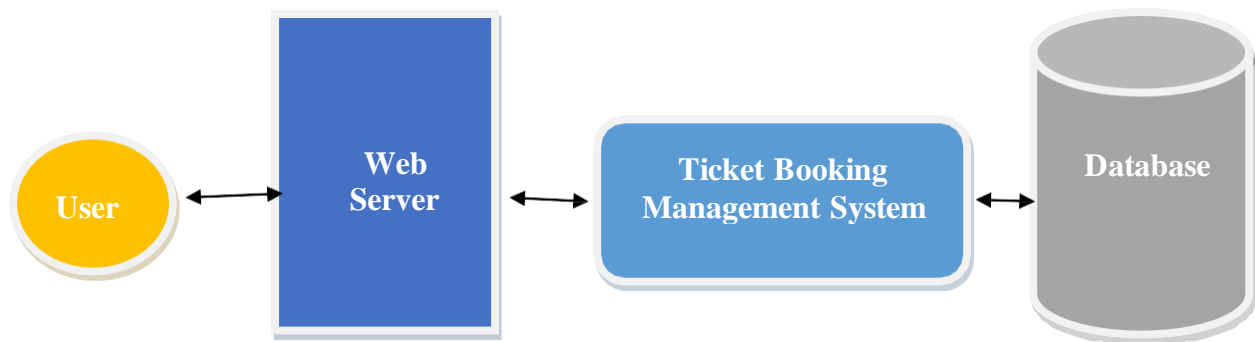


Fig 4.1: System Architecture

4.6 User Requirements Specification

The software requirements specification document enlists the requirements that are required for the project development. To derive the requirements, the developer needs to have a clear and thorough understanding of the products to be developed. Also, the requirements must be described at length for making the client clear and concise about what is going to be developed. The user requirements are described in great detail in table 4.3.

A. Use Cases

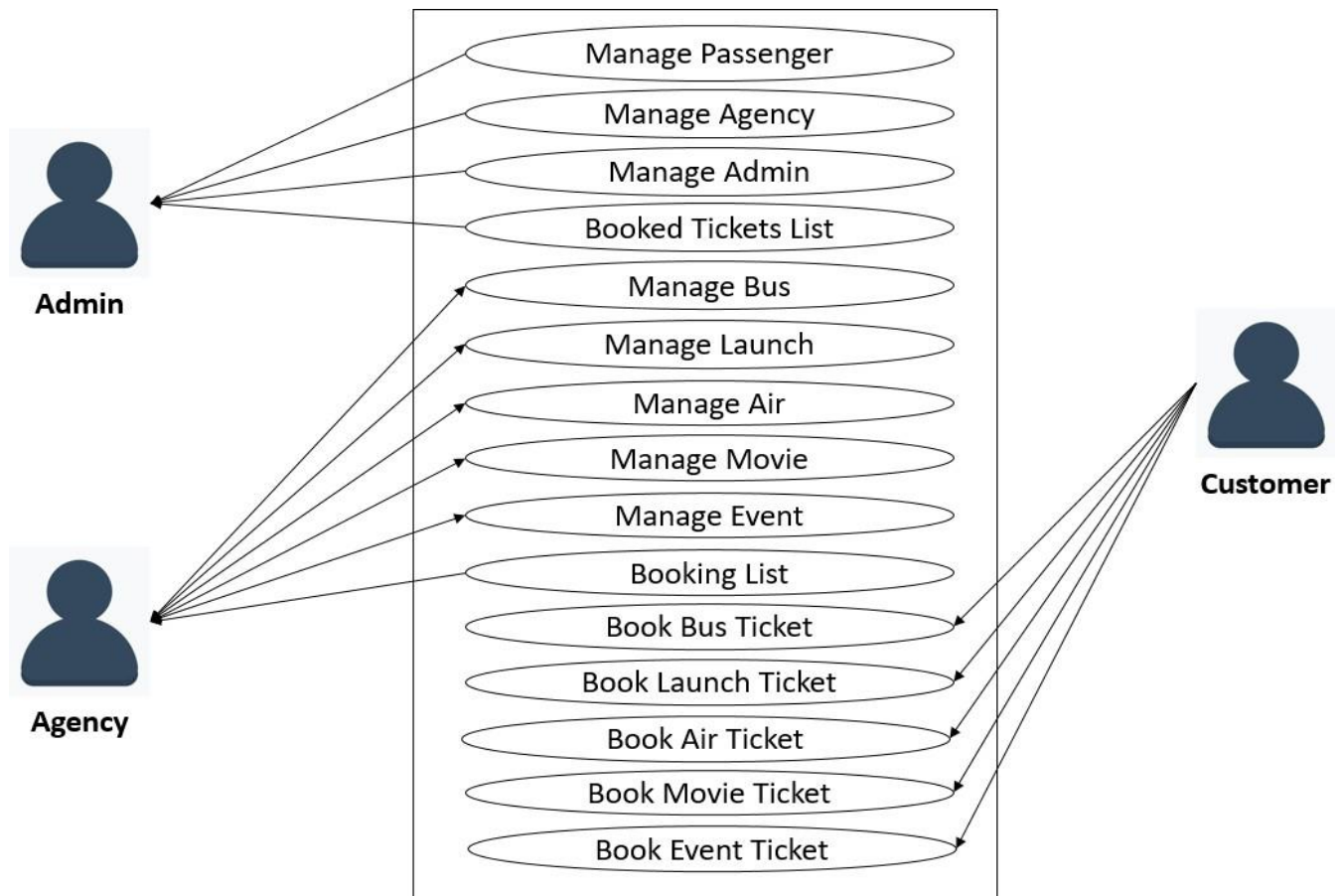
A use case is a list of actions or event steps typically defining the interactions between a role and a system to achieve a goal. Use cases are important to understand how the system interacts with the user or other systems. The use cases of the system are listed in table 4.4.

Table 4.4: List of Use Cases

Use Case	Title
UC1	Manage Passenger
UC2	Manage Agency
UC3	Add & Manage Admin
UC4	Booked Tickets List
UC5	Add & Manage Bus
UC6	Add & Manage Launch
UC7	Add & Manage Air
UC8	Add & Manage Movie
UC9	Add & Manage Event
UC10	Booking Ticket List for Company
UC11	Book Bus, Launch, Air, Movie, Event Ticket

B. Use Case Diagram

The use case diagram shows the interaction between the user and the system graphically. The use case diagram is shown in figure 4.2



Here, Admin can manage passenger, agency and admin. Admin can view booked ticket also. Agency can manage bus, launch, air, movie and event info. Agency check their booked list. Passenger can book bus, launch, air, movie and event ticket. Passenger can print their ticket and also can view upcoming events.

4.7 System Evolution

Over time, software systems, programs as well as applications, continue to develop. These changes will require new laws and theories to be created and justified. Software evolution is the term used in software engineering to refer to the process of developing software initially, then repeatedly updating it for various reasons. **Ticket Booking Manage System** is developed to be an adaptive system. It is implemented in such a way that it adjusts its performance concerning the specification of the hardware such as servers. With version 1.0, the system can process as much as 200 requests per second for a server with a processor speed of 3.3 GHz, consisted of 4 logical cores and a maximum memory of 4GB. Any change to the hardware would change the performance of the system in proportion to the change of processor speed, number of cores, and maximum memory.

4.8 Appendices

Appendices contain the texts that is explanatory, statistical, or bibliographic. The appendix for this document contains the hardware specification, database specification for the system.

Appendix A: Hardware Specification

The system is developed using the server “HPE ProLiant ML10 Gen9 Tower Server”. The specification of the server is given in Table 4.5:

Table 4.5: Server Specification

Processor	Intel® Xeon® E3-1225 v5
Number of Processors	1
Processor Core Available	4
Processor Cache	8MB (1 x 8MB) Level 3 cache
Processor Speed	3.3GHz
Chipset	Intel® C236 Chipset
Power Supply Type	300W Multi-Output Power Supply
Memory	4GB DDR4
Memory Slots	4 DIMM slots
Memory Type	1R x8 PC4-2133P-E-15
Memory Protection Features	Un-buffered ECC
Included Hard Drives	LFF SATA; 1TB
Maximum Internal Storage	24TB
Optical Drive Type	SATA 9.5mm DVD RW
System Fan Features	Non-Pluggable Fan
Network Controller	Intel® Ethernet Connection I219-LM
Storage Controller	Integrated SATA RAID
Infrastructure Management	Intel® Active Management Technology (Intel® AMT 11.0)

Appendix B: Database Specification

The system uses the “MSSQL Enterprise Edition” as a database management system. The technical specification of the database management system is given in Table 4.6.

Table 4.6: Database Specification

Version	5.7
Data Type	Static
Architecture	Relational Model
Operating System	Linux, Solaris, FreeBSD, Mac OS, Windows
Software License	GNU General Public License
Interface	GUI, CLI
Security	<ul style="list-style-type: none">• SSL Support• Built-in Data Encryption/Decryption• View Support• Triggers for auditing• Query Logs for auditing
Access Control	<ul style="list-style-type: none"><input type="checkbox"/> Enterprise Directory Compatibility<input type="checkbox"/> Native Network Encryption<input type="checkbox"/> Run Privilege<input type="checkbox"/> Security Certification
Indexes	<ul style="list-style-type: none">• R-/R+ Tree• Hash• Full-text• Spatial
Partitioning	<ul style="list-style-type: none"><input type="checkbox"/> Range, Hash, List, Key<input type="checkbox"/> Composite<input type="checkbox"/> 8k partitions per table

	<input type="checkbox"/> Portable partitions between tables <input type="checkbox"/> Explicit querying by partition
	<input type="checkbox"/> Transparent Pruning
Max Database Size	Unlimited
Max Table Size	256 TB (MyISAM) 64 TB (InnoDB)
Max Row Size	64 KB
Max Column Per Row	4096
Max Blob/Clob Size	4 GB
Max CHAR Size	64 KB
Max NUMBER Size	64 Bit
Min Date Value	1000
Max Date Value	9999
Max Column Name Size	64

4.10 Index

List of Figures

Figures are a graphical representation of information. The figures used in this document are listed in Table 5.7

Table 4.7: Figures are graphical representation

Figure Name	Name of Figure	Page No
Figure 5.1	System Architecture	20
Figure 4.1	Use Case Diagram	21

Alphabetical Index

A

Authentication, 21-23

C

Constraint, 21-22

Credentials, 22,25

D

Database, 22, 26, 32, 33

E

Encryption, 22, 25, 33

F

Failure Rate, 25

L

Login, 22, 24-27

O

On-line, 22-23

Operation, 23, 25

R

Requirement,

System Requirement, 21

Functional Requirement, 21-24

Non-functional Requirement, 21-22

User Requirement, 21-24

Response Time, 22, 25

S

Scenario, 22, 26-31

Server, 22, 26-30, 32

Software, 21, 23-26, 32-33

System Architecture, 22-23

System Model,
Conceptual Model, 23

T

Table VII: List of Figures

U

Use Case, 21, 22, 26, 31

Chapter – 5

System Designing

Ticket Booking Management System is a computerized system developed to manage Ticket Bookings. The purpose of this chapter is to provide design details of the Ticket Booking Management System. This chapter includes the architectural design of the system and also the system model from external, structural, interactional, and behavioral perspectives.

The chapter is composed of five sections. Section 1 describes the architectural pattern of the system, class diagram of the system is shown in section 2, Section 3 shows the code map of the system, Section 4 shows the er-diagram of the system and the conclusion for this chapter is given in section 5.

5.1 Architectural Pattern

An architectural pattern is a general, reusable solution to a commonly occurring problem in software architecture within a given context. An architectural pattern defines the systems in terms of structural organization, components, connectors, and constraints on how they can be combined. It also addresses various issues in software engineering, such as performance, availability, reliability, maintainability, and security of a system. Bit Online Ticket Booking Web System (TBMS) System has a Three-tier architecture pattern. The architectural pattern of TBMS system is graphically shown in figure 5.1

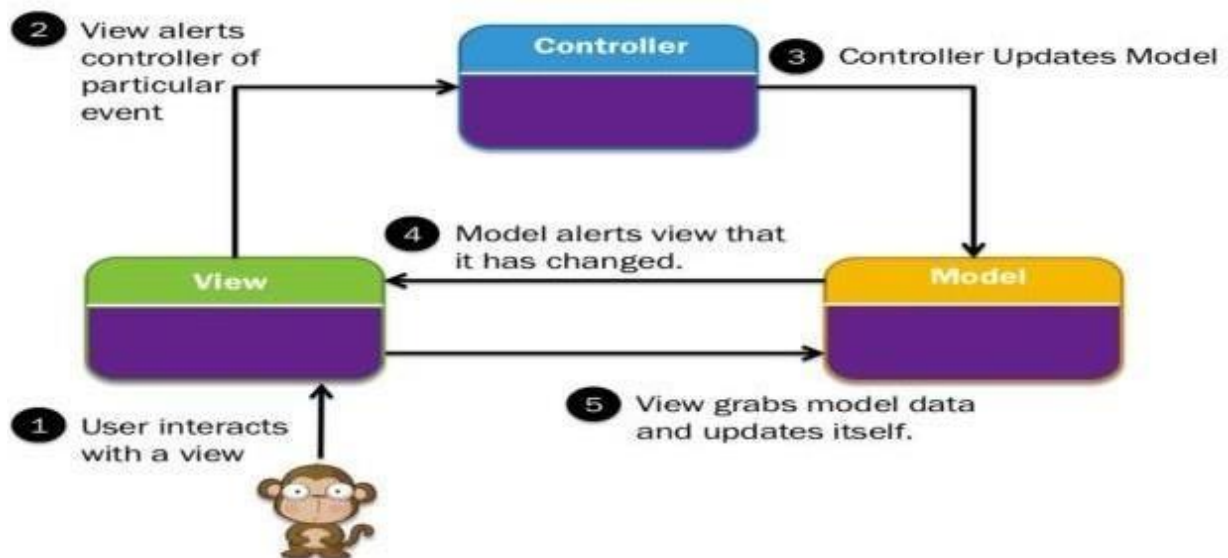


Figure 5.1: Architectural Pattern of Ticket Booking Management System

5.2 System Diagram

A system diagram could be a visual model of a system, its elements, and their interactions. With supporting documentation, it will capture all the essential data of a system's style. There square measure several variations of schematization vogue that everyone represents this rubric. the fashion conferred here is meant to be optimally in line with the remainder of this courseware.

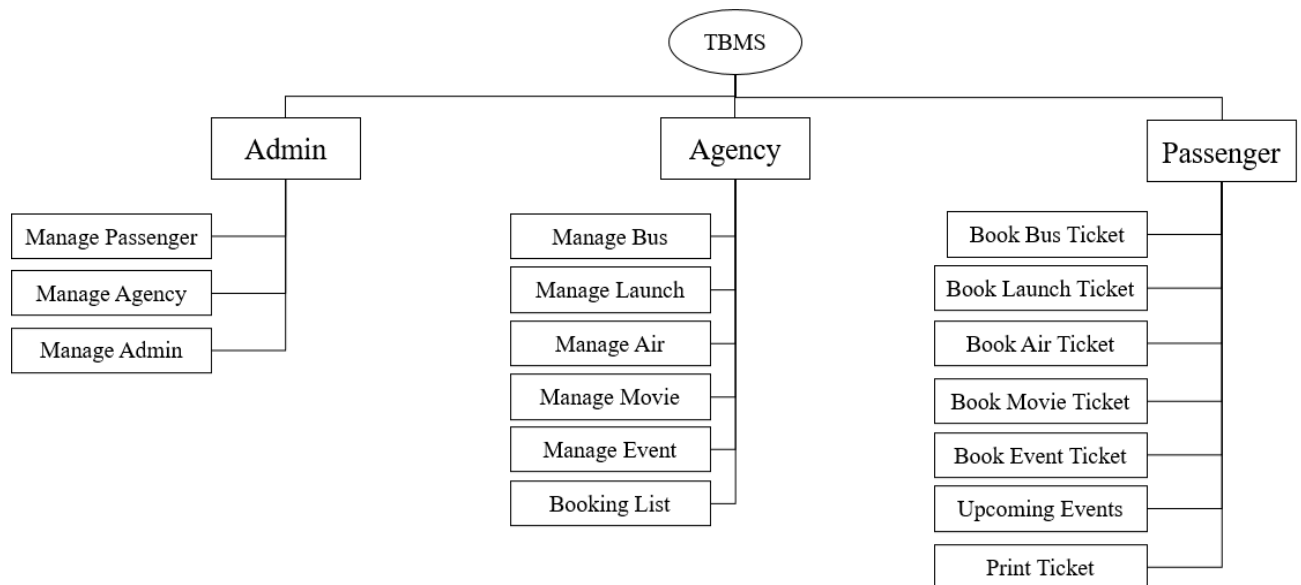


Figure 5.2: System Diagram

5.3 ER Diagram

Entity-relationship diagram displays the relationships of entity set hold on in an exceedingly information. In alternative words, we will say that ER diagrams assist you to elucidate the logical structure of databases. initially look, associate degree ER diagram appearance terribly the same as the flow chart. However, ER Diagram includes several specialized symbols, and its meanings build this model distinctive.

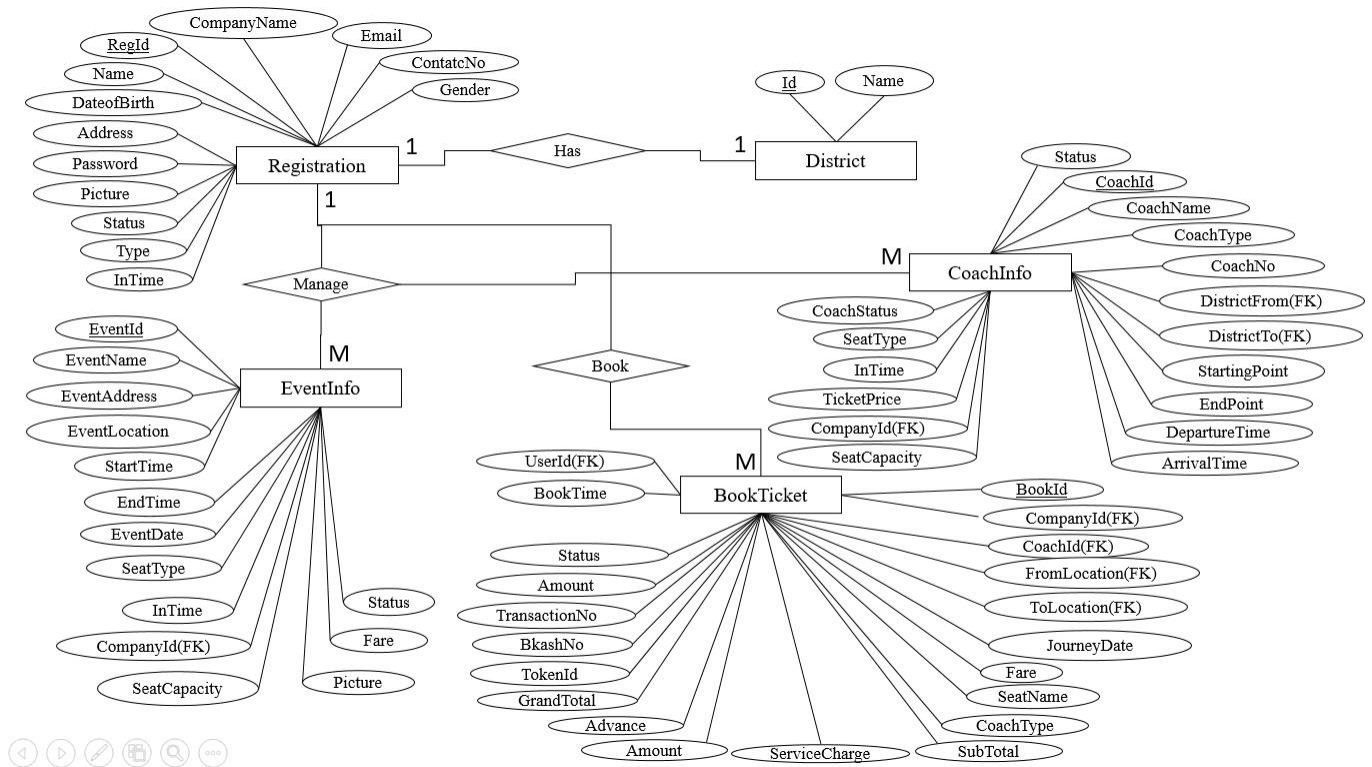


Figure 5.3: ER Diagram

5.4 Conclusion

Implementation is the process of converting the design into an executable program. The design chapter for the Ticket Booking Management System will help the developers to develop the system efficiently and also enable the testers to verify and validate the system precisely, therefore creating a system that fulfills the user requirements and meeting the stakeholder's expectation entirely.

Chapter – 6

Implementation

Ticket Booking Management System is a computerized system for management Ticket Bookings all works easily. The user interface (UI) is everything designed into an information device with which a person may interact. This can include display screens, keyboards, a mouse, and the appearance of a desktop. It is also the way through which a *user* interacts with an application or a website.

6.1 System Demonstration

The Ticket Booking management system is a web application with several functionalities. The functionalities of the system are shown using screenshots in the following figure 6.1 through 6.25.

Home

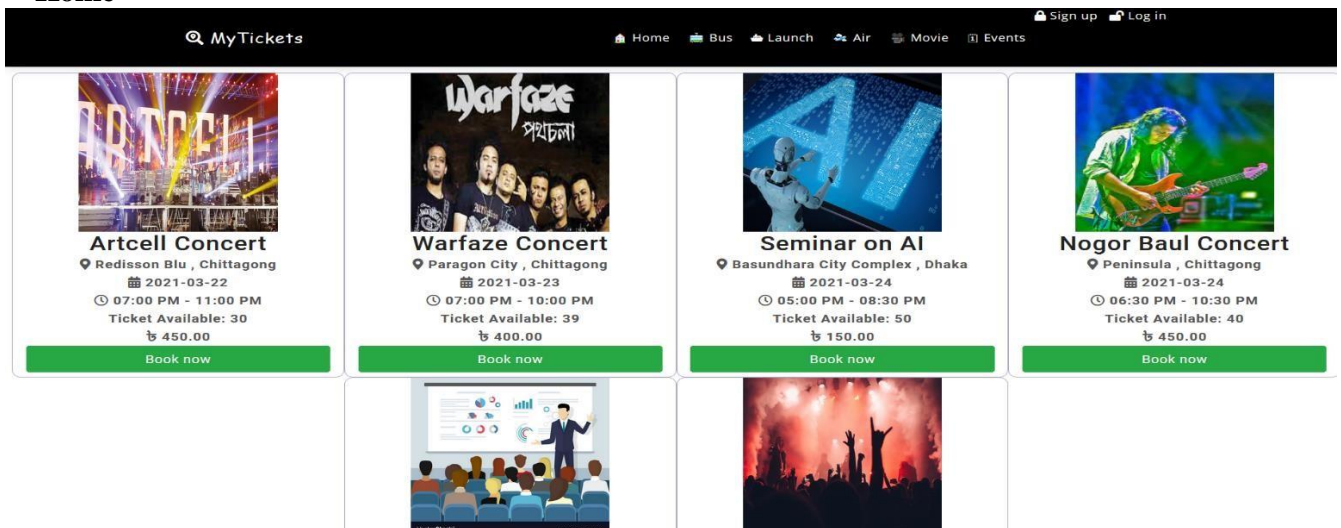


Fig 6.1: Home

Book Bus Ticket

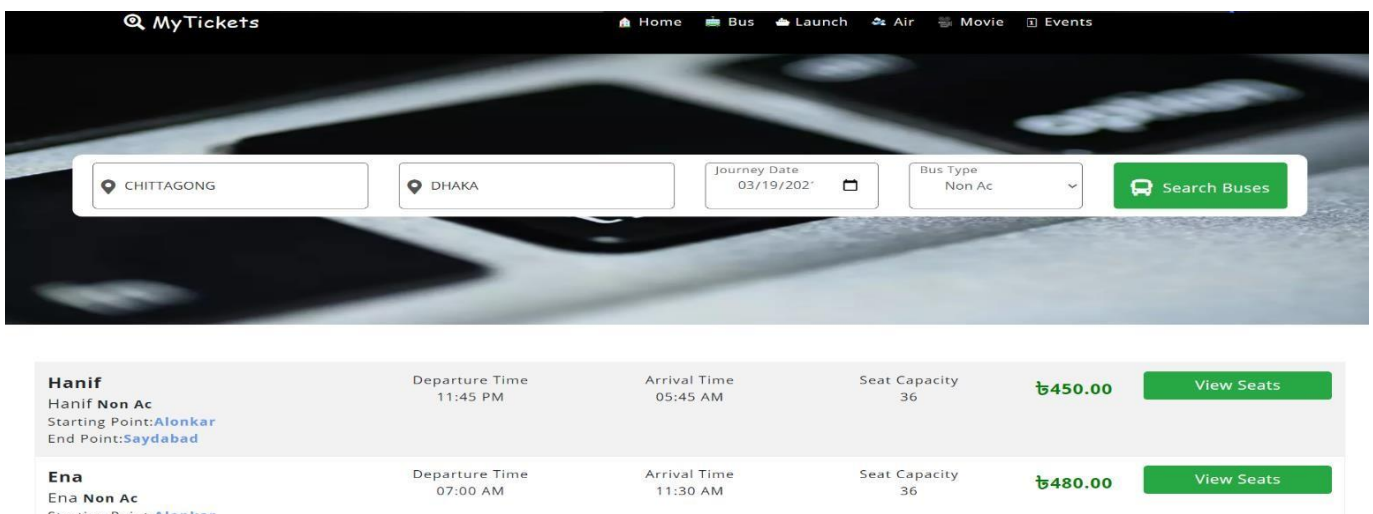


Fig 6.2: Book Bus Ticket

Book Bus Ticket

Booked

Available

Selected

A1	A2	A3	A4
B1	B2	B3	B4
C1	C2	C3	C4
D1	D2	D3	D4
E1	E2	E3	E4
F1	F2	F3	F4
G1	G2	G3	G4
H1	H2	H3	H4

Seat Information

Seat Name	Fare
C3	450
C4	450

Sub Total: 900৳

Service Charge: 50৳

Grand Total: 950৳

Confirm Ticket

MyTickets

Home
Bus
Launch
Air
Movie
Events
Niloy...

Fig 6.3: Book Bus Ticket

Book Launch Ticket

DHAKA

BARISAL

Journey Date
03/20/2021

Search Launches

Star line
Parabot Non Ac
E Class
Starting Point: Sadarghat
End Point: Kirtonkhola

Departure Time
11:30 PM

Arrival Time
06:00 AM

Seat Capacity
60

৳450.00

Book Tickets

Green Line
Green Line Non Ac
B Class
Starting Point: Sadarghat
End Point: Kirtonkhola

Departure Time
08:30 AM

Arrival Time
02:30 PM

Seat Capacity
100

৳750.00

Book Tickets

MyTickets

Home
Bus
Launch
Air
Movie
Events
Niloy...

Book Launch Ticket

Seat Available
60

Sub Total
900

Total Price
950

Number of Seats
2

Service Charge
50

Payment Method

Please pay BDT 190 as advance for book your ticket. You won't get it back if you cancel your ticket. Bkash your advance to this number 01866335118. Use "WRPXOFYC" as reference id.

Bkash No.(Last 6 digits only)

XXXXXX

Transaction no.

XXXXXXXXXX

Amount

190

Confirm Ticket

Fig 6.4: Book Launch Ticket

Book Air Ticket

MyTickets

[Home](#)
[Bus](#)
[Launch](#)
[Air](#)
[Movie](#)
[Events](#)

[Sign up](#)
[Log in](#)

CHITTAGONG

DHAKA

Journey Date
03 / 23 / 2021

Search Air

	US Bangla BS-110	Departure Time 08:00 PM	Quickest Time 08:50 PM	Seat Capacity 50	₳3200.00	Book Tickets
	Biman Bangladesh BG-215	Departure Time 07:00 PM	Quickest Time 07:45 PM	Seat Capacity 50	₳3000.00	Book Tickets

MyTickets

[Home](#)
[Bus](#)
[Launch](#)
[Air](#)
[Movie](#)
[Events](#)

Niloy...

Book Air Ticket

Seat Available
50

Sub Total
Sub Total

Total Price
0

Number of Seats
0

Service Charge
Service charge

Payment Method

Please pay BDT as advance for book your ticket. You won't get it back if you cancel your ticket. Bkash your advance to this number .Use "RYSMZEXT" as reference id.

Bkash No.(Last 6 digits only)

XXXXXX

Transaction no.

XXXXXXXXXX

Amount

XXXXXXXXXX

Confirm Ticket

Fig 6.5 Book Air Ticket

Book Movie Ticket

MyTickets

[Home](#)
[Bus](#)
[Launch](#)
[Air](#)
[Movie](#)
[Events](#)

[Sign up](#)
[Log in](#)

MOVIE NAME

CHITTAGONG

Premier Date
03 / 22 / 2021

Search Movies

	Inception Cineplex , Chittagong A Class	Movie_Time 08:30 PM - 10:00 PM	Premier_Date 2021-03-22	Seat Capacity 40	₳450.00	Book Tickets
	Joker Almas , Chittagong A Class	Movie_Time 07:00 PM - 09:30 PM	Premier_Date 2021-03-22	Seat Capacity 40	₳400.00	Book Tickets

Book Movie Ticket

You'll get your seat serially by counting number of tickets you bought.

Seat Available
30

Sub Total
Sub Total

Total Price
0

Number of Seats
0

Service Charge
Service charge

Payment Method

Please pay BDT as advance for book your ticket. You won't get it back if you cancel your ticket. Bkash your advance to this number .Use "MRSGYJGK" as reference id.

Bkash No.(Last 6 digits only)

XXXXXX

Transaction no.

XXXXXXXXXX

Amount

XXXXXXXXXX

Confirm Ticket

Fig 6.6 Book Movie Ticket

Book Event Ticket

The screenshot shows the MyTickets website interface. At the top, there's a navigation bar with links: Home, Bus, Launch, Air, Movie, Events, Sign up, and Log in. Below the navigation bar is a search bar with a location dropdown set to 'CHITTAGONG' and a 'Search Events' button. The main content area displays two event listings:

Event Image	Event Name	Location	Event Time	Event Date	Ticket Limit	Price	Action
	Nogor Baul Concert	Peninsula, Chittagong	06:30 PM - 10:30 PM	2021-03-24	40	₹450.00	Book Tickets
	Seminar on Business	Hall 24, Chittagong	05:30 PM - 08:00 PM	2021-03-24	60	₹100.00	Book Tickets

At the bottom, there's a footer with the MyTickets logo, navigation links, and a user profile icon labeled 'Niloy...'.

Book Event Ticket

You'll get your ticket serially by counting number of tickets you bought.

Ticket Available: 150

Number of Ticket:

Sub Total:

Service Charge:

Total Price:

Payment Method

Please pay BDT as advance for book your ticket. You won't get it back if you cancel your ticket. Bkash your advance to this number .Use "EHSMNMDN" as reference id.

Bkash No.(Last 6 digits only):

Transaction no.:

Amount:

Fig 6.7 Book Event Ticket

Sign up as Customer

Sign up

Name:

Email:

Contact No.:

Gender:

Date of Birth:

Address:

New Password:

Confirm Password:

Next

Fig 6.8 Sign up as Customer

Sign up as Agency

Sign up

Profile Creator Name:

Company Name:

Email:

Contact No.:

Gender:

Main Office Address:

New Password:

Confirm Password:

Next

Fig 6.9 Sign up as Agency

Login

Log in

Email

example@example.com

Password

[Forgot password](#)

Log in

[Back to home](#)

Fig 6.10 Login

Add Bus

MyTickets

Bus Info

Add Bus

Bus List

Launch Info

Add Launch

Launch List

Air Info

Add Air

Air List

Movie Info

Event Info

John
Doe

Niloy ▾

Add Bus

Bus Name

xyz

Bus No.

XX-XXX

Bus Type

Select ▾

District From

~SELECT~ ▾

District To

~SELECT~ ▾

Starting Point

Place Name

End Point

Place Name

Departure Time

Fig 6.11: Add Bus

Add Launch

MyTickets

Bus Info

Launch Info

Air Info

Movie Info

Event Info

Booking List

John
Doe

Niloy ▾

Add Launch

Launch Name

xyz

Launch No.

XXX-XXX

Launch Type

Select ▾

District From

~SELECT~ ▾

District To

~SELECT~ ▾

Starting Point

Place Name

End Point

Place Name

Departure Time

--:--

Arrival Time

Fig 6.12: Add Launch

Add Air

MyTickets

Bus Info

Launch Info

Air Info

Movie Info

Event Info

Booking List

JohnDoeNiloy

Add Air

Flight No

Flight-XX

District From

--SELECT--

District To

--SELECT--

Departure Time

--:--

Quickest Time

--:--

Seat Capacity

Ticket Price

BDT XXX

Air Status

Select

Add Air

Fig 6.13: Add Air

Add Movie

MyTickets

Bus Info

Launch Info

Air Info

Movie Info

Event Info

Booking List

JohnDoeNiloy

Add Movie

Movie Name

xyz

Theatre Address

Location,Thana

District

--SELECT--

Start Time

--:--

End Time

--:--

Movie Premier Date

mm / dd / yyyy

Seat Type

Select

Seat Capacity

0

Fig 6.14: Add Movie

Add Event

MyTickets

Bus Info

Launch Info

Air Info

Movie Info

Event Info

Booking List

JohnDoeNiloy

Add Event

Event Name

xyz

Event Address

Location,Thana

District

--SELECT--

Start Time

--:--

End Time

--:--

Event Date

mm / dd / yyyy

Ticket Limit

0

Ticket Price

BDT XXX

Fig 6.15: Add Event

Agency List

Admin

User List

Passenger List

Agency List

Admin

Create Admin

Admin List

Booked List

Agency List

Active

Search by name

View Booked List

Passenger_Name	Company_Name	Email	Mobile_No.	Address	Picture	Action
sarif	Biman Bangladesh	sarif05@gmail.com	01601607071	Saydabad, Dhaka		✕ Restrict
Sazal	Ena	sazal80@gmail.com	01866335118	Kazir dewri, Chattogram		✕ Restrict
Sazzad	Green Line	moderatech17@gmail.com	01864483548	Dhaka		✕ Restrict
niloy	Hanif	niloy05@gmail.com	01864483548	chittagong		✕ Restrict

Fig 6.16: Agency List

Passenger List

Admin

User List

Passenger List

Agency List

Admin

Create Admin

Admin List

Booked List

Passenger List

Active

Search by name

Passenger_Name	Email	Mobile_No.	Gender	Date_of_Birth	Address	Picture	Action
Minhaz	minhaz@gmail.com	01622115277	Male	2021-01-14	Ctg		✕ Restrict
Niloy Hasan	nil@gmail.com	01954236520	Male	2020-12-02	ctg		✕ Restrict
Niloy Uddin	niloycse05995@gmail.com	01685400001	Male	2020-12-01	ctg		✕ Restrict
sarif	mdsarif940@gmail.com	01521401302	Male	2021-02-26	Lalkhan bazar		✕ Restrict

Fig 6.17: Passenger List

Booked List

Admin

User List

Admin

Booked List

Booked List

Choose Company:

Choose Coach Type:

~SELECT~

~SELECT~

Niloy Uddin niloycse05995@gmail.com	Departure Time: 11:45 PM	Total_Ticket 2	Journey Date: 2021-03-19	Bkash No: 483548 Trans No: 7as5dc4f	Service Charge: 50.00	Advance 190.00	Total: ৳950
Niloy Uddin niloycse05995@gmail.com	Departure Time: 11:45 PM	Total_Ticket 2	Journey Date: 2021-03-19	Bkash No: 836952 Trans No: 1h5gy7vg	Service Charge: 50.00	Advance 190.00	Total: ৳950
Niloy Uddin niloycse05995@gmail.com	Departure Time:	Total_Ticket 2	Journey Date: 2021-03-17	Bkash No:	Service Charge:	Advance 190.00	Total:

Fig 6.18: Booked List

Add Admin

MyTickets

User List

Admin

Booked List

Admin

Add Admin

Name

Mr. X,Y

Email

example@example.com

Contact No.

01XXXXXXXX

Gender

Select

Date of Birth

mm / dd / yyyy

Address

House no.,Area,Thana,District,Division

New Password

Confirm Password

Upload Profile Picture

Browse...

No file selected.




Fig 6.19: Add Admin

Admin List

MyTickets

User List

Admin


Booked List

Admin

Admin List

Active

Search by name

Admin_Name	Email	Mobile_No.	Gender	Date_of_Birth	Address	Picture	Action
Admin	admin@gmail.com	01500112233	Male	1996-08-02	ctg		<div>✕ Restrict</div>

Copyright © 2020 Bits. All rights reserved.

Fig 6.20: Admin List

Chapter – 7

Conclusion

7.1 System Limitation

There exists no system developed by man that is perfect and complete. TBMS System too has some limitations on its functions. The limitations of the Ticket Booking Management System are stated below:

- There is no online payment system for registration.
- There is no agency rating system.
- Super Admin has to be added to the database manually.
- Passenger can't location agency pickup point in google map.
- Agency can't create sub account for their employees.
- The system is available for only for web.

7.2 Future work

Technology is ever-changing. To cope up with the change and the satisfaction of a user, a system needs to be kept constantly updated. To overcome the limitations of the system as well as to satisfy the need of its user, the following changes are planned to make in future versions of this the system:

- An online payment system will be added like bKash, PayPal.
- Agency rating system will be added.
- Passenger can location pickup point of agency.
- Agency will be able to create sub account.
- The system will be available for android so that all will be able to do their work separately.

7.3 Conclusion

The Ticket Booking management system is a computerized system focusing on managing Ticket Booking works. It is hoped that this system will work successfully. It will be helpful for passenger & agencies for book and manage ticket online. This system will be most useful for a Ticket Booking System.

Appendices

Model:

```
public class RegistrationModel
{
    private static RegistrationModel _instance;
    public static RegistrationModel GetInstance()
    {
        if (_instance == null)
        {
            _instance = new RegistrationModel();
        }
        return _instance;
    }

    public string RegId { get; set; }
    public string Name { get; set; }
    public string CompanyName { get; set; }
    public string Email { get; set; }
    public string ContactNo { get; set; }
    public string Gender { get; set; }
    public string DateofBirth { get; set; }
    public string Address { get; set; }
    public string Password { get; set; }
    public string Picture { get; set; }
    public string Status { get; set; }
    public string Type { get; set; }
    public string InTime { get; set; }
}

public class EventModel
{
    private static EventModel _instance;
    public static EventModel GetInstance()
    {
        if (_instance == null)
        {
            _instance = new EventModel();
        }
        return _instance;
    }

    public int EventId { get; set; }
    public string EventName { get; set; }
    public string Picture { get; set; }
    public int EventLocation { get; set; }
    public string EventAddress { get; set; }
    public double Fare { get; set; }
    public string StartTime { get; set; }
    public string EndTime { get; set; }
    public string SeatType { get; set; }
    public int SeatCapacity { get; set; }
    public string EventDate { get; set; }
    public int CompanyId { get; set; }
    public string Status { get; set; }
    public string Type { get; set; }
    public string InTime { get; set; } }
```

```

public class CoachModel
{
    private static CoachModel _instance;
    public static CoachModel GetInstance()
    {
        if (_instance == null)
        {
            _instance = new CoachModel();
        }
        return _instance;
    }

    public int CoachId { get; set; }
    public string CoachName { get; set; }
    public string CoachType { get; set; }
    public string CoachNo { get; set; }
    public int DistrictFrom { get; set; }
    public int DistrictTo { get; set; }
    public string StartingPoint { get; set; }
    public string EndPoint { get; set; }
    public string DepartureTime { get; set; }
    public string ArrivalTime { get; set; }
    public double TicketPrice { get; set; }
    public string Status { get; set; }
    public string CompanyId { get; set; }
    public string InTime { get; set; }
    public string SeatType { get; set; }
    public string CoachStatus { get; set; }
    public string SeatCapacity { get; set; }
}

public class BookTicketModal
{
    private static BookTicketModal _instance;
    public static BookTicketModal GetInstance()
    {
        if (_instance == null)
        {
            _instance = new BookTicketModal();
        }
        return _instance;
    }

    public int BookId { get; set; }
    public int CompanyId { get; set; }
    public int CoachId { get; set; }
    public int FromLocation { get; set; }
    public int ToLocation { get; set; }
    public string JourneyDate { get; set; }
    public double Fare { get; set; }
    public string SeatName { get; set; }
    public string CoachType { get; set; }
    public double SubTotal { get; set; }
    public double ServiceCharge { get; set; }
    public double Advance { get; set; }
    public double GrandTotal { get; set; }
    public string TokenId { get; set; }
    public string BkashNo { get; set; }
    public string TransactionNo { get; set; }
}

```

```

    public string Amount { get; set; }
    public string BookTime { get; set; }
    public string Status { get; set; }
    public string UserId { get; set; }
}

```

Gateway:

```

public class RegistrationGateway
{
    private MasterClass;
    private SqlConnection connection;
    private SqlCommand command;
    private static RegistrationGateway _instance;
    public static RegistrationGateway GetInstance()
    {
        if (_instance == null)
        {
            _instance = new RegistrationGateway();
        }
        return _instance;
    }
    public RegistrationGateway()
    {
        masterClass = MasterClass.GetInstance();
        connection = new SqlConnection(masterClass.Connection);
    }

    internal bool Save(RegistrationModel model)
    {
        bool result = false;
        SqlTransaction transaction = null;
        try
        {
            if (connection.State != ConnectionState.Open)
                connection.Open();
            transaction = connection.BeginTransaction();
            command = new SqlCommand("INSERT INTO
Registration(RegId,Name,CompanyName,Email,ContactNo,Gender,DateofBirth,Address>Password,Picture,Status,
Type,InTime)
VALUES(@RegId,@Name,@CompanyName,@Email,@ContactNo,@Gender,@DateofBirth,@Address,@Passw
ord,@Picture,@Status,@Type,@InTime)", connection);
            command.Parameters.AddWithValue("@RegId", model.RegId);
            command.Parameters.AddWithValue("@Name", model.Name);
            command.Parameters.AddWithValue("@CompanyName", model.CompanyName);
            command.Parameters.AddWithValue("@Email", model.Email);
            command.Parameters.AddWithValue("@ContactNo", model.ContactNo);
            command.Parameters.AddWithValue("@Gender", model.Gender);
            command.Parameters.AddWithValue("@DateofBirth", model.DateofBirth);
            command.Parameters.AddWithValue("@Address", model.Address);
            command.Parameters.AddWithValue("@Password", model.Password);
            command.Parameters.AddWithValue("@Picture", model.Picture);
            command.Parameters.AddWithValue("@Status", model.Status);
            command.Parameters.AddWithValue("@Type", model.Type);
            command.Parameters.AddWithValue("@InTime", model.InTime);

            command.Transaction = transaction;
            command.ExecuteNonQuery();

```

```

        transaction.Commit();
        result = true;
        if (connection.State != ConnectionState.Closed)
            connection.Close();
    }
    catch (Exception ex)
    {
        transaction.Rollback();
    }
    return result;
}

internal bool Update(RegistrationModel model)
{
    bool result = false;
    SqlTransaction transaction = null;
    try
    {
        if (connection.State != ConnectionState.Open)
            connection.Open();
        transaction = connection.BeginTransaction();
        command = new SqlCommand("UPDATE Registration SET Picture=@Picture WHERE
RegId=@RegId", connection);
        command.Parameters.AddWithValue("@Picture", model.Picture);
        command.Parameters.AddWithValue("@RegId", model.RegId);

        command.Transaction = transaction;
        command.ExecuteNonQuery();
        transaction.Commit();
        result = true;
        if (connection.State != ConnectionState.Closed)
            connection.Close();
    }
    catch (Exception)
    {
        transaction.Rollback();
    }
    return result;
}

internal bool UpdateStatus(RegistrationModel model)
{
    bool result = false;
    SqlTransaction transaction = null;
    try
    {
        if (connection.State != ConnectionState.Open)
            connection.Open();
        transaction = connection.BeginTransaction();
        command = new SqlCommand("UPDATE Registration SET Status=@Status WHERE RegId=@RegId",
connection);
        command.Parameters.AddWithValue("@Status", model.Status);
        command.Parameters.AddWithValue("@RegId", model.RegId);

        command.Transaction = transaction;
        command.ExecuteNonQuery();
        transaction.Commit();
        result = true;
        if (connection.State != ConnectionState.Closed)
            connection.Close();
    }

```

```

    }
    catch (Exception)
    {
        transaction.Rollback();
    }
    return result;
}
internal bool CreateAdmin(RegistrationModel model)
{
    bool result = false;
    SqlTransaction transaction = null;
    try
    {
        if (connection.State != ConnectionState.Open)
            connection.Open();
        transaction = connection.BeginTransaction();
        command = new SqlCommand("INSERT INTO
Registration(RegId,Name,Email,ContactNo,Gender,DateofBirth,Address>Password,Picture,Status,Type,InTime)
VALUES(@RegId,@Name,@Email,@ContactNo,@Gender,@DateofBirth,@Address,@Password,@Picture,@St
atus,@Type,@InTime)", connection);
        command.Parameters.AddWithValue("@RegId", model.RegId);
        command.Parameters.AddWithValue("@Name", model.Name);
        command.Parameters.AddWithValue("@Email", model.Email);
        command.Parameters.AddWithValue("@ContactNo", model.ContactNo);
        command.Parameters.AddWithValue("@Gender", model.Gender);
        command.Parameters.AddWithValue("@DateofBirth", model.DateofBirth);
        command.Parameters.AddWithValue("@Address", model.Address);
        command.Parameters.AddWithValue("@Password", model.Password);
        command.Parameters.AddWithValue("@Picture", model.Picture);
        command.Parameters.AddWithValue("@Status", model.Status);
        command.Parameters.AddWithValue("@Type", model.Type);
        command.Parameters.AddWithValue("@InTime", model.InTime);

        command.Transaction = transaction;
        command.ExecuteNonQuery();
        transaction.Commit();
        result = true;
        if (connection.State != ConnectionState.Closed)
            connection.Close();
    }
    catch (Exception)
    {
        transaction.Rollback();
    }
    return result;
}
}

public class EventGateway
{
    private MasterClass;
    private SqlConnection connection;
    private SqlCommand command;
    private static EventGateway _instance;
    public static EventGateway GetInstance()
    {
        if (_instance == null)
        {
            _instance = new EventGateway();

```

```

    }
    return _instance;
}
public EventGateway()
{
    masterClass = MasterClass.GetInstance();
    connection = new SqlConnection(masterClass.Connection);
}
internal bool InsertEvent(EventModel model)
{
    bool result = false;
    SqlTransaction transaction = null;
    try
    {
        if (connection.State != ConnectionState.Open)
            connection.Open();
        transaction = connection.BeginTransaction();
        command = new SqlCommand("INSERT INTO
EventInfo(EventName,EventAddress,EventLocation,StartTime,EndTime,EventDate,SeatType,SeatCapacity,Fare,
Picture,CompanyId,Status,InTime,Type)
VALUES(@EventName,@EventAddress,@EventLocation,@StartTime,@EndTime,@EventDate,@SeatType,@S
eatCapacity,@Fare,@Picture,@CompanyId,@Status,@InTime,@Type)", connection);
        command.Parameters.AddWithValue("@EventName", model.EventName);
        command.Parameters.AddWithValue("@EventAddress", model.EventAddress);
        command.Parameters.AddWithValue("@EventLocation", model.EventLocation);
        command.Parameters.AddWithValue("@StartTime", model.StartTime);
        command.Parameters.AddWithValue("@EndTime", model.EndTime);
        command.Parameters.AddWithValue("@EventDate", model.EventDate);
        command.Parameters.AddWithValue("@SeatType", model.SeatType);
        command.Parameters.AddWithValue("@SeatCapacity", model.SeatCapacity);
        command.Parameters.AddWithValue("@Fare", model.Fare);
        command.Parameters.AddWithValue("@Picture", model.Picture);
        command.Parameters.AddWithValue("@CompanyId", model.CompanyId);
        command.Parameters.AddWithValue("@Status", model.Status);
        command.Parameters.AddWithValue("@InTime", model.InTime);
        command.Parameters.AddWithValue("@Type", model.Type);

        command.Transaction = transaction;
        command.ExecuteNonQuery();
        transaction.Commit();
        result = true;
        if (connection.State != ConnectionState.Closed)
            connection.Close();
    }
    catch (Exception ex)
    {
        transaction.Rollback();
    }
    return result;
}

internal bool UpdateEventStatus(EventModel model)
{
    bool result = false;
    SqlTransaction transaction = null;
    try
    {
        if (connection.State != ConnectionState.Open)

```

```

        connection.Open();
        transaction = connection.BeginTransaction();
        command = new SqlCommand("UPDATE EventInfo SET Status=@Status WHERE
EventId=@EventId", connection);
        command.Parameters.AddWithValue("@Status", model.Status);
        command.Parameters.AddWithValue("@EventId", model.EventId);

        command.Transaction = transaction;
        command.ExecuteNonQuery();
        transaction.Commit();
        result = true;
        if (connection.State != ConnectionState.Closed)
            connection.Close();
    }
    catch (Exception)
    {
        transaction.Rollback();
    }
    return result;
}

internal bool UpdateEvent(EventModel model)
{
    bool result = false;
    SqlTransaction transaction = null;
    try
    {
        if (connection.State != ConnectionState.Open)
            connection.Open();
        transaction = connection.BeginTransaction();
        command = new SqlCommand("UPDATE EventInfo SET EventName=@EventName,
EventAddress=@EventAddress,EventLocation=@EventLocation,StartTime=@StartTime,EndTime=@EndTime,E
ventDate=@EventDate,SeatType=@SeatType,SeatCapacity=@SeatCapacity,Fare=@Fare,Picture=@Picture,Statu
s=@Status WHERE EventId=@EventId", connection);
        command.Parameters.AddWithValue("@EventAddress", model.EventAddress);
        command.Parameters.AddWithValue("@EventLocation", model.EventLocation);
        command.Parameters.AddWithValue("@StartTime", model.StartTime);
        command.Parameters.AddWithValue("@EndTime", model.EndTime);
        command.Parameters.AddWithValue("@EventDate", model.EventDate);
        command.Parameters.AddWithValue("@SeatType", model.SeatType);
        command.Parameters.AddWithValue("@SeatCapacity", model.SeatCapacity);
        command.Parameters.AddWithValue("@Fare", model.Fare);
        command.Parameters.AddWithValue("@Picture", model.Picture);
        command.Parameters.AddWithValue("@Status", model.Status);
        command.Parameters.AddWithValue("@EventName", model.EventName);
        command.Parameters.AddWithValue("@EventId", model.EventId);

        command.Transaction = transaction;
        command.ExecuteNonQuery();
        transaction.Commit();
        result = true;
        if (connection.State != ConnectionState.Closed)
            connection.Close();
    }
    catch (Exception ex)
    {
        transaction.Rollback();
    }
    return result;
}

```

```

    }
    }
public class CoachGateway
{
    private MasterClass;
    private SqlConnection connection;
    private SqlCommand command;
    private static CoachGateway _instance;
    public static CoachGateway GetInstance()
    {
        if (_instance == null)
        {
            _instance = new CoachGateway();
        }
        return _instance;
    }
    public CoachGateway()
    {
        masterClass = MasterClass.GetInstance();
        connection = new SqlConnection(masterClass.Connection);
    }
    internal bool AddBus(CoachModel model)
    {
        bool result = false;
        SqlTransaction transaction = null;
        try
        {
            if (connection.State != ConnectionState.Open)
                connection.Open();
            transaction = connection.BeginTransaction();
            command = new SqlCommand("INSERT INTO
CoachInfo(CoachName,CoachType,CoachNo,DistrictFrom,DistrictTo,StartingPoint,EndPoint,DepartureTime,Arri
valTime,TicketPrice,Status,CompanyId,InTime,SeatType,CoachStatus,SeatCapacity)
VALUES(@CoachName,@CoachType,@CoachNo,@DistrictFrom,@DistrictTo,@StartingPoint,@EndPoint,@D
epartureTime,@ArrivalTime,@TicketPrice,@Status,@CompanyId,@InTime,@SeatType,@CoachStatus,@SeatC
apacity)", connection);
            command.Parameters.AddWithValue("@CoachName", model.CoachName);
            command.Parameters.AddWithValue("@CoachType", model.CoachType);
            command.Parameters.AddWithValue("@CoachNo", model.CoachNo);
            command.Parameters.AddWithValue("@DistrictFrom", model.DistrictFrom);
            command.Parameters.AddWithValue("@DistrictTo", model.DistrictTo);
            command.Parameters.AddWithValue("@StartingPoint", model.StartingPoint);
            command.Parameters.AddWithValue("@EndPoint", model.EndPoint);
            command.Parameters.AddWithValue("@DepartureTime", model.DepartureTime);
            command.Parameters.AddWithValue("@ArrivalTime", model.ArrivalTime);
            command.Parameters.AddWithValue("@TicketPrice", model.TicketPrice);
            command.Parameters.AddWithValue("@Status", model.Status);
            command.Parameters.AddWithValue("@CompanyId", model.CompanyId);
            command.Parameters.AddWithValue("@InTime", model.InTime);
            command.Parameters.AddWithValue("@SeatType", model.SeatType);
            command.Parameters.AddWithValue("@CoachStatus", model.CoachStatus);
            command.Parameters.AddWithValue("@SeatCapacity", model.SeatCapacity);

            command.Transaction = transaction;
            command.ExecuteNonQuery();
            transaction.Commit();
            result = true;
            if (connection.State != ConnectionState.Closed)

```



```

        connection.Close();
    }
    catch (Exception ex)
    {
        string x=ex.Message;
        transaction.Rollback();
    }
    return result;
}
internal bool Delete(CoachModel model)
{
    bool result = false;
    SqlTransaction transaction = null;
    try
    {
        if (connection.State != ConnectionState.Open)
            connection.Open();
        transaction = connection.BeginTransaction();
        command = new SqlCommand("DELETE FROM CoachInfo WHERE CoachId=@CoachId",
connection);
        command.Parameters.AddWithValue("@CoachId", model.CoachId);

        command.Transaction = transaction;
        command.ExecuteNonQuery();
        transaction.Commit();
        result = true;
        if (connection.State != ConnectionState.Closed)
            connection.Close();
    }
    catch (Exception)
    {
        transaction.Rollback();
    }
    return result;
}
internal bool UpdateStatus(CoachModel coachModel)
{
    bool result = false;
    SqlTransaction transaction = null;
    try
    {
        if (connection.State != ConnectionState.Open)
            connection.Open();
        transaction = connection.BeginTransaction();
        command = new SqlCommand("UPDATE CoachInfo SET Status=@Status WHERE
CoachId=@CoachId", connection);
        command.Parameters.AddWithValue("@Status", coachModel.Status);
        command.Parameters.AddWithValue("@CoachId", coachModel.CoachId);

        command.Transaction = transaction;
        command.ExecuteNonQuery();
        transaction.Commit();
        result = true;
        if (connection.State != ConnectionState.Closed)
            connection.Close();
    }
    catch (Exception)
    {

```

```

        transaction.Rollback();
    }
    return result;
}
internal bool UpdateAir(CoachModel model)
{
    bool result = false;
    SqlTransaction transaction = null;
    try
    {
        if (connection.State != ConnectionState.Open)
            connection.Open();
        transaction = connection.BeginTransaction();
        command = new SqlCommand("UPDATE CoachInfo SET
CoachName=@CoachName,DistrictFrom=@DistrictFrom,DistrictTo=@DistrictTo,DepartureTime=@DepartureT
ime,ArrivalTime=@ArrivalTime,TicketPrice=@TicketPrice,SeatCapacity=@SeatCapacity WHERE
CoachId=@CoachId", connection);
        command.Parameters.AddWithValue("@CoachName", model.CoachName);
        command.Parameters.AddWithValue("@DistrictFrom", model.DistrictFrom);
        command.Parameters.AddWithValue("@DistrictTo", model.DistrictTo);
        command.Parameters.AddWithValue("@DepartureTime", model.DepartureTime);
        command.Parameters.AddWithValue("@ArrivalTime", model.ArrivalTime);
        command.Parameters.AddWithValue("@TicketPrice", model.TicketPrice);
        command.Parameters.AddWithValue("@SeatCapacity", model.SeatCapacity);
        command.Parameters.AddWithValue("@CoachId", model.CoachId);

        command.Transaction = transaction;
        command.ExecuteNonQuery();
        transaction.Commit();
        result = true;
        if (connection.State != ConnectionState.Closed)
            connection.Close();
    }
    catch (Exception)
    {
        transaction.Rollback();
    }
    return result;
}
}

public class BookTicketGateway
{
    private MasterClass;
    private SqlConnection connection;
    private SqlCommand command;
    private static BookTicketGateway _instance;
    public static BookTicketGateway GetInstance()
    {
        if (_instance == null)
        {
            _instance = new BookTicketGateway();
        }
        return _instance;
    }
    public BookTicketGateway()
    {
        masterClass = MasterClass.GetInstance();
        connection = new SqlConnection(masterClass.Connection);

```

```

}
internal bool BookTicket(BookTicketModal modal)
{
    bool result = false;
    SqlTransaction transaction = null;
    try
    {
        if (connection.State != ConnectionState.Open)
            connection.Open();
        transaction = connection.BeginTransaction();
        command = new SqlCommand("INSERT INTO
BookTicket(CompanyId,CoachId,FromLocation,ToLocation,JourneyDate,Fare,SeatName,CoachType,SubTotal,Se
rviceCharge,Advance,GrandTotal,TokenId,BkashNo,TransactionNo,Amount,BookTime,Status,UserId)
VALUES(@CompanyId,@CoachId,@FromLocation,@ToLocation,@JourneyDate,@Fare,@SeatName,@CoachT
ype,@SubTotal,@ServiceCharge,@Advance,@GrandTotal,@TokenId,@BkashNo,@TransactionNo,@Amount,@
BookTime,@Status,@UserId)", connection);
        command.Parameters.AddWithValue("@CompanyId", modal.CompanyId);
        command.Parameters.AddWithValue("@CoachId", modal.CoachId);
        command.Parameters.AddWithValue("@FromLocation", modal.FromLocation);
        command.Parameters.AddWithValue("@ToLocation", modal.ToLocation);
        command.Parameters.AddWithValue("@JourneyDate", modal.JourneyDate);
        command.Parameters.AddWithValue("@Fare", modal.Fare);
        command.Parameters.AddWithValue("@SeatName", modal.SeatName);
        command.Parameters.AddWithValue("@CoachType", modal.CoachType);
        command.Parameters.AddWithValue("@SubTotal", modal.SubTotal);
        command.Parameters.AddWithValue("@ServiceCharge", modal.ServiceCharge);
        command.Parameters.AddWithValue("@Advance", modal.Advance);
        command.Parameters.AddWithValue("@GrandTotal", modal.GrandTotal);
        command.Parameters.AddWithValue("@TokenId", modal.TokenId);
        command.Parameters.AddWithValue("@BkashNo", modal.BkashNo);
        command.Parameters.AddWithValue("@TransactionNo", modal.TransactionNo);
        command.Parameters.AddWithValue("@Amount", modal.Amount);
        command.Parameters.AddWithValue("@BookTime", modal.BookTime);
        command.Parameters.AddWithValue("@Status", modal.Status);
        command.Parameters.AddWithValue("@UserId", modal.UserId);

        command.Transaction = transaction;
        command.ExecuteNonQuery();
        transaction.Commit();
        result = true;
        if (connection.State != ConnectionState.Closed)
            connection.Close();
    }
    catch (Exception ex)
    {
        transaction.Rollback();
    }
    return result;
}
internal bool UpdateStatus(BookTicketModal modal)
{
    bool result = false;
    SqlTransaction transaction = null;
    try
    {
        if (connection.State != ConnectionState.Open)
            connection.Open();
        transaction = connection.BeginTransaction();

```

```

        command = new SqlCommand("UPDATE BookTicket SET Status=@Status WHERE
TokenId=@TokenId", connection);
        command.Parameters.AddWithValue("@Status", modal.Status);
        command.Parameters.AddWithValue("@TokenId", modal.TokenId);

        command.Transaction = transaction;
        command.ExecuteNonQuery();
        transaction.Commit();
        result = true;
        if (connection.State != ConnectionState.Closed)
            connection.Close();
    }
    catch (Exception)
    {
        transaction.Rollback();
    }
    return result;
}
internal bool UpdateNAC(BookTicketModal ob)
{
    bool result = false;
    SqlTransaction transaction = null;
    try
    {
        if (connection.State != ConnectionState.Open)
            connection.Open();
        transaction = connection.BeginTransaction();
        command = new SqlCommand("UPDATE BookTicket SET
BkashNo=@BkashNo, TransactionNo=@TransactionNo, Amount=@Amount WHERE TokenId=@TokenId",
connection);
        command.Parameters.AddWithValue("@BkashNo", ob.BkashNo);
        command.Parameters.AddWithValue("@TransactionNo", ob.TransactionNo);
        command.Parameters.AddWithValue("@Amount", ob.Amount);
        command.Parameters.AddWithValue("@TokenId", ob.TokenId);

        command.Transaction = transaction;
        command.ExecuteNonQuery();
        transaction.Commit();
        result = true;
        if (connection.State != ConnectionState.Closed)
            connection.Close();
    }
    catch (Exception ex)
    {
        transaction.Rollback();
    }
    return result;
}
}

```