CSE-180008,

Govt & UGC Apporved
Sonargaon University (SU)
we will rise up, we will shine
147/I, Green Road, Panthapath, Dhaka

ONLINE CINEPLEX RENT & TICKET MANAGEMENT SYSTEM

Submitted By

Md. Iqbal Hossain
ID: CSE1602008059

Md. Billal Hossain
ID: CSE1501004004

Aslam Hossain
ID: CSE1501004005

Md. Ilias Mia
ID: CSE1501004011

Md. Zuboraj
ID: CSE1501004006

Supervised By

Bakir Hossain
Lecturer
Department of Computer Science & Engineering
Sonargaon University

This project has been submitted to the Department of the Computer Science & Engineering at Sonargaon University in the partial fulfillment of the requirement for the degree of Bachelor of Science in Computer Science and Engineering.

September, 2018

DECLARATION

The project has been submitted to the Department of the Computer Science & Engineering, Sonargaon University in the partial fulfillment of the requirement for the degree of Bachelor of Science in CSE performed by us under supervision of **Bakir Hossain, Lecturer, Department of CSE** at Soanrgaon University. We hereby, declare that this project has not been submitted elsewhere for the requirement of any degree or diploma or any other purposes.

Supervised by:

Bakir Hossain
Lecturer
Department of CSE
Sonargaon University

Submitted by:

Md. Iqbal Hossain
ID: CSE1602008059

Md. Billal Hossain
ID: CSE1501004004

Aslam Hossain
ID: CSE1501004005

Md. Ilias Mia
ID: CSE1501004011

Md. Zuboraj
ID: CSE1501004006

Letter of Acceptance

This Project entitled **"Online Cineplex Rent and Ticket Management System"** submitted By Md. Iqbal Hossain ID: CSE1602008059, Md. Billal Hossain ID: CSE150100400, Aslam Hossain ID: CSE1501004005, Md. Ilias Mia ID: CSE1501004011, Md. Zuboraj ID: CSE1501004006 to the Department of Computer Science and Engineering, Sonargaon University, Dhaka, Bangladesh has been accepted as satisfactory for the partial fulfillment of the requirement for the degree of Bachelor of Science in Computer Science & Engineering and approved as to its style and contents.

Board of Examiners

Examiner 1

Examiner 2

Examiner 3

Examiner 4

Abstract

"Online Cineplex Rent and Ticket Management system" is the selling of cinema tickets and complex rental services over an electronic network, primarily the internet. These business transactions occur either as business-to-business, business-to-consumer. The terms e-ticketing and e-rental are often used interchangeably. The term e-ticketing is also sometimes used in reference to booking or buying processes for online ticketing.

This project provides the user with a category of different running movie tickets with various categories available for purchase in the store. In order to make the purchase happen, a ticketing cart is available to the user. The features that are provided by the website are Customers and Admins registration which means they can create account and log in, ticketing cart, movie searching, payment options and interacting facilities through email or message in mobile phone.

This system is based on a 3-tier approach, with a backend database, a middle tier of Apache web server and Object oriented PHP in Laravel Framework, and a web browser as the front end client.

The tools used for this project include PHP in Laravel Framework, HTML5, CSS and JavaScript as programming language and MySQL for backend database. The ER diagrams of the system and use case model were developed.

Thus, Online Cineplex Rent and Ticket Management system can be one of the best models for purchasing tickets and rental service online in the modern world. When a customer will visit our website he will get all the movies show time, arranged in systematic way. He will be able to see movie details as well as the purchasing process. In the menu bar there will be Home, Movies, Tickets, Rental and About. However, for the proper functionality he or she has to login. In the login page he will get sign in or sign up option. The customer can register in this site with email and basic information. After login the customer will get some extra feature and it is unique in our web application. In our web application the customer will get full access to check the status of the purchasing tickets and can be confirmed after getting auto generated online ticket.

ACKNOWLEDGEMENTS

It has been great pleasure for us to develop Web Application for **"Online Cineplex Rent and Ticket Management System"**. We have gathered sufficient knowledge and experience during this project.

It was a great pleasure to study and work with many gifted people who influenced us in many ways. First of all, we would like to thank our project Supervisor Bakir Hossain, Lecturer, Department of Computer Science and Engineering, Sonargaon University, who guided us to proper analysis of the system and helped to develop an elegant and efficient system.

He does not only gave us the great idea, but also encouraged us to seek out the clearest and deepest description of theoretical ideas as well as experimental findings. I am very grateful to him for his continuous support, advice and guidance.

Our special thanks goes to course coordinator and entire department teachers who helped us relentlessly and we want to thank our vice chancellor sir for his moral support.

We would like to thank our entire course mate in Sonargaon University, who took part in this discuss while completing the course work.

Finally, we would like to convey my distinctive thanks to my parents whom have always given me tremendous Support. Without their love and encouragement I would not had achieve this far.

TABLE OF CONTENTS

Title **Page No.**

Declaration..... i
Letter of Acceptance..... ii
Abstract..... iii
Acknowledgement..... iv

Chapter 01 Introduction (1-3)

1 Introduction..... 1
1.1 Motivation..... 1
1.2 Purpose of the project..... 2
1.3 Objective..... 2
1.4 Scope..... 2-3
1.5 Methodology and Tools..... 3

Chapter 02 Observation Existing System(4-6)

2.1 Observation on Existing Online Systems..... 4-6

Chapter 03 Requirement Analysis(7-10)

3.1 Requirement Analysis.....7
3.2 System Analysis.....7
3.3 Data Requirement.....7-8
3.4 Process Requirement.....8-9
3.5 Physical Design.....9
3.6 Software Requirements..... 9
3.7 Software Implementation.....10
3.7.1 HTML5.....10
3.7.2 PHP in Laravel Framework.....10
3.7.3 Apache Server (XAMPP).....10
3.7.4 Editor (Sublime Text).....10

Chapter 04 Design of the System(11-15)

4.1 Use Case Model..... 11-12
4.2 Use Case Model for Admin.....12
4.3 Use Case Model for Customer.....13
4.4 Sequence Diagram for Admin.....14
4.5 Sequence Diagram for User.....15

Chapter 05 Database Design(16-23)

5.1.1 Structure of the Database System.....16
5.1.2 Requirements of Data.....17
5.2 Entity Relationship Diagram..... 17
5.2.1 Entities.....18
5.2.2 Relationships.....18-19
5.2.3 Attributes.....19
5.3 Entity Relationship Diagram (ERD).....20-21
5.4 E-R Diagram of Online Cineplex Rent & Ticket Management System..... 22
5.5 Design of Database..... 23

Chapter 06 System Implementation(24-34)

6.1 User Account Login.....24
6.2 Admin Login.....25
6.3 Account Information.....25
6.4 Customer Details.....26
6.5 Customer Update Profile.....26
6.6 Ticket Details.....27
6.7 Running Movies.....28
6.8 Payment Method.....28
6.9 Movie List.....29
6.10 Admin Panel Home.....29
6.11 Posting a New Movie.....31
6.15 Movie Post Edit Option.....32
6.16 Rental Service.....34

Chapter 07 Security of the System

7.1 Security for User 35
7.2 Security for Admin.....35
7.3 Secured with Session Cookie.....35
7.4 Security of Database 36-37

Chapter 08 Testing

8.1 Importance of Software Testing.....38
8.2 Types of Software Testing.....39
8.2.1 Process.....39
8.2.2 All Life Cycle Activities.....39
8.2.3 Static Testing.....40
8.2.4 Dynamic Testing.....40
8.2.5 Planning.....40
8.2.6 Preparation.....40
8.2.7 Evaluation.....40
8.2.8 Software products and related work products.....40

Chapter 09 Conclusion

9.1 Conclusion & Future Scope.....41-42

References: 43

Appendix: 44-64

List of Figures:

Figure 1: Star Cineplex Ticket Pricing.....5

Figure 2: Movie details from www.cineplexbd.com 5

Figure 3: Specific ticketing and other options.....6

Figure 4: Use Case Model for Admin.....12

Figure 5: Use Case Model for Admin.....13

Figure 6: Sequence Diagram for Admin.....14

Figure 7: Sequence Diagram for User.....15

Figure 8: One-to-One relationship.....20

Figure 9: One-to-Many relationship.....20

Figure 10: Many-to-Many relationship.....21

Figure 11: Entity Relationship diagram.....21

Figure 11: shows the Entity Relationship Diagram of Online Cineplex Rent & Ticket Management System.....22

Figure 12: shows the database design of Online Cineplex Rent And Ticket Management System.....23

Figure 13: User Account Login.....24

Figure 14: Admin login.....25

Figure 15: Account Information.....25

Figure 16: Customer Details.....26

Figure 17: Update Profile.....27

Figure 18: Product details.....27

Figure 19: Running Movies.....28

Figure 20: Payment method.....28

Figure 21: Movie List.....29

Figure 22: Admin Panel log in box.....30

Figure 23: Admin Panel log in page.....30

Figure 24: Post New Movie.....31

Figure 25 : Create New Post.....31

Figure 26: View movies archive.....32

Figure 27: Edit post option.....32

Figure 28: Successfully saved the post.....33

Figure 29: View movie list with edit option.....33

Figure 30: rental details.....34

Chapter 1

Introduction

To develop an Online Cineplex Rent and Ticket Management System which will provide buying and selling of movie tickets and complex rental service through the use of internet is the main target of this project. Moreover, the system has been developed in such a way which makes it reliable, secure and user friendly. Online Cineplex Rent and Ticket Management System is a system which keeps a list of Running and Upcoming movies and can do operations on ticket booking or selling. The another major operation is Purchase in which all the billing details, transactions and stock purchasing details are involved in it. An auto generated ticket will be given to the users.

Online Cineplex Rent and Ticket Management System is the service by which users can directly buy movie tickets and confirm rental service for various purposes over the internet. So it is the process of buying for the customer and selling for the merchant or organization who sell on the internet. Since the advancement of technology, the customers are becoming more and more eager to buy necessary things sitting at home and just using the internet and computer rather than actually go to the market physically. That's why because of the emergence of the World Wide Web, merchants have sought to sell their products to people who surf the internet. Thus people can buy just about anything from online that provide their needs with the help of internet and computer.

Motivation:

In the existing system, a client has to visit every cinema theater to buy a ticket physically. He faces difficulties to collect a certain movie ticket. Sometimes people has to stand in a long line to get the expected ticket. In that situation chaos among people is very common scenario in our country. Moreover, standing in a line is quite time consuming and need physical strength while waiting for purchasing movie tickets in front of the Cineplex. So the key points are: 1. Very time consuming, 2. Physical strength consuming, 3. Ticket may not be available when you reach to the counter, 4. Lack of personal rental service for various purpose.

1.2 Purpose of the Project

The system is willing to replace the existing system for the buying and selling ticket process which is not efficient, lengthy process and time consuming. This system is more user friendly than existing system that is why the consumers can easily access the system and can look for the desired movie tickets they are desiring to buy. The administrators can easily access their information. The main purpose of the system will be efficient buying, selling and maintaining a secure transaction process between the merchants and the customers. So the key points are:

- Improve customer service by providing buy and sell tickets through internet.
- Reduce certain costs such as –travel cost, ordering costs and stock out costs.
- Invoice system and easy reporting process for the administrator. Administrator can easily manage his store and tickets and category through online.

1.3 Objective

The Online Cineplex Rent and Ticket Management System is web based application where the customer registers with the website and finds the desired movie ticket they are willing to buy. And after choosing the movie, it is added to a virtual ticketing cart until finished. So it is much easier, time and cost effective process with a simple click.

1.4 Scope

This system provides the services from the admin to the customer with all necessary steps. The users select their movie from the entire list with details. It provides the users with all the necessary information about the movie including the price with details. Following are the scopes of the developed system that are elaborately discussed:

- **For Admin:**
    - **Manage user's info:** This module captures all the personal information of a customer. Admin can also edit, active and delete all the users for necessity purpose.
    - **Insert Information:** With this system Admin can insert their ticket or rental service information with details to be available to the Users.

- **For User:**

  - **Insert Detail:** A user can register and can insert their personal information for further activity confirmation in the application.

  - **View Movie and Ticket Information:** This system shows that the Users can see some basic information of movies without going for full details.

1.5 Methodology and Tools

The system is implanted with a 3-tier approach, with a backend MYSQL database, a middle tier of PHP in Laravel Framework, and a web browser as the front end client. System development life cycle (SDLC) model has been used for the software development. The Software Requirements Analysis Process covers the complex task of documenting the requirements of all these users, modeling and analyzing these requirements and documenting them as a basis for system design.

- Language Used: HTML, XHTML
- Scripts Used: PHP in Laravel Framework, CSS, Java Script.
- Designs: Photoshop.
- Database Used: MYSQL
- Server: Apache Server
- Editor: Sublime Text

Chapter 2
Observation on Existing System

There may be different online ticketing based websites in the net, but the point is, they don't have the same features. The number of people preferring ticket purchase via online is in the increase rate. Although, security is one of the main reason people doesn't want to buy tickets online. But the security of online ticketing is increasing now a days. So if security can be maintained, then people will gain more and more confidence in using online ticketing technology. But many of the online ticketing sites may not have the expected security and privacy from the user.

Another important issue in online ticketing sector is the interactive cycle between a user and a website. If the website doesn't responds to the command entered by the user, then it is not complete. So these factors should be a must for the upcoming and growing e-commerce websites.

2.1 Observation on Existing Online Systems

We have surfed through some of the e-ticketing based websites in Bangladesh and we observed those and compared the features with the system we proposed. It is to say that most of the website doesn't cover full features of all the e-ticketing features. We are giving some screenshots of the websites we observed to get this more clear.

www.cineplexbd.com : Ticket pricing from cineplexbd.com.

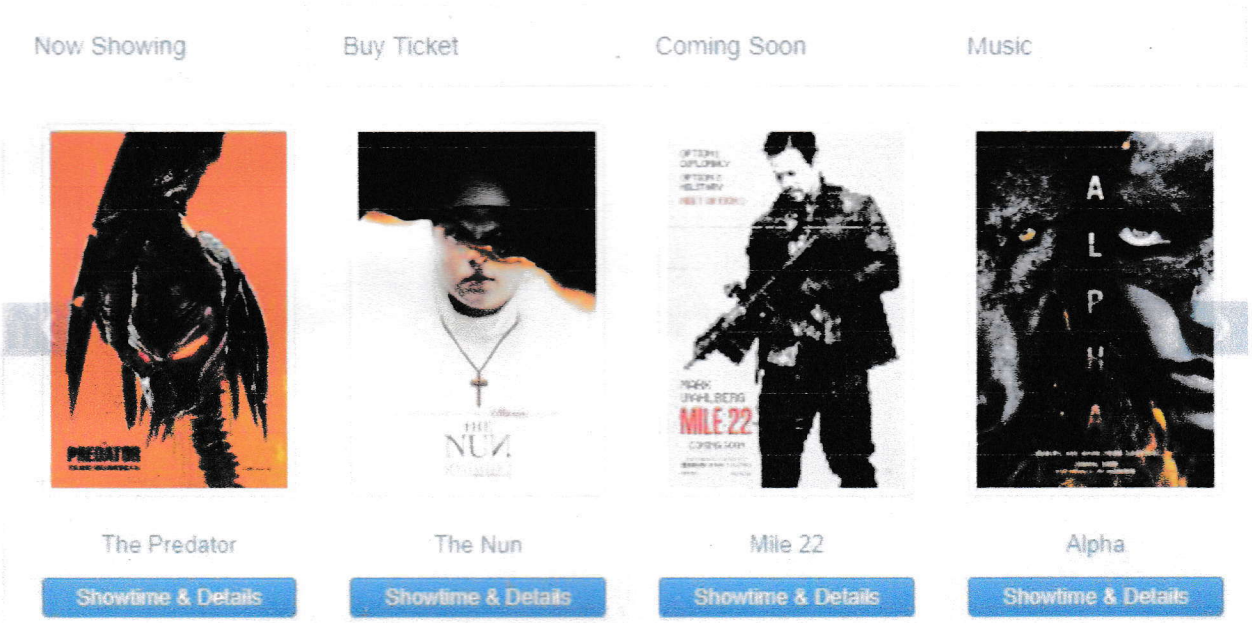


Figure 1: Star Cineplex Ticket Pricing

Figure 2 shows a movie details from www.cineplexbd.com

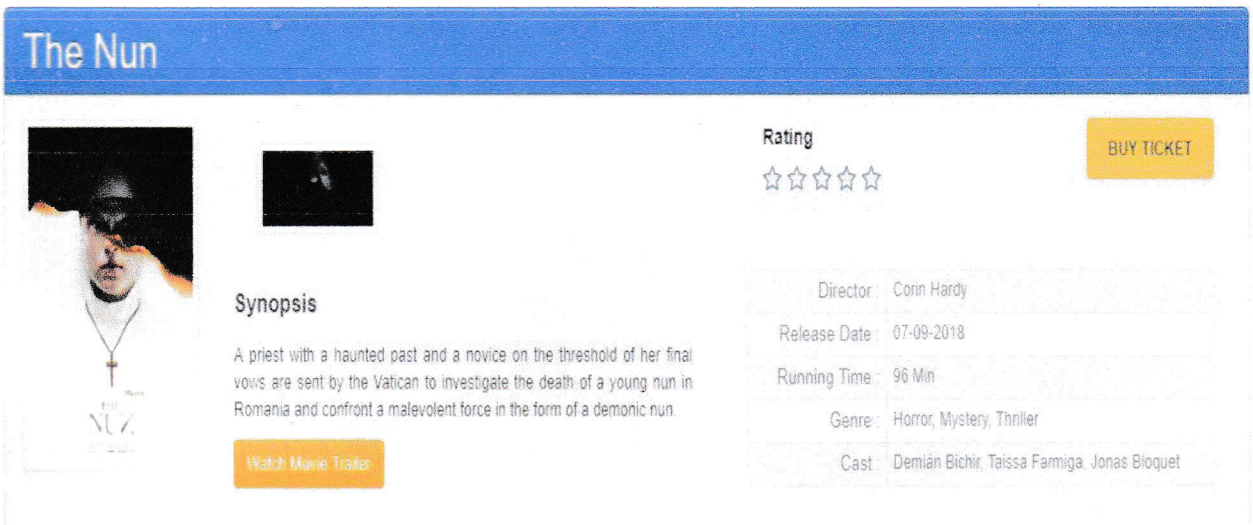


Figure 2: Movie details from www.cineplexbd.com

- **Features which are unavailable here:**

  - The first thing to talk about is that there is no option to compare between different Cineplex's ticket pricing. This feature is needed to compare ticket prices which are similar. It will help the users to get the best ticketing service.

  - No payment option integrated to buy the tickets via mobile air credit.

❖ **www.blockbasterbd.com:** Various options of e-ticketing and service.



Figure 3: Specific ticketing and other options

- **Features which are unavailable here:**

This is a top charted website and users can view, search as well as buy tickets. But these don't cover the complete features of an e-ticketing and rental service website.

Chapter 3
Requirement Analysis

Requirements analysis, also called requirements engineering, is the process of determining user expectations for a new or modified product. These features, called requirements, must be quantifiable, relevant and detailed. In software engineering, such requirements are often called functional specifications.

It is the process that covers what the software is expected to do and how it behaves. So it first collects what the expectations of the users are and then it reflects them in the system.

3.1 Requirement Analysis

This Online Cineplex Rent and Ticket Management system required the following requirements. This has mainly two actors. Those are Admin and User. In an E-Ticketing service Business holder is known as Admin and Customer is known as the User.

3.2 System Analysis

System Analysis is the act, process, or profession of studying an activity such as a procedure, a business, or a physiological function typically by mathematical means in order to define its goals or purposes and to discover operations and procedures for accomplishing them most efficiently.

During this process, the present system is observed and data are collected from the available files. So, system observation is the main tool used for system analysis.

3.3 Data Requirement

During requirement analysis the following data have been identified for an e-ticketing system:

- At first each person need to register himself/herself as a User or an Admin for accessing the necessary information. They also have a user id and password. They can login to the system from the web using their id and password.

- Admin need to login to the system to operate the system. Admin has an individual or unique login user id and password. Through this id and password admin can login to the system.
- A User can search and browse for the movies he is willing to watch.
- Admin can update all the information of the registered items. Any registered items can be deleted by the Admin. And also can view all the details of the Users.
- Admin can update the category list of the movies. An admin can edit or delete a category from the category list. Admin can also insert a new category menu in the category list. Admin can also insert movies with price and details.
- Users can choose and buy their expected movie tickets from the system and can confirm the order by choosing the payment method.
- There is a virtual ticketing cart available in the system in it until the shopping is complete.
- After completing the purchase, the users can choose the payment method which is via credit card, bkash, Duth-Bangla Mobile Banking.
- There will be payment method in this website which is very new that is an user can make payment via mobile air credit in near future.

3.4 Process Requirement

The following process requirements are identified for job system:

- A valid login is required for all process to be performed. A valid login is required for every registered users and admin. All of them have a valid user id and password. System will authenticate their valid login.
- After valid login Admin and User can check his/her information and can see personal information.

- Admin can login to the system. Admin can view, delete and update all members' information. Admin can also enter new category in the list and insert new products.

3.5 Physical Design

The physical design relates to the actual input and output processes of the system. This is laid down in terms of how data is input into a system, how it is verified or authenticated, how it is processed, and how it is displayed as In Physical design; the following requirements about the system are decided.

- Input requirements
- Output requirements
- Storage requirements
- Processing Requirements

Put another way, the physical portion of systems design can generally be broken down into three subtasks-

- User Interface Design
- Data Design
- Process Design

3.6 Software Requirements

For installing the system, we mainly required the following software, installed on the hosting system.

➢ **Web Server**
- Apache (Xampp Apache/3.2.2 (Win64))

➢ **Database Language:**
- MySQL

➢ **Server Side Language**
- PHP in Laravel Framework

➢ **Visual Design**
- Sublime Text

3.7 Software Implementation

To implement this software the tools are used fully open sources. So that there are no costing involves developing this software. For designing this project HTML, CSS, JAVASCRIPT are used which is open source. Apache web server is used as web server. PHP is used for database coding. MySQL is used as database server.

3.7.1 HTML5

HTML5 is short form of **Hyper Text Markup Language** and it is a language that are used to create electronic documents, especially pages on the World Wide Web that contain connections called hyperlinks to other pages. Every web page you see on the Internet, including this one contains HTML code that helps format and show text and images in an easy to read format. Without HTML a browser would not know how to format a page and would only display plain text with no formatting that contained no links.

3.7.2 PHP in Laravel Framework

PHP is probably the most popular scripting language on the web. It is used to enhance web pages. With PHP in Laravel Framework, we can do things like create username and password login pages, check details from a form, create forums, picture galleries, surveys, and a whole lot more. If we have come across a web page that ends in PHP, then the author has written some programming code to liven up the plain, old HTML. PHP is known as a server-sided language. That's because the PHP doesn't get executed on our computer. However, on the computer we request the page. The results are then handed over to us, and displayed in our browser.

3.7.3 Apache Server (XAMPP)

The Apache HTTP Server, commonly referred to as Apache is a web server application notable for playing a key role in the initial growth of the World Wide Web. Apache is developed and maintained by an open community of developers under the auspices of the Apache Software Foundation. Most commonly used on a Unix-like system, the software is available for a wide variety of operation systems, including UNIX, FreeBSD, Linux, Solaris, Novell NetWare, OSX, Microsoft Windows, OS/2, TPF, OpenVMS and e-ComStation. Released under the Apache License, Apache is open-source software.

3.7.4 Editor (Sublime Text)

Sublime Text is the web development tool that lets you efficiently design, develop and maintain standards-based websites and applications. Sublime text provides a powerful combination of visual layout tools, application development features, and code editing support.

Chapter 4
Design of the System

System design is the process of defining the architecture, modules, interfaces, and data for a system to satisfy specified requirements. Systems design could be seen as the application of systems theory to product development. There is some overlap with the disciplines of systems analysis, systems architecture and systems engineering.

It implies a systematic approach to the design of a system. It may take a bottom-up or top-down approach, but either way the process is systematic wherein it takes into account all related variables of the system that needs to be created—from the architecture, to the required hardware and software, right down to the data and how it travels and transforms throughout its travel through the system. Systems design then overlaps with systems analysis, systems engineering and systems architecture.

The systems design approach first appeared right before World War II, when engineers were trying to solve complex control and communications problems. They needed to be able to standardize their work into a formal discipline with proper methods, especially for new fields like information theory, operations research and computer science in general.

4.1 Use Case Model

A use case diagram is a graphic depiction of the interactions among the elements of a system.

It is a methodology used in system analysis to identify, clarify, and organize system requirements. In this context, the term "system" refers to something being developed or operated, such as a mail-order product sales and service Web site. Use case diagrams are employed in UML (Unified Modeling Language), a standard notation for the modeling of real-world objects and systems.

System objectives can include planning overall requirements, validating a hardware design, testing and debugging a software product under development, creating an online help reference, or performing a consumer-service-oriented task. For example, use cases in a product sale environment would include item ordering, catalog updating, payment processing, and customer relations. A use case diagram contains four components.

- The boundary, which defines the system of interest in relation to the world around it.
- The actors, usually individuals involved with the system defined according to their roles.
- The use cases, which the specific roles are played by the actors within and around the system.
- The relationships between and among the actors and the use cases.

4.2 Use Case Model for Admin: A **use case diagram** is a graphic depiction of the interactions among the elements of a system Figure 4 shows the Use case diagram of admin where the admin can login, add/delete customer, add products, delete/update product, add/update category, logout.

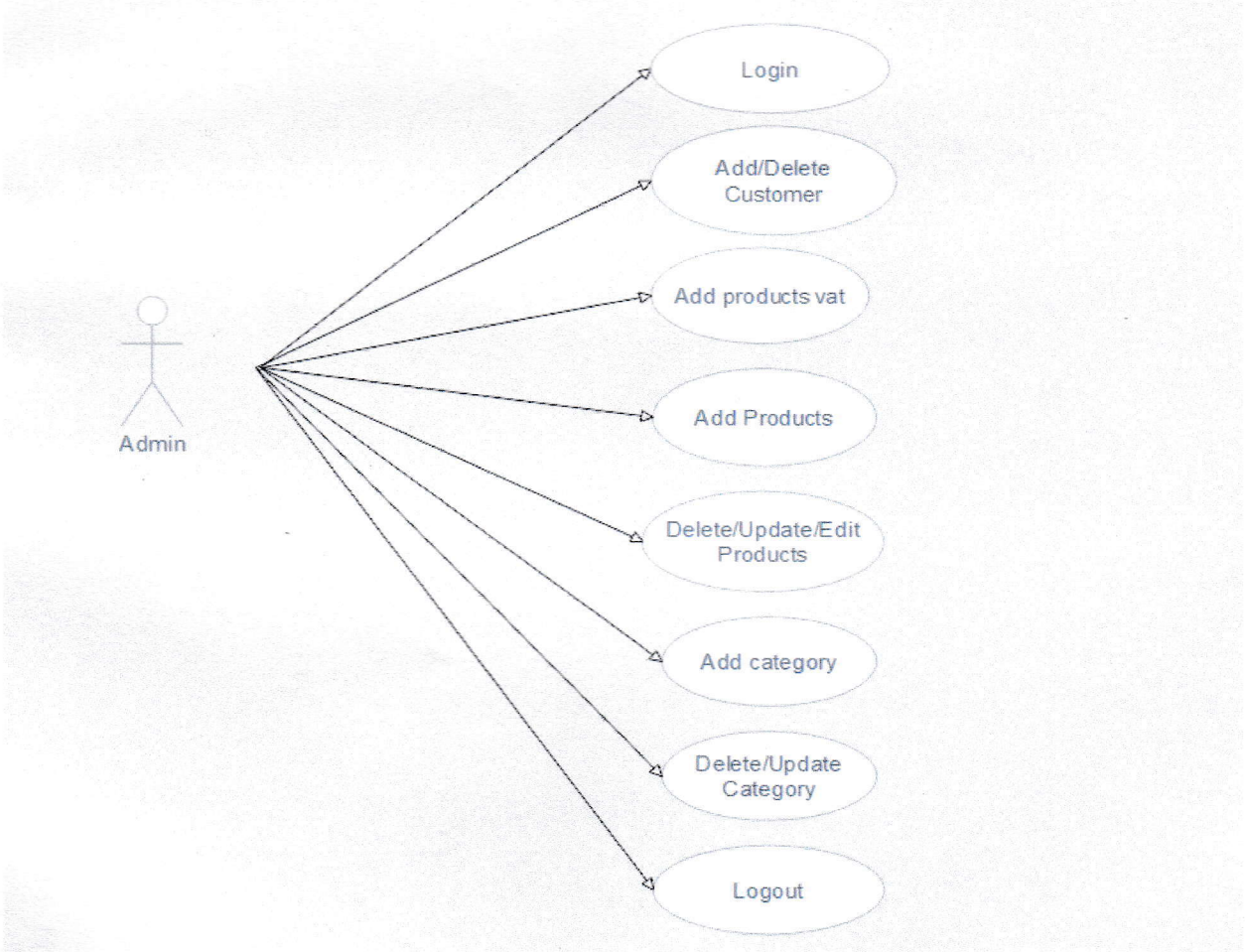


Figure 4: Use Case Model for Admin

4.3 Use Case Model for Customer: Figure 5 shows the Use case diagram of Customer where the customer can login, register, view products, compare products, place order, payment, wish list, update profile.

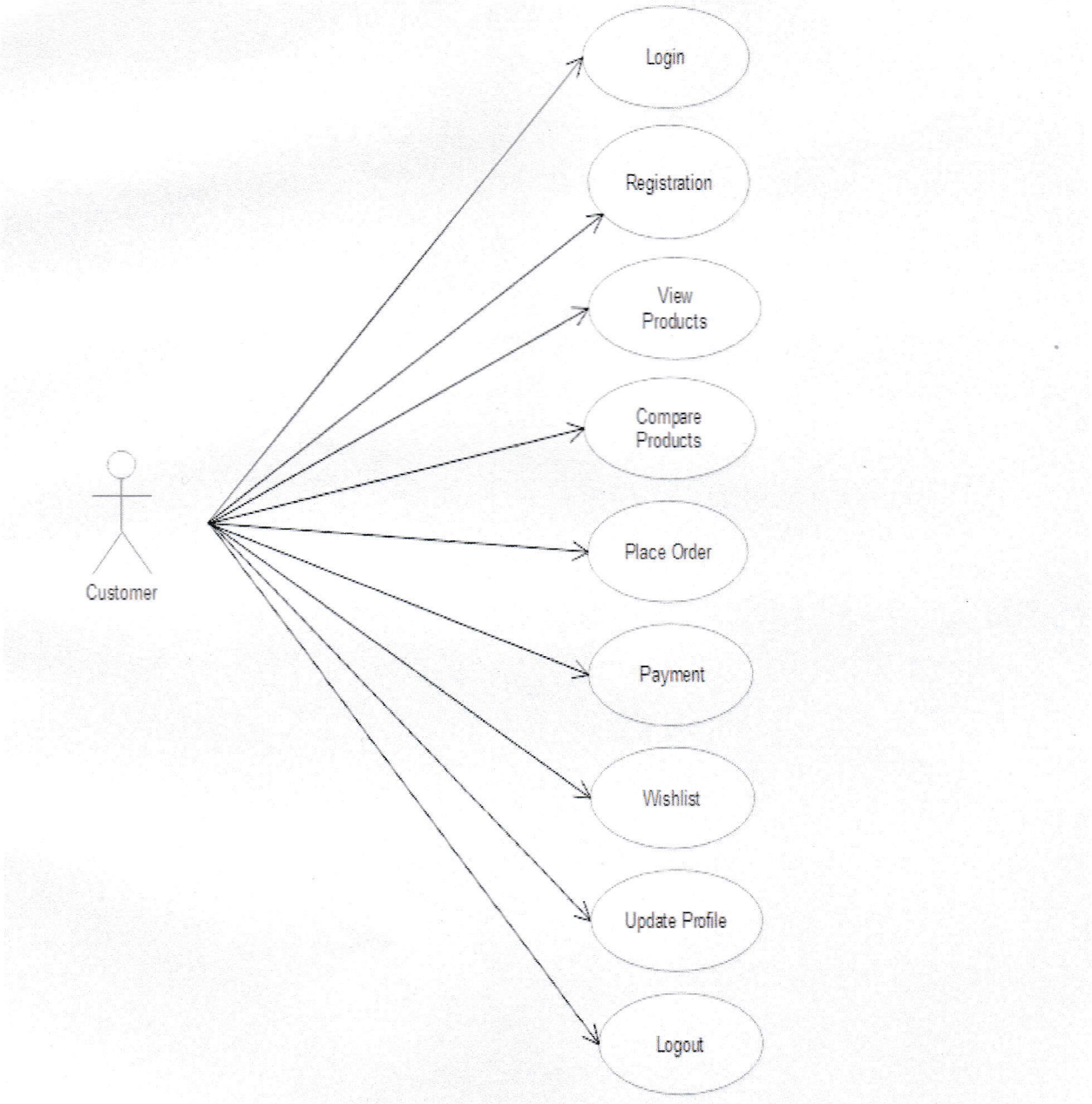


Figure 5: Use Case Model for Admin

4.4 Sequence Diagram for Admin: A **sequence diagram** is an interaction **diagram** that shows how objects operate with one another and in what order. It is a construct of a message **sequence** chart. A **sequence diagram** shows object interactions arranged in time **sequence** Figure 6 shows the sequence diagram of admin

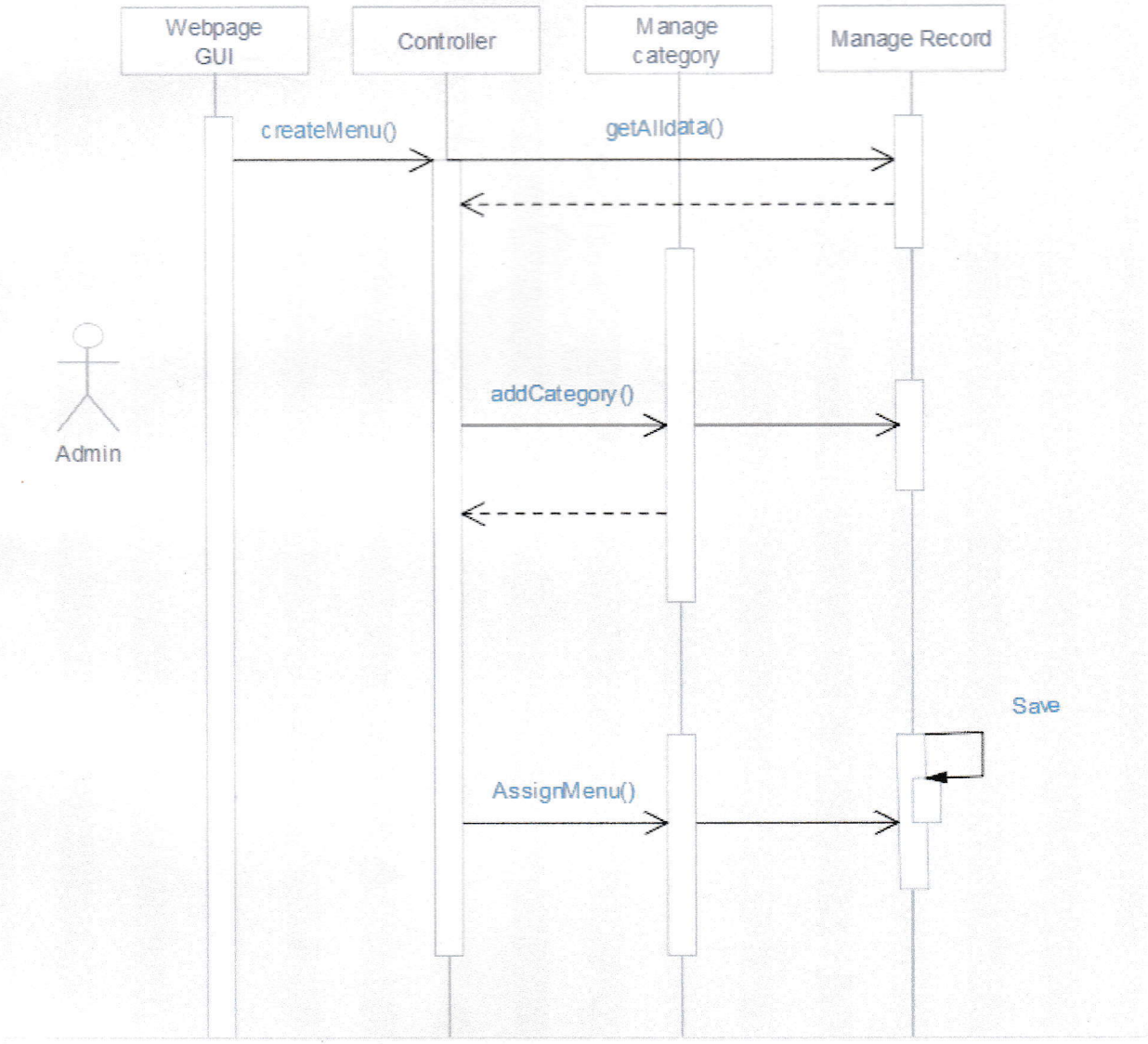


Figure 6: Sequence Diagram for Admin

4.5 Sequence Diagram for User: Figure 7 shows the sequence diagram of User

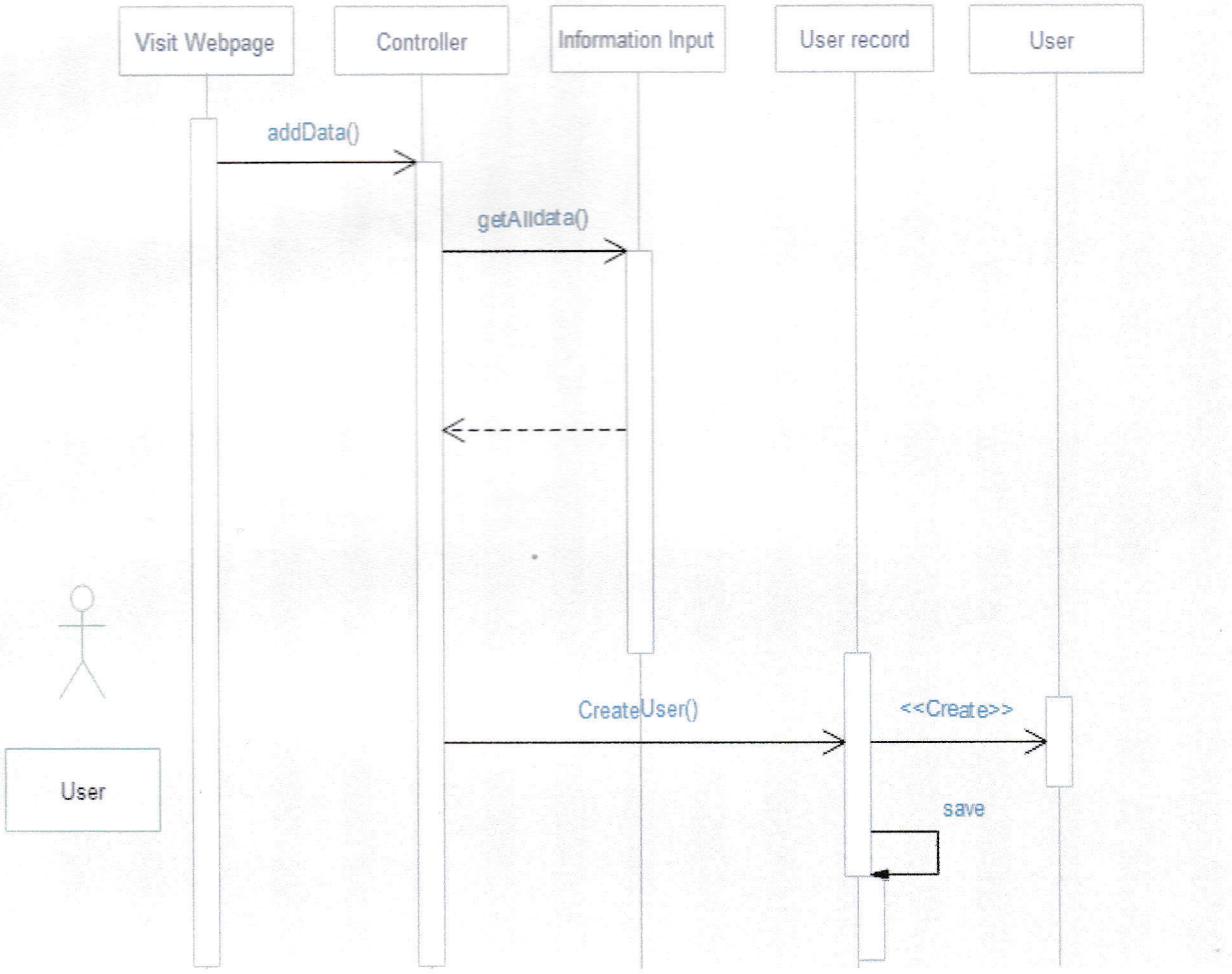


Figure 7: Sequence Diagram for User

Chapter 5
Database Design

Database design is the process of producing a detailed data model of database. This data model contains all the needed logical and physical design choices and physical storage parameters needed to generate a design in a data definition language, which can then be used to create a database. A fully attributed data model contains detailed attributes for each entity.

It is a collection of information that is organized so that it can be easily accessed, managed and updated.

Data is organized into rows, columns and tables, and it is indexed to make it easier to find relevant information. Data gets updated, expanded and deleted as new information is added. Databases process workloads to create and update themselves, querying the data they contain and running applications against it.

Computer databases typically contain aggregations of data records or files, such as sales transactions, product catalogs and inventories, and customer profiles.

Typically, a database manager provides users with the ability to control read/write access, specify report generation and analyze usage. Some databases offer ACID (atomicity, consistency, isolation and durability) compliance to guarantee that data is consistent and that transactions are complete.

Databases are prevalent in large mainframe systems, but are also present in smaller distributed workstations and midrange systems, such as IBM's AS/400 and personal computers.

5.1.1 Structure of the Database System

The database structure of a database system is its structure described in a formal language supported by the database management system (DBMS). The term "schema" refers to the organization of data as a blueprint of how the database is constructed (divided into database tables in the case of relational databases). The formal definition of a database schema is a set of formulas (sentences) called integrity constraints imposed on a database. These integrity constraints ensure compatibility between parts of the schema. All constraints are expressible in the same language. A database can be considered a structure in realization of the database language. The states of a created conceptual schema are transformed into an explicit mapping, the database schema. This describes how real-world entities are modeled in the database.

5.1.2 Requirements of Data

Data requirements are prescribed directives or consensual agreements that define the content and/or structure that constitute high quality data instances and values. Data requirements can thereby be stated by several different individuals or groups of individuals.

It is required to maintain system data or files. It provides a brief description of the required data and files that the system must need to run its functions. It can be separated into two categories: static and dynamic. The first step in gathering system data is to determine what data is required for building the model. This should be dictated primarily by the scope and level of detail required to achieve the model objectives as described earlier. It is best to go from general to specific in gathering system data. The initial focus should be on defining the overall process flow to provide a skeletal framework for attaching more detailed information. Detailed information can then be added gradually as it becomes available (e.g., resource requirements, processing times, etc.). Starting with the overall process flow not only provides an orderly approach to data gathering, but also enables the model building process to get started which reduces the amount of time to build and debug the model later. Often, missing data becomes more apparent as the model is being built.

In defining the basic flow of entities through the system, a flow diagram can be useful as a way of documenting and visualizing the physical flow of entities from location to location. Once a flow diagram is made, a structured walk-through can be conducted with those familiar with the operation to ensure that the flow is correct and that nothing has been overlooked. The next step might be to define the detail of how entities move between locations and what resources are used for performing operations at each location. At this point it is appropriate to identify location capacities, move times, processing times, etc.

5.2 Entity Relationship Diagram

Entity Relationship Diagram shortly known as ERD shows the relationships among entities. ERD graphically illustrates an information system's entities and the relationships between those entities. An ERD is a conceptual and representational model of data used to represent the entity framework infrastructure.

The elements of an ERD are:

- Entities
- Relationships
- Attributes

5.2.1 Entities

An entity is any singular, identifiable and separate object. It refers to individuals, organizations, and systems, bits of data or even distinct system components that are considered significant in and of them.

The term is used in a number of programming languages/concepts, database management, systems design and other arenas.

There are 3 types of entities:

- **Strong Entities** are the entities which can be fully identified by its own attributes and it is independent from other entity types.
- **Weak Entities** are the entity which can't be fully identified by its own attributes and takes the foreign key as an attribute (generally it takes the primary key of the entity it is related to) in conjunction.
- **Associative Entities** are used in relational and entity–relationship theory. A relational database requires the implementation of a base relation (or base table) to resolve many-to-many relationships. This kind of base relation is called an associative table.

5.2.2 Relationships

A relationship, in the context of databases, is a situation that exists between two relational database tables when one table has a foreign key that references the primary key of the other table. Relationships allow relational databases to split and store data in different tables, while linking disparate data items.

There are three types of relationships:

- **One-to-one:** Both tables can have only one record on either side of the relationship. Each primary key value relates to only one (or no) record in the related table. They're like spouses—you may or may not be married, but if you are, both you and your spouse have only one spouse. Most one-to-one relationships are forced by business rules and don't flow naturally from the data. In the absence of such a rule, you can usually combine both tables into one table without breaking any normalization rules.
- **One-to-many:** The primary key table contains only one record that relates to none, one, or many records in the related table. This relationship is similar to the one between you and a parent. You have only one mother, but your mother may have several children.

- **Many-to-many:** Each record in both tables can relate to any number of records (or no records) in the other table. For instance, if you have several siblings, so do your siblings (have many siblings). Many-to-many relationships require a third table, known as an associate or linking table, because relational systems can't directly accommodate the relationship.

5.2.3 Attributes

A database attribute is simply a column name in a table in a database.

A database is comprised of tables. Each table has columns and rows; the columns in a database are called database attributes.

This makes sense when you think of it, given that the non-technical definition of an attribute is that it defines a characteristic or quality of something.

In computing, an attribute is a specification that defines a property of an object, element, or file. It may also refer to or set the specific value for a given instance of such. For clarity, attributes should more correctly be considered metadata. An attribute is frequently and generally a property of a property. However, in actual usage, the term attribute can and is often treated as equivalent to a property depending on the technology being discussed. An attribute of an object usually consists of a name and a value; of an element, a type or class name; of a file, a name and extension.

Single valued Attributes: An attribute, that has a single value for a particular entity is known as single valued attributes. For example, age of an employee entity.

Multi valued Attributes: An attributes that may have multiple values for the same entity is known as multi valued attributes. For example colors of a car entity.

Compound Attribute/Composite Attribute: Attribute can be subdivided into two or more other Attribute. For Example, Name can be divided into First name, Middle name and Last name.

Simple Attributes/Atomic Attributes: The attributes which cannot be divided into smaller subparts are called simple or atomic attributes. For example, age of employee entity

Stored Attribute: An attribute, which cannot be derived from other attribute, is known as stored attribute. For example: Birthdate of employee.

Derived Attribute: Attributes derived from other stored attribute. For example age from Date of Birth and Today's date.

5.3 Entity Relationship Diagram (ERD)

An *entity relationship model*, also called an *entity-relationship (ER) diagram*, is a graphical representation of entities and their relationships to each other, typically used in computing in regard to the organization of data within databases or information systems. An entity is a piece of data-an object or concept about which data is stored.

Relationships between Entities

A relationship is how the data is shared between entities. There are three types of relationships between entities:

1. One-to-One

One instance of an entity (A) is associated with one other instance of another entity (B). For example, figure 8 shows in a database of employees, each employee name (A) is associated with only one social security number (B).

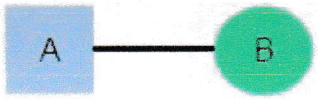


Figure 8: One-to-One relationship

2. One-to-Many

One instance of an entity (A) is associated with zero, one or many instances of another entity (B), but for one instance of entity B there is only one instance of entity A. For example, figure 9 shows for a company with all employees working in one building, the building name (A) is associated with many different employees (B), but those employees all share the same singular association with entity A.

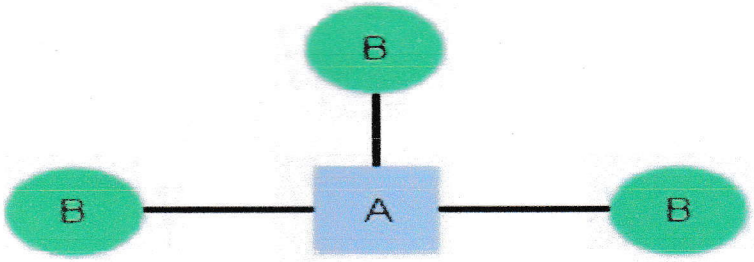


Figure 9: One-to-Many relationship

3. Many-to-Many

One instance of an entity (A) is associated with one, zero or many instances of another entity (B), and one instance of entity B is associated with one, zero or many instances of entity A. For example, figure 10 shows for a company in which all of its employees work on multiple projects, each instance of an employee (A) is associated with many instances of a project (B), and at the same time, each instance of a project (B) has multiple employees (A) associated with it.

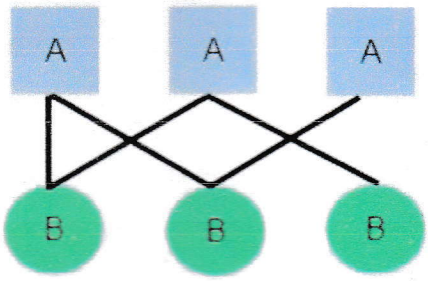


Figure 10: Many-to-Many relationship

Here are the entity relationship diagram symbols which are used to draw an Entity Relationship Diagram.

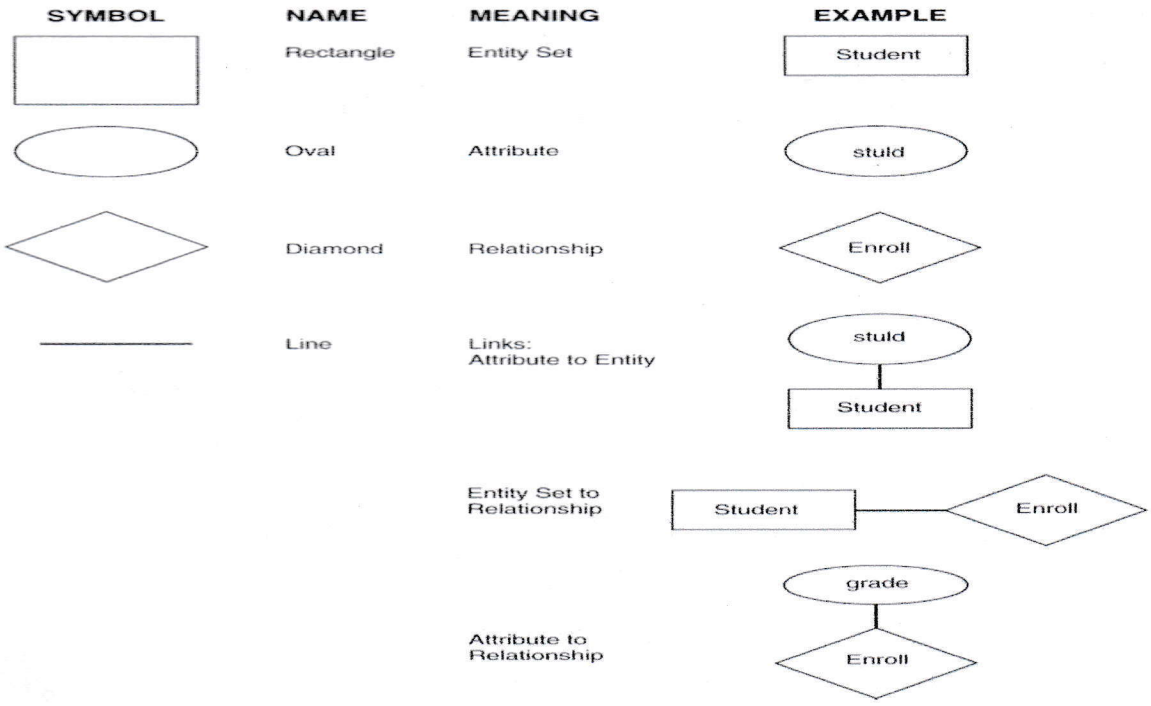


Figure 11: Entity relationship diagram symbols

5.4 Entity Relationship Diagram of Online Cineplex Rent & Ticket Management System: An entity relationship model, also called an entity-relationship (**ER**) diagram, is a graphical representation of entities and their relationships to each other, typically used in computing in regard to the organization of data within databases or information systems. Figure 12 shows the Entity Relationship Diagram of Online Marketplace.



Figure 12: shows the Entity Relationship Diagram of Online Cineplex Rent & Ticket Management System

5.5 Design of Database: Database design is the process of producing a detailed data model of database. This data model contains all the needed logical and physical design choices and physical storage parameters needed to generate a design in a data definition language, which can then be used to create a database. Figure 13 shows the database design of Online Marketplace.

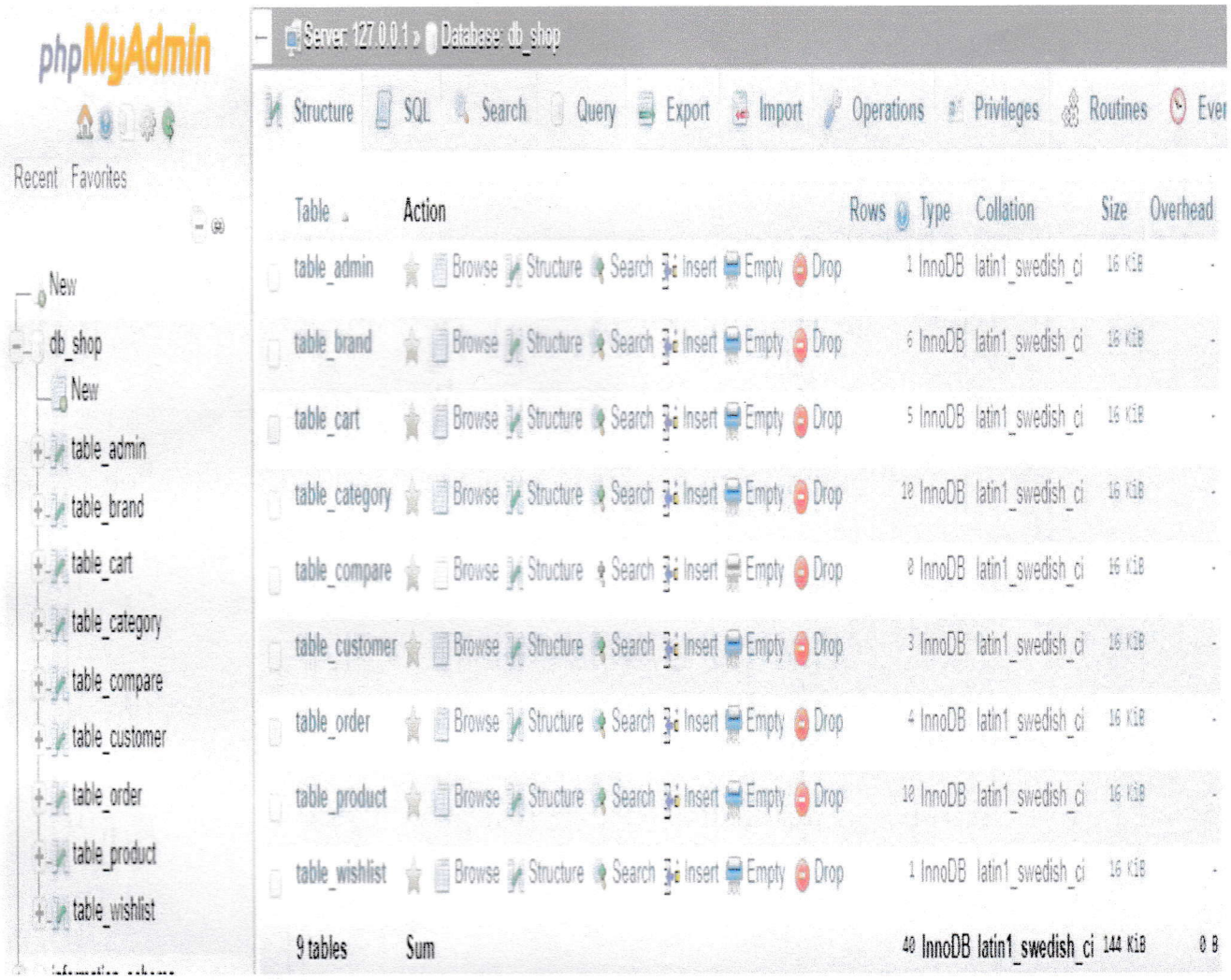


Figure 13: shows the database design of Online Cineplex Rent and Ticket Management System

Chapter 6
System Implementation

Implementation is the carrying out, execution, or practice of a plan, a method, or any design, idea, model, specification, standard or policy for doing something. As such, implementation is the action that must follow any preliminary thinking in order for something to actually happen.

In an information technology (IT) context, software or hardware implementation encompasses all the post-sale processes involved in something operating properly in its environment, including analyzing requirements, installation, configuration, customization, running, testing, systems integrations, user training, delivery and making necessary changes. The word "deployment" is sometimes used to mean the same thing.

In this chapter, there is a brief description of our project including images.

6.1 User Account Login:

This is the Account Login page of the system. New users can register from Register New Account option and if the user already an account then user can access his or her by e-mail address and password on this page. **Figure 14** shows the Account Login Page.

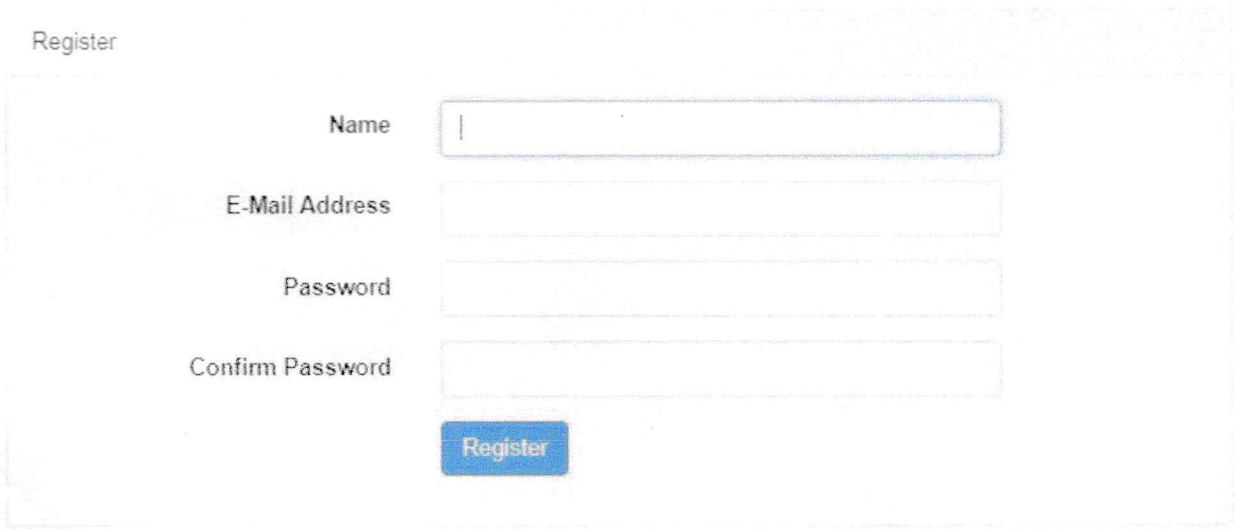


Figure 14: User Account Login

6.2 Admin Login: Figure 15 shows the Admin login page.

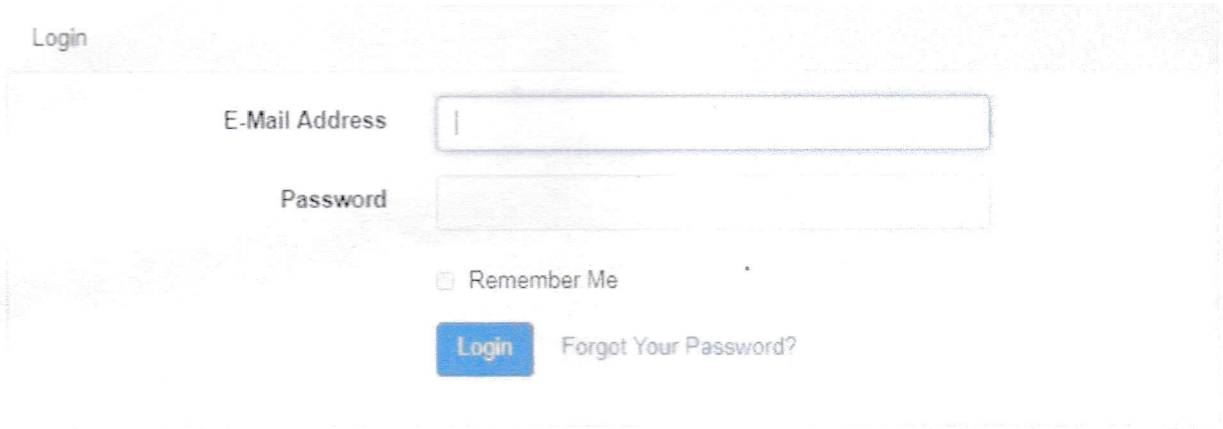


Figure 15: Admin login

6.3 Account Information: This is the Account information page of our system. When a user opens the system he will see this page. Figure 16 shows the Account Information page.

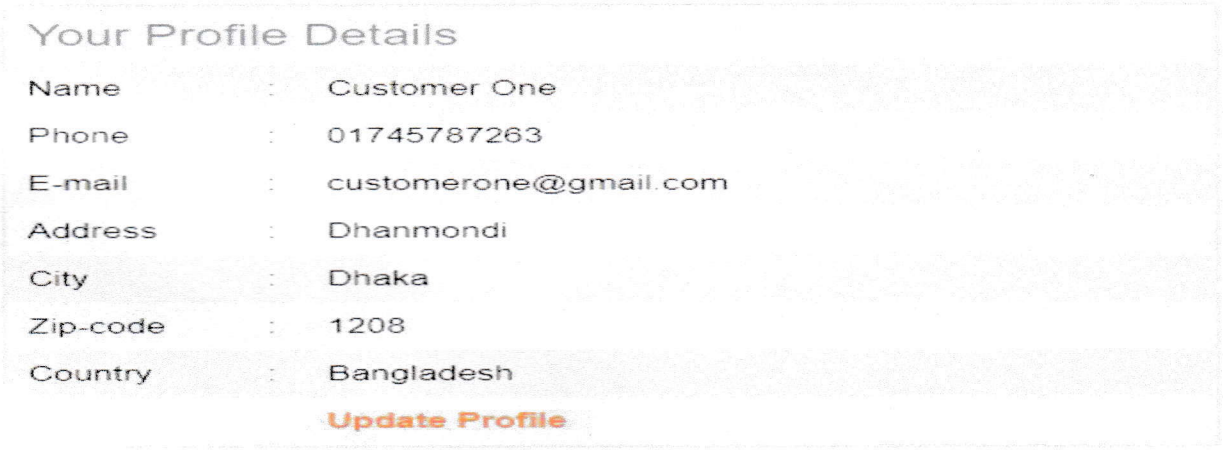


Figure 16: Account Information

6.4 Customer Details:

The customer details section is located in admin panel > Customer Details. Admin can access this page for getting the customer information. Figure 17 shows the Customer Details.

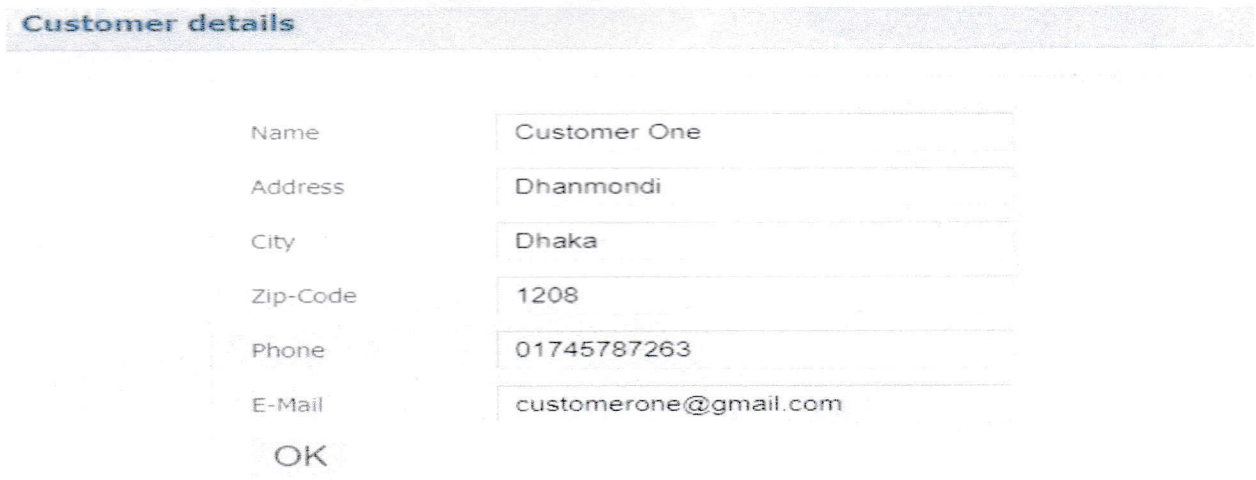


Figure 17: Customer Details

6.5 Customer Update Profile: After Login Account, It is Possible to Change Customer information by the customer. The Update Profile Section is Located Profile > Update Profile. Figure 18 shows the update profile page.

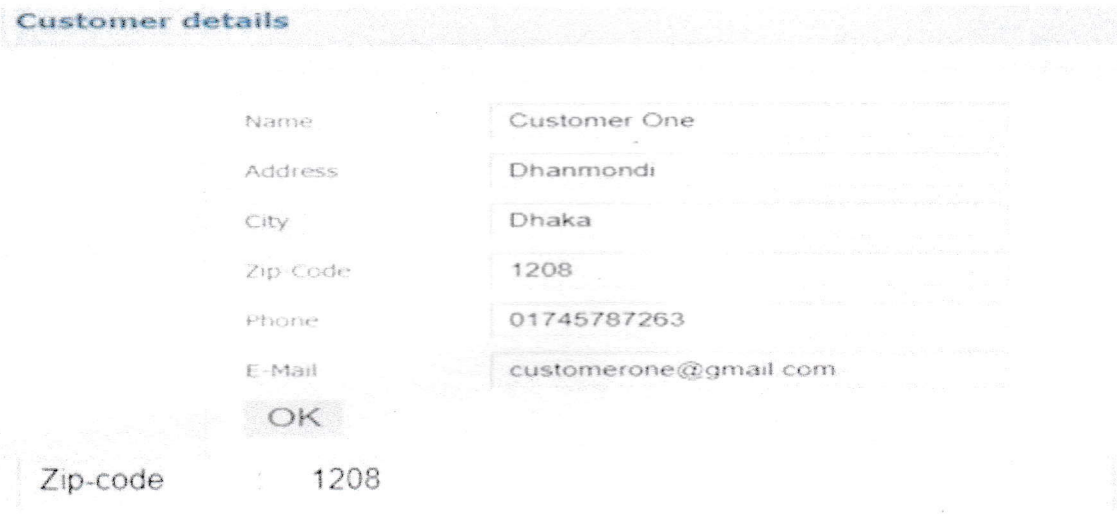


Figure 18: Update Profile

6.6 Ticket Details: The Ticket section, accessed under Catalog, will display all the movies available in the Cineplex store. Every movie ticket will contain the following information: Image, Movie Name, Price, Movie Category, Movie Summery etc. Figure 19 shows the Product details page.

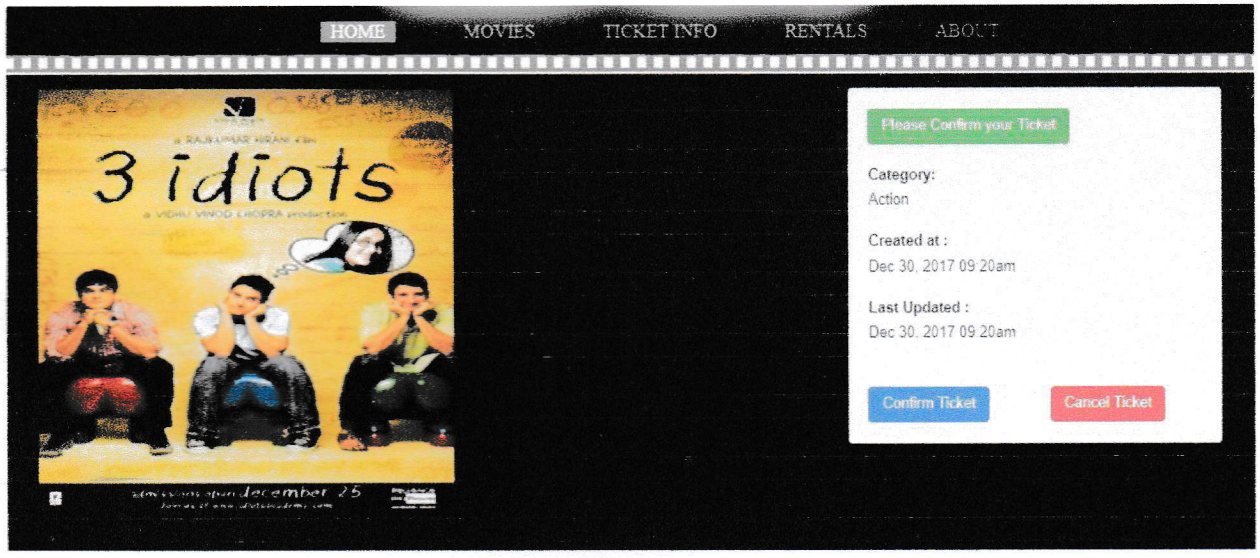


Figure 19: Product details

6.7 Running Movies: this feature shows the running movies at front page of the system in Figure 20

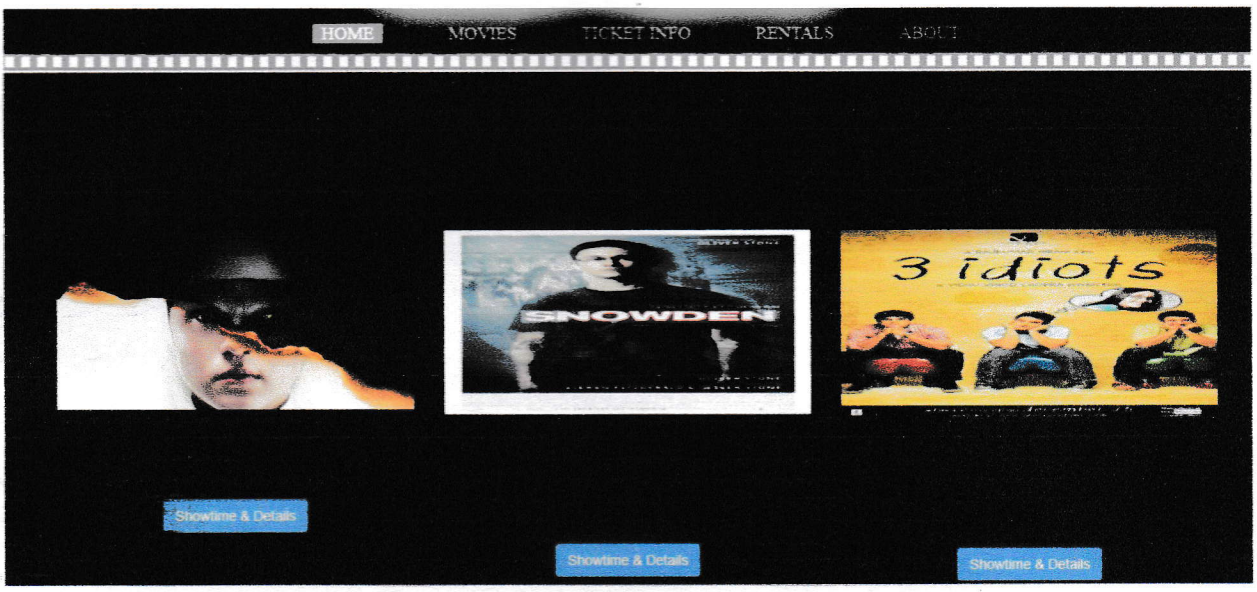


Figure 20: Running Movies

6.8 Payment Method: Figure 23 shows the Payment method. There are two option here, one is offline payment and another is online payment.

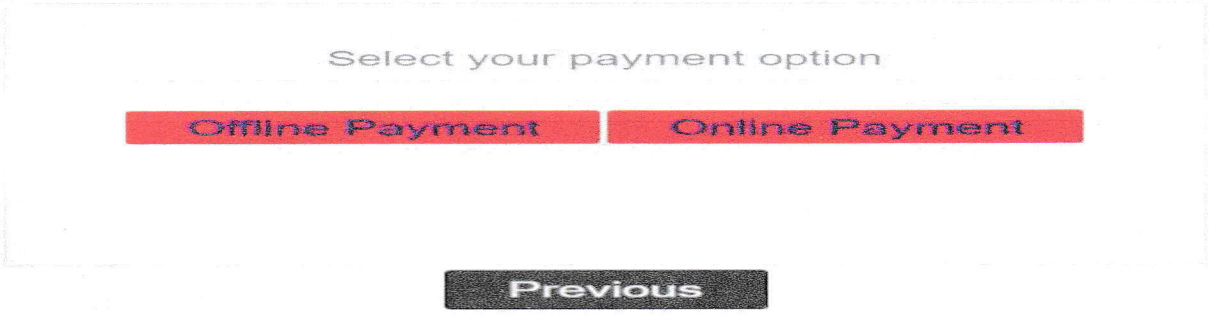


Figure 21: Payment method

6.9 Movies List: Figure 24 shows the movies list

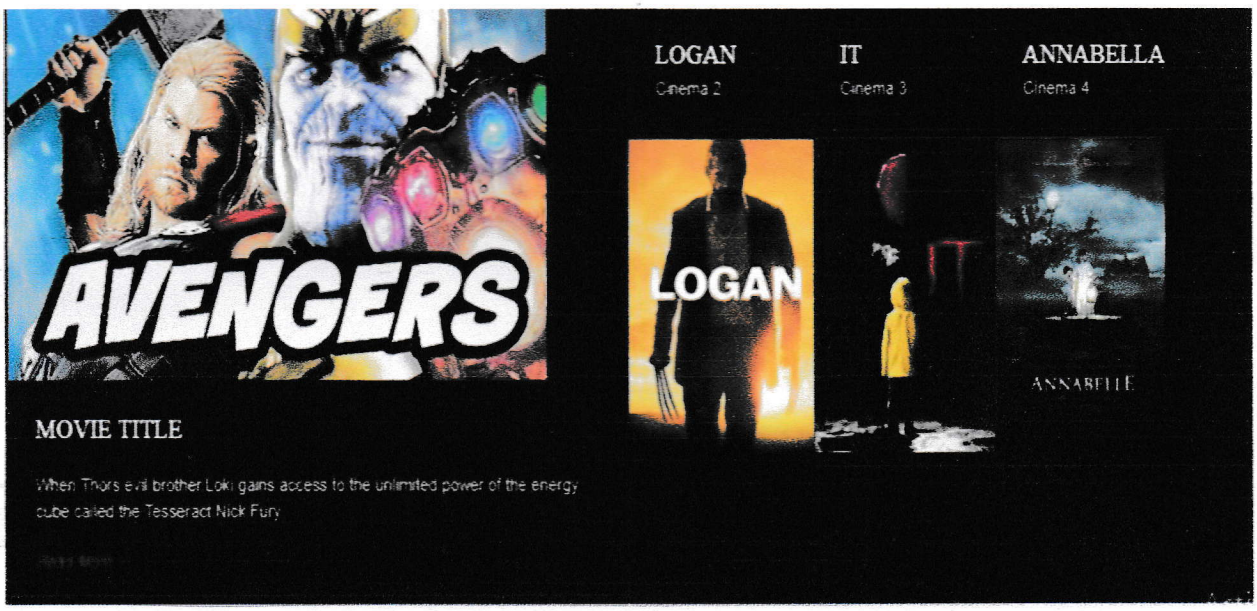


Figure 22: Movie List

6.10 Admin Panel Home:

The admin panel home is the first page when Admin entering into Admin Panel. Before entering the admin panel log in is needed with unique password. The main function of the admin panel is to give an overview of how the Cineplex is running. Which movie is running or coming next. Admin can post, Edit or delete movies from the archive option. Figure 25 shows the admin panel log in box. Figure shows the admin panel page.

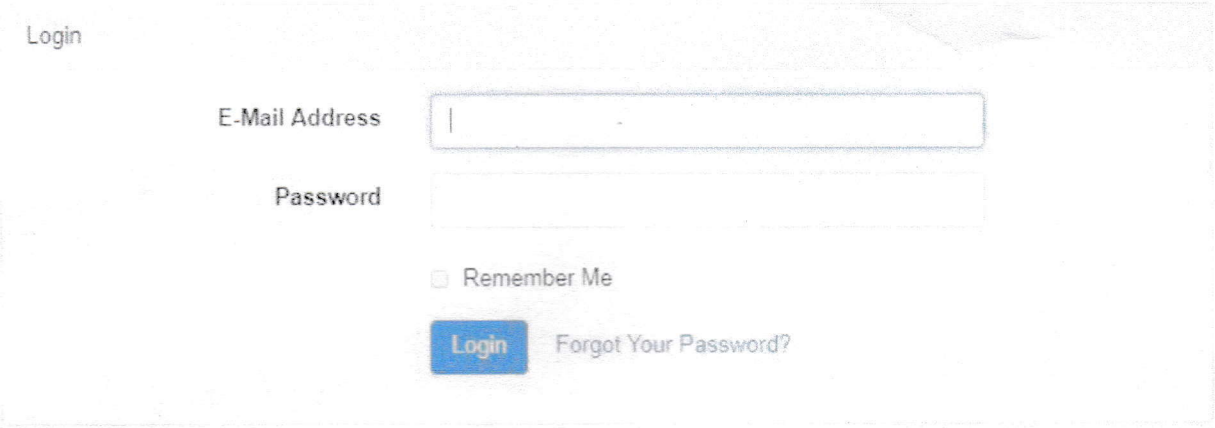


Figure 23: Admin Panel log in box.

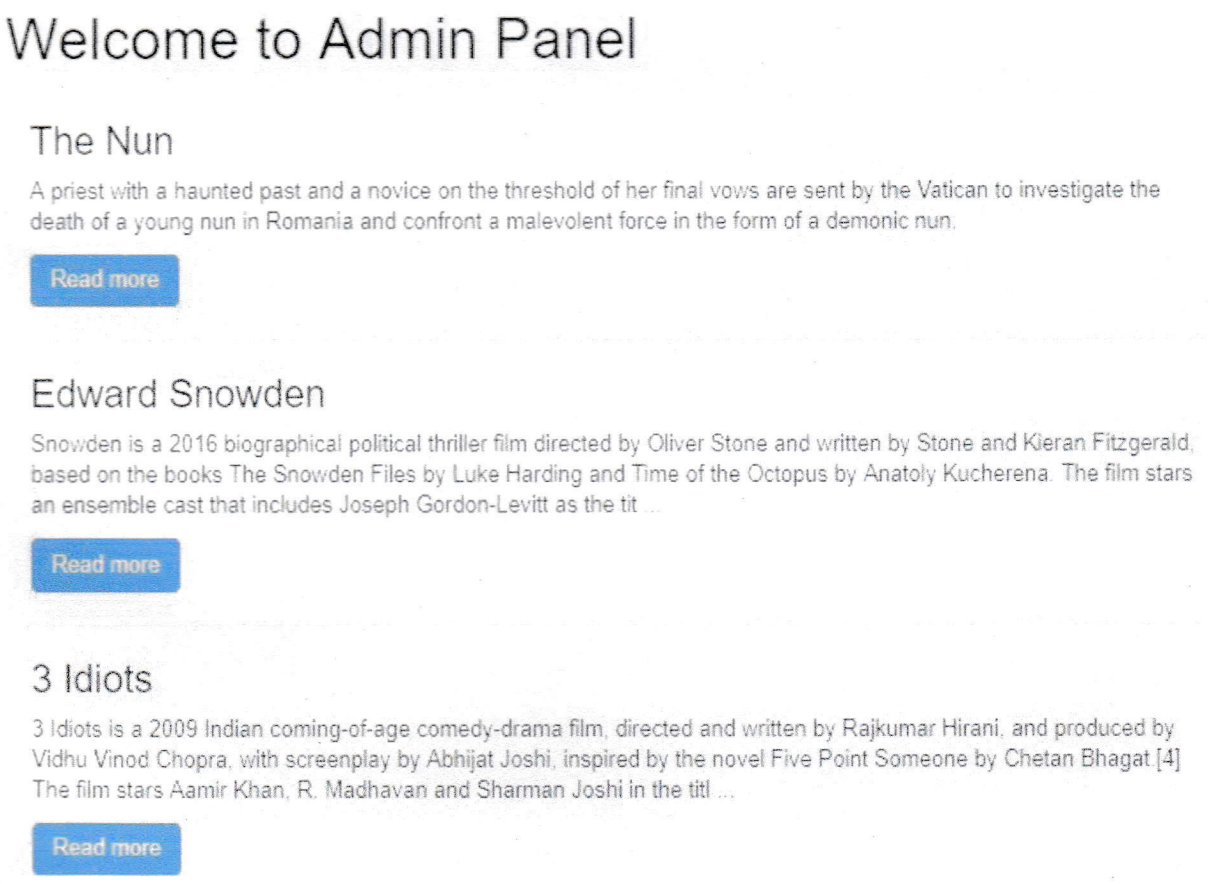


Figure 24: Admin Panel log in page.

6.11 Posting a New Movie: Figure 26 shows the posting new movie option. On this page the admin will get options to post a new movie in an organized way. The admin can add movie and can see movie list in the archive. From movie list the admin can also change, edit or delete the movie.

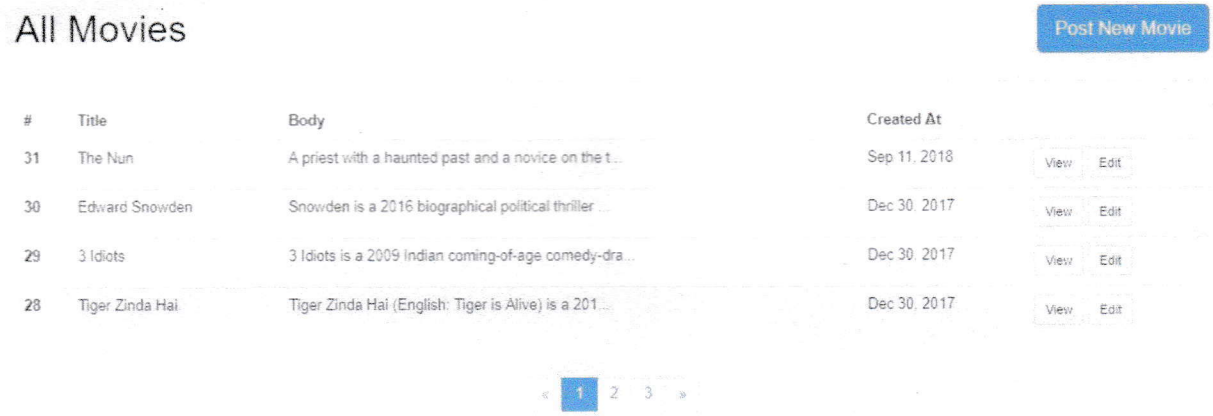


Figure 25: Post New Movie

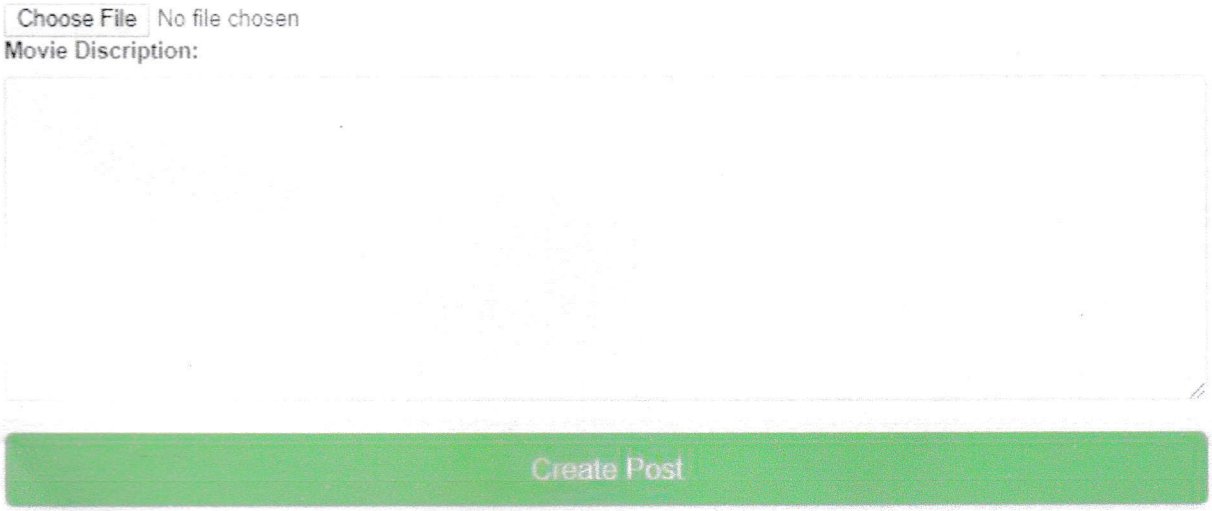


Figure 26 : Create New Post

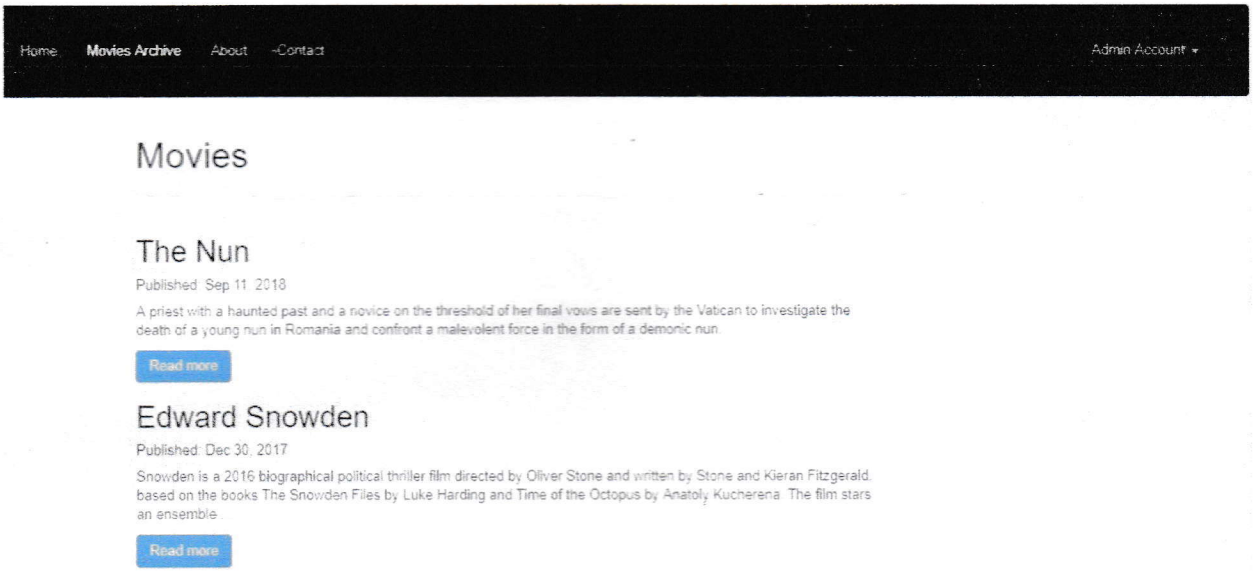


Figure 27: View movies archive

6.15 Movie Post Edit Option:

Here the admin will get editing option of a movie post. The admin can add movie information and can see movie list. From movie list the admin can also change, edit or delete the movie information. After editing the post admin can save the change and exit editing option.



Figure 28: Edit post option

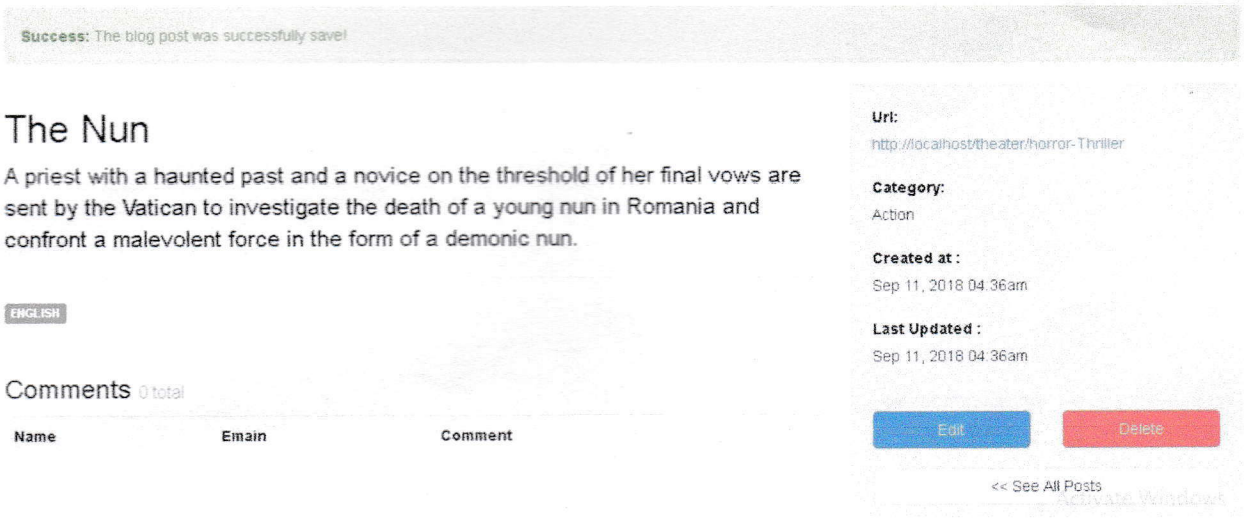


Figure 29: Successfully saved the post

Figure 29 shows the movie list which is accessible from admin panel can be edited.



Figure 30: View movie list with edit option

6.16 Rental Service:

There are some rental option of the complex. Clients can hire the complex for business conference or celebrate various types of parties like birthday, anniversary etc. The rental cost details is given there. Figure 30 shows the rental details.



Figure 31: rental details

Chapter 7

Security of the System

System Security is responsible for controlling access to system resources, which will include sensitive data. The system must therefore include a certain amount of protection for such data, and must in turn control access to those parts of the system that administer this protection. System security is concerned with all aspects of these arrangements. The purpose of security of the system is to prevent unauthorized access to the system as well as to store the data in such a way that I doesn't gets lost. That is why, in this project we have added the different and functionalities for both the Admin and the Users. The general Users can only view the data and shop that. But only the Admin has the privileges to modify the data which includes inserting, updating and deleting the data whenever it is necessary.

7.1 Security for User:

In this system, the Users have a unique id and password. So no unauthorized persons get the permission to access any other's account. And so it is secured with a id and password and so the personal information of the users are secured and nobody can access that until knowing the id and password.

7.2 Security for Admin

In this system, the Admin have also a unique id and password and it is fully secured. So unauthorized persons don't have the access to the Admin area and without entering into the Admin area, nobody can change or edit and delete the system information.

7.3 Secured with Session Cookie:

Webpages have no memories. A user going from page to page will be treated by the website as a completely new visitor. Session cookies enable the website you are visiting to keep track of your movement from page to page so you don't get asked for the same information you've already given to the site. Cookies allow you to proceed through many pages of a site quickly and easily without having to authenticate or reprocess each new area you visit.

Session cookies allow users to be recognized within a website so any page changes or item or data selection we do is remembered from page to page. The most common example of this functionality is the shopping cart feature of any e-commerce site. When we visit one page of a catalog and select some items, the session cookie remembers our selection so our shopping cart will have the items selected when we are ready to check out. Without session cookies, if we click CHECKOUT, the new page does not recognize our past activities on prior pages and our shopping cart will always be empty.

We can adjust your session cookies through the settings feature of your browser.

Without cookies, websites and their servers have no memory. A cookie, like a key, enables swift passage from one place to the next. Without a cookie every time we open a new web page the server where that page is stored will treat us like a completely new visitor.

Websites typically use session cookies to ensure that we are recognized when we move from page to page within one site and that any information we have entered is remembered. For example, if an e-commerce site did not use session cookies then items placed in a shopping basket would disappear by the time we reach the checkout. We can choose to accept session cookies by changing the settings in your browser.

7.4 Security of Database

Database security refers to the collective measures used to protect and secure a database or database management software from illegitimate use and malicious threats and attacks.

It is a broad term that includes a multitude of processes, tools and methodologies that ensure security within a database environment. Database security covers and enforces security on all aspects and components of databases. This includes:

- Data stored in database
- Database server
- Database management system (DBMS)
- Other database workflow applications

Database security is generally planned, implemented and maintained by a database administrator

Or other information security professional.

Some of the ways database security is analyzed and implemented include:

- Restricting unauthorized access and use by implementing strong and multifactor access and data management controls.
- Load/stress testing and capacity testing of a database to ensure it does not crash in a distributed denial of service (DoS) attack or user overload.
- Physical security of the database server and backup equipment from theft and natural disasters.
- Reviewing existing system for any known or unknown vulnerabilities and defining and implementing a road map/plan to mitigate them.

Chapter 8
Testing

Software testing is an activity to check whether the actual results match the expected results and to ensure that the software system is defect free. It involves execution of a software component or system component to evaluate one or more properties of interest.

Software testing also helps to identify errors, gaps or missing requirements in contrary to the actual requirements. It can be either done manually or using automated tools. Some prefer saying Software testing as a white box and black box testing.

It is an investigation conducted to provide stakeholders with information about the quality of the software product or service under test. Software testing can also provide an objective, independent view of the software to allow the business to appreciate and understand the risks of software implementation. Test techniques include the process of executing a program or application with the intent of finding software bugs (errors or other defects), and verifying that the software product is fit for use.

Software testing involves the execution of a software component or system component to evaluate one or more properties of interest.

8.1 Importance of Software Testing

Testing is important because software bugs could be expensive or even dangerous. Software bugs can potentially cause monetary and human loss, history is full of such examples.

- In April 2015, Bloomberg terminal in London crashed due to software glitch affected more than 300,000 traders on financial markets. It forced the government to postpone a 3bn pound debt sale.
- Nissan cars have to recall over 1 million cars from the market due to software failure in the airbag sensory detectors. There has been reported two accident due to this software failure.
- Starbucks was forced to close about 60 percent of stores in the U.S and Canada due to software failure in its POS system. At one point store served coffee for free as they unable to process the transaction.
- Some of the Amazon's third party retailers saw their product price is reduced to 1p due to a software glitch. They were left with heavy losses.
- Vulnerability in Window 10. This bug enables users to escape from security sandboxes through a flaw in the win32k system.

- In 2015 fighter plane F-35 fell victim to a software bug, making it unable to detect targets correctly.
- China Airlines Airbus A300 crashed due to a software bug on April 26, 1994, killing 264 innocent live
- In 1985, Canada's Therac-25 radiation therapy machine malfunctioned due to software bug and delivered lethal radiation doses to patients, leaving 3 people dead and critically injuring 3 others.
- In April of 1999, a software bug caused the failure of a $1.2 billion military satellite launch, the costliest accident in history
- In MAV of 1996, a software bug caused the bank accounts of 823 customers of a major U.S. bank to be credited with 920 million US dollars.

8.2 Types of Software Testing

Software Testing can be divided into the following types:

1. Process.
2. Life Cycle Activities.
3. Static Testing.
4. Dynamic Testing.
5. Planning.
6. Preparation.
7. Evaluation.
8. Software Products and related work products.

8.2.1 Process:

A software development process, also known as a software development lifecycle, is a structure imposed on the development of a software product. A software process is represented as a set of work phases that is applied to design and build a software product. Testing is a process rather than a single activity.

- In our system, we have followed a sequential process by which it has been possible to develop the system gradually.

8.2.2 All Life Cycle Activities:

Testing is a process that's take place throughout the Software Development Life Cycle (SDLC).

- The process of designing tests early in the life cycle can help to prevent defects from being introduced in the code. Sometimes it's referred as "verifying the test basis via the test design".
- The test basis includes documents such as the requirements and design specifications.

8.2.3 Static Testing:

It can test and find defects without executing code. Static Testing is done during verification process. This testing includes reviewing of the documents (including source code) and static analysis. This is useful and cost effective way of testing. For example: reviewing, walkthrough, inspection, etc.

8.2.4 Dynamic Testing:

In dynamic testing the software code is executed to demonstrate the result of running tests. It's done during validation process. For example: unit testing, integration testing, system testing, etc.

8.2.5 Planning:

We need to plan as what we want to do. We control the test activities, we report on testing progress and the status of the software under test.

- Planning has been very important for our project because without proper planning, nothing of this would be possible.
- To develop the system, we have first planned the development process and then made the plan into reality to complete the development of the system.

8.5.6 Preparation:

We need to choose what testing we will do, by selecting test conditions and designing test cases.

8.2.7 Evaluation:

During evaluation we must check the results and evaluate the software under test and the completion criteria, which helps us to decide whether we have finished testing and whether the software product has passed the tests.

8.2.8 Software products and related work products:

Along with the testing of code the testing of requirement and design specifications and also the related documents like operation, user and training material is equally important.

Chapter 9
Conclusion & Future Scope

In general, today's businesses must always strive to create the next best thing that consumers will want because consumers continue to desire their products, services etc. to continuously be better, faster, easier and cheaper. In this world of new technology, businesses need to accommodate to the new types of consumer needs and trends because it will prove to be vital to their business' success and survival. E-ticketing is continuously progressing and is becoming more and more important to businesses as technology continues to advance and is something that should be taken advantage of and implemented.

From the inception of the Internet and e-ticketing, the possibilities have become endless for both businesses and consumers. Creating more opportunities for profit and advancements for businesses, while creating more options for consumers.

However, just like anything else, e-ticketing has its disadvantages including consumer uncertainties, but nothing that cannot be resolved or avoided by good decision-making and business practices.

There are several factors and variables that need to be considered and decided upon when starting an e-ticketing business. Some of these include: types of e-commerce, marketing strategies, and countless more. If the correct methods and practices are followed, a business will prosper in an e-ticketing setting with much success and profitability.

At present the products which are getting high response are mobile and gadgets, fashion accessories and other electronic products. By their policies and payment methods encourage the buyer to go with them. Online marketplace business opportunities in Bangladesh is higher than any other country in the world because of the fact that India is the fasting growing economy and youngest country of the world, purchasing capability of the population is increasing day by day.

- We will collect the feedback from the customer and will develop and upgrade the system to make it better and better.
- We will develop different gateway of payment systems which will be available outside the country in the near future because of the increasing economy rate.
- We will integrate Point of Sale in shorter, the POS system with this website and it will make the system more advanced.
- We will try to develop it a multipurpose business website which means if a customer wants to maintain different business with the help of just one website, then he will be able to do it with the help of the system.
- We will make payments system more updated and easier. In near future clients can make their payments through mobile air credit which is easier than any other payment options.

REFERENCES:

1. https://www.w3schools.com/html/default.asp ,4.09.18
2. http://php.net/manual/en/tutorial.php ,4.09.18
3. http://stackoverflow.com/documentation/sql/topics ,4.09.18
4. https://www.codeschool.com/learn/php ,4.09.18
5. https://www.tutorialspoint.com/php/ ,05.09.18
6. https://www.codecademy.com/ ,06.09.18
7. https://devzone.zend.com/6/php-101-php-for-the-absolute-beginner/ ,06.09.18
8. **www.cineplexbd.com** , 10.09.18
9. **www.blockbasterbd.com** , 10.09.18
10. diuf.unifr.ch/is/userfiles/file/courses/eBiz_fs08/nicolas_bienz_paper.pdf **1Presented by Nicolas Beinz, 10.09.18**
11. **iosrjournals.org/iosr-jbm/papers/Vol18-issue6/Version-4/F1806043238.pdf** , by Dr. Chaya Bagrecha , Sadiq Alam
12. **http://www.ijicse.com/papers/dec14/ijicse141103.pdf_** by M.Prakas , S.Nandhini , 11.09.18
13. **https://www.computer.org/csdl/proceedings/icii/2001/7010/06/00982996.pdf** by Feng Bao, Lakshminarayanan Anantharaman, Robert Deng , 13.09.18

Appendix
Source Code:

Config.php:

```
<?php
define("DB_HOST", "localhost");
define("DB_USER", "root");
define("DB_PASS", "");
define("DB_NAME", " db_shop");
?>
```

Format.php:

```
<?php
class Format{
 public function formatDate($date){
  return date('F j, Y, g:i a', strtotime($date));
 }
  public function textShorten($text, $limit = 400){
    $text = $text. " ";
    $text = substr($text, 0, $limit);
    $text = substr($text, 0, strrpos($text, ' '));
    $text = $text.".....";
    return $text;
      }
 public function validation($data){
    $data = trim($data);
    $data = stripcslashes($data);
    $data = htmlspecialchars($data);
    return $data;
   }
 public function title(){
  $path = $_SERVER['SCRIPT_FILENAME'];
  $title = basename($path, '.php');
  if ($title == 'index') {
   $title = 'home';
  }elseif ($title == 'contact') {
   $title = 'contact';  }
  return $title = ucfirst($title);
```

```
} } ?>
```

Database.php

```
<?php
 $filepath = realpath(dirname(__FILE__));
 include_once ($filepath.'/../config/config.php');
?>
<?php
Class Database{
 public $host   = DB_HOST;
 public $user   = DB_USER;
 public $pass   = DB_PASS;
 public $dbname = DB_NAME;
  public $link;
 public $error;
 public function __construct(){
  $this->connectDB();
 }
private function connectDB(){
 $this->link = new mysqli($this->host, $this->user, $this->pass,
  $this->dbname);
 if(!$this->link){
   $this->error ="Connection fail".$this->link->connect_error;
  return false;
   }  }
public function select($query){
 $result = $this->link->query($query) or die($this->link->error.__LINE__);
 if($result->num_rows > 0){
   return $result;
 } else {
   return false;
   }  }
public function insert($query){
 $insert_row = $this->link->query($query) or die($this->link->error.__LINE__);
 if($insert_row){
   return $insert_row;
 } else {
   return false;
 } }
 public function update($query){
```

```
$update_row = $this->link->query($query) or die($this->link->error.__LINE__);
if($update_row){
return $update_row;
} else {
 return false;
  } }
public function delete($query){
$delete_row = $this->link->query($query) or die($this->link->error.__LINE__);
if($delete_row){
 return $delete_row;
} else {
 return false;
  }   }   }
```

Session.php

```
  <?php
class Session{
   public static function init(){
    if (version_compare(phpversion(), '5.4.0', '<')) {
        if (session_id() == '') {
           session_start();
        }
     } else {
        if (session_status() == PHP_SESSION_NONE) {
           session_start();
        }   }   }
    public static function set($key, $val){
   $_SESSION[$key] = $val;
     }
   public static function get($key){
   if (isset($_SESSION[$key])) {
    return $_SESSION[$key];
    } else {
    return false;
    }    }
   public static function checkSession(){
   self::init();
   if (self::get("adminlogin")== false) {
   self::destroy();
   header("Location:login.php");
```

```
    }    }
  public static function checkLogin(){
   self::init();
   if (self::get("adminlogin")== true) {
    header("Location:dashbord.php");
    }    }
   public static function destroy(){
   session_destroy();
   header("Location:login.php");
    }    } ?>
```

Adminlogin.php

```
<?php
$filepath = realpath(dirname(__FILE__));
  include_once ($filepath.'/../lib/Session.php');
  Session::checkLogin();
 include_once ($filepath.'/../lib/Database.php');
 include_once ($filepath.'/../helpers/Format.php');
?>
<?php
 class Adminlogin{
        private $db;
        private $fm;
   public function __construct()
        {
        $this->db= new Database();
        $this->fm= new Format();   }
     public function adminLogin($adminUser,$adminPass){
     $adminUser =  $this->fm->validation($adminUser);
     $adminPass =  $this->fm->validation($adminPass);
     $adminUser = mysqli_real_escape_string($this->db->link,$adminUser);
     $adminPass = mysqli_real_escape_string($this->db->link,$adminPass);
         if (empty( $adminUser) || empty( $adminPass)) {
                $loginmsg= "Username or Passward must not empty ";
                return $loginmsg;
         }else{
          $query = "SELECT * FROM table_admin WHERE adminUser='$adminUser'
AND adminPass= '$adminPass'";
                $result= $this->db->select($query);
                if ($result !=false) {
```

```
                $value= $result->fetch_assoc();
                session::set("adminlogin",true);
                session::set("adminId",$value['adminId']);
                session::set("adminName",$value['adminName']);
                session::set("adminUser",$value['adminUser']);
                header("Location:dashbord.php");
              }else{
                      $loginmsg= "Username or Passward not match ";
                      return $loginmsg;
              }   }   }   } ?>
```

Brand.php

```
<?php
$filepath = realpath(dirname(__FILE__));
include_once ($filepath.'/../lib/Database.php');
include_once ($filepath.'/../helpers/Format.php');   ?>
  <?php
  class Brand{
  private $db;
        private $fm;
         public function __construct()
                {
                $this->db= new Database();
                 $this->fm= new Format();   }
 public function brandInsert($brandName){
       $brandName =  $this->fm->validation($brandName);
       $brandName = mysqli_real_escape_string($this->db->link,$brandName);
                if (empty( $brandName)) {
                $msg = "<span class='error'>Brand field must not be empty</span> ";
                return $msg;
                 }else{
       $query= "INSERT INTO table_brand(brandName) VALUES('$brandName')";
                $brandinsert = $this->db->insert($query);
                if ($brandinsert) {
                $msg = "<span class='success'>Brand name Inserted Successfully</span>";
                   return $msg;
                     }else{
          $msg ="<span class='error'>Brand name Not Inserted</span>";
           return $msg;
                  }   }   }
```

```
public function getAllBrand(){
        $query = "SELECT * FROM table_brand ORDER BY brandId DESC ";
$result = $this->db->select($query);
return $result;
          }
public function getBrandById($id){
        $query = "SELECT * FROM table_brand WHERE brandId = '$id' ";
$result = $this->db->select($query);
return $result;
}
public function brandUpdate($brandName, $id){
          $brandName =  $this->fm->validation($brandName);
$brandName = mysqli_real_escape_string($this->db->link,$brandName);
$id = mysqli_real_escape_string($this->db->link,$id);
        if (empty( $brandName)) {
        $msg = "<span class='error'>Brand field must not be empty </span> ";
        return $msg;
}else{
        $query = "UPDATE table_brand SET brandName = '$brandName'
                  WHERE     brandId = '$id'";
        $updaed_row = $this->db->update($query);
        if ($updaed_row) {
                $msg = "<span class='success'>Brand Inserted Successfully</span>";
           return $msg;
           }else{
                $msg ="<span class='error'>Brand Not Updated</span>";
   return $msg;
           }  }  }
 public function delBrandById($id){
          $query = "DELETE FROM table_brand WHERE brandId ='$id'";
$deldata = $this->db->delete($query);
 if ($deldata) {
 $msg = "<span class='success'>Brand name deleted successfully</span>";
            return $msg;
  }else{
 $msg = "<span class='Error'>Brand name not deleted</span>";
            return $msg;
  }   }   } ?>
```

Cart.php

```
<?php
$filepath = realpath(dirname(__FILE__));
include_once ($filepath.'/../lib/Database.php');
include_once ($filepath.'/../helpers/Format.php');
?>
<?php
class Cart{
        private $db;
        private $fm;
        public function __construct()
                {
                 $this->db= new Database();
                 $this->fm= new Format();
                }
        public function addToCart($quantity, $id){
        $quantity =  $this->fm->validation($quantity);
              $quantity = mysqli_real_escape_string($this->db->link,$quantity);
              $productId = mysqli_real_escape_string($this->db->link,$id);
              $sId    = session_id();
              $squery = "SELECT *FROM table_product WHERE productId= '$productId'";
              $result = $this->db->select($squery)->fetch_assoc();
              $productName = $result['productName'];
              $price = $result['price'];
              $image = $result['image'];
 $checkquery = "SELECT *FROM table_cart WHERE productId= '$productId'
     AND sId = '$sId'";
      $getPro = $this->db->select($checkquery);
        if ($getPro) {
                $msg = "Product already added ! ";
                return $msg;
         }else{
                $query= "INSERT INTO
table_cart(sId,productId,productName,price,quantity,image)
        VALUES('$sId','$productId','$productName','$price','$quantity','$image')";
      $productinsert = $this->db->insert($query);
                if ($productinsert) {
                   header("Location:cart.php");
                    }else{
           header("Location:error.php");
           }    }   }
```

```
public function getCartProduct(){
  $sId = session_id();
  $query = "SELECT * FROM table_cart WHERE sId  = '$sId' ";
  $result = $this->db->select($query);
  return $result;
      }
  public function updateCartQuantity($cartId, $quantity){
    $cartId = mysqli_real_escape_string($this->db->link,$cartId);
    $quantity = mysqli_real_escape_string($this->db->link,$quantity);
    $query = "UPDATE table_cart SET quantity = '$quantity' WHERE cartId = '$cartId'";
          $updated_row = $this->db->update($query);
          if ($updated_row) {
          $msg = "<span class='success'>Quantity updated Successfully</span>";
            return $msg;
           }else{
           $msg ="<span class='error'>Quantity Not Updated</span>";
          return $msg;
           }  }
public function delProductByCart($delId){
  $delId = mysqli_real_escape_string($this->db->link,$delId);
  $query = "DELETE FROM table_cart WHERE cartId ='$delId'";
  $deldata = $this->db->delete($query);
  if ($deldata) {
  echo "<script>window.location = 'cart.php';</script>";
  }else{
  $msg = "<span class='Error'>Category not deleted</span>";
   return $msg;
    }     }
  public function checkCartTable(){
          $sId = session_id();
    $query = "SELECT * FROM table_cart WHERE sId  = '$sId' ";
          $result = $this->db->select($query);
          return $result;  }
  public function delCustomerCart(){
    $sId = session_id();
    $query = "DELETE FROM table_cart WHERE sId = '$sId'";
    $this->db->delete($query);
      }
  public function orderProduct($cmrId){
    $sId = session_id();
    $query = "SELECT * FROM table_cart WHERE sId  = '$sId' ";
```

```
            $getPro = $this->db->select($query);
            if ($getPro) {
              while ($result=$getPro->fetch_assoc()) {
                $productId = $result['productId'];
                $productName = $result['productName'];
                $quantity = $result['quantity'];
                $price = $result['price']*$quantity;
                $image = $result['image'];
    $query= "INSERT INTO table_order(cmrId,productId,productName,quantity,price,image)
          VALUES('$cmrId','$productId','$productName','$quantity','$price','$image')";
    $productinsert = $this->db->insert($query);
          }   }   }
        public function payableAmount($cmrId){
         $query = "SELECT price FROM table_order WHERE cmrId  = '$cmrId' AND date=
now()";
      $result = $this->db->select($query);
      return $result;
        }
        public function getOrderProduct($cmrId){
          $query = "SELECT * FROM table_order WHERE cmrId  = '$cmrId' ORDER BY date
DESC";
      $result = $this->db->select($query);
      return $result;
        }
        public function checkOrder($cmrId){
          $query = "SELECT * FROM table_order WHERE cmrId  = '$cmrId' ";
          $result = $this->db->select($query);
          return $result;
        }
        public function getAllOrderProduct(){
          $query = "SELECT * FROM table_order ORDER BY date DESC";
          $result = $this->db->select($query);
          return $result;
          }
        public function productShifted($id,$date,$price){
          $id = mysqli_real_escape_string($this->db->link,$id);
          $date = mysqli_real_escape_string($this->db->link,$date);
          $price = mysqli_real_escape_string($this->db->link,$price);
```



```
          $query = "UPDATE table_order SET status = '1' WHERE cmrId = '$id' AND date='$date'
AND price='$price'";
```

```
    $updated_row = $this->db->update($query);
    if ($updated_row) {
     $msg = "<span class='success'>Updated Successfully</span>";
      return $msg;
     }else{
         $msg = "<span class='error'>Not Successfully</span>";
        return $msg;
      }    }
     public function delProductShifted($id, $date, $price){
       $id = mysqli_real_escape_string($this->db->link,$id);
       $date = mysqli_real_escape_string($this->db->link,$date);
       $price = mysqli_real_escape_string($this->db->link,$price);
       $query = "DELETE FROM table_order WHERE cmrId = '$id' AND date='$date' AND
price='$price'";
        $deldata = $this->db->delete($query);
        if ($deldata) {
         $msg = "<span class='success'>Data deleted successfully</span>";
      return $msg;
        }else{
        $msg = "<span class='Error'>Data not deleted</span>";
      return $msg;
        }    }
     public function productShifConfirm($id, $date, $price){
       $id = mysqli_real_escape_string($this->db->link,$id);
       $date = mysqli_real_escape_string($this->db->link,$date);
       $price = mysqli_real_escape_string($this->db->link,$price);
       $query = "UPDATE table_order SET status = '2' WHERE cmrId = '$id' AND date='$date'
AND price='$price'";
    $updated_row = $this->db->update($query);
    if ($updated_row) {
     $msg = "<span class='success'>Updated Successfully</span>";
      return $msg;
     }else{
         $msg = "<span class='error'>Not Successfully</span>";
        return $msg;
       }  }  }  ?>
```

Category.php

```
<?php
```

```
$filepath = realpath(dirname(__FILE__));
  include_once ($filepath.'/../lib/Database.php');
 include_once ($filepath.'/../helpers/Format.php'); ?>
<?php
class Category{
      private $db;
      private $fm;
      public function __construct()
            {
             $this->db= new Database();
             $this->fm= new Format();
             }
     public function catInsert($catName){
       $catName =  $this->fm->validation($catName);
       $catName = mysqli_real_escape_string($this->db->link,$catName);
            if (empty( $catName)) {
            $msg = "<span class='error'>Category field must not be empty </span> ";
            return $msg;
             }else{
     $query= "INSERT INTO table_category(catName) VALUES('$catName')";
            $catinsert = $this->db->insert($query);
            if ($catinsert) {
            $msg = "<span class='success'>Category Inserted Successfully</span>";
               return $msg;
                }else{
      $msg ="<span class='error'>Category Not Inserted</span>";
        return $msg;
          }  }  }
public function getAllcat(){
     $query = "SELECT * FROM table_category ORDER BY catId DESC ";
     $result = $this->db->select($query);
     return $result;
   }
 public function getCatById($id){
$query = "SELECT * FROM table_category WHERE catId = '$id' ";
     $result = $this->db->select($query);
     return $result;
   }
  public function catUpdate($catName, $id){
  $catName =  $this->fm->validation($catName);
     $catName = mysqli_real_escape_string($this->db->link,$catName);
```

```
$id = mysqli_real_escape_string($this->db->link,$id);
        if (empty( $catName)) {
        $msg = "<span class='error'>Category field must not be empty </span> ";
        return $msg;
  }else{
 $query = "UPDATE table_category SET catName = '$catName' WHERE catId = '$id'";
        $updaed_row = $this->db->update($query);
        if ($updaed_row) {
        $msg = "<span class='success'>Category Inserted Successfully</span>";
          return $msg;
         }else{
         $msg ="<span class='error'>Category Not Updated</span>";
          return $msg;
         }   }   }
public function delCatById($id){
  $query = "DELETE FROM table_category WHERE cartId ='$id'";
  $deldata = $this->db->delete($query);
if ($deldata) {
 $msg = "<span class='success'>Category deleted successfully</span>";
          return $msg;
}else{
$msg = "<span class='Error'>Category not deleted</span>";
          return $msg;
}   }   } ?>
```

Customer.php

```
<?php
 $filepath = realpath(dirname(__FILE__));
 include_once ($filepath.'/../lib/Database.php');
 include_once ($filepath.'/../helpers/Format.php');
 ?>
<?php
 class Customer{
        private $db;
        private $fm;
        public function __construct(){
         $this->db= new Database();
          $this->fm= new Format();
         }
     public function customerRegistration($data){
```

```
        $name = mysqli_real_escape_string($this->db->link,$data['name']);
        $address = mysqli_real_escape_string($this->db->link,$data['address']);
        $city = mysqli_real_escape_string($this->db->link,$data['city']);
        $country = mysqli_real_escape_string($this->db->link,$data['country']);
        $zip = mysqli_real_escape_string($this->db->link,$data['zip']);
        $phone = mysqli_real_escape_string($this->db->link,$data['phone']);
        $email = mysqli_real_escape_string($this->db->link,$data['email']);
        $password = mysqli_real_escape_string($this->db->link,md5($data['password']));
        if ($name =="" || $address== "" || $city== "" || $country== "" || $zip== "" || $phone== "" ||
$email== "" || $password== "") {
            $msg = "<span class='error'>Fields must not be empty </span> ";
                    return $msg;
                }
                $mailquery = "SELECT *FROM table_customer WHERE email='$email' LIMIT 1 ";
                $mailcheck = $this->db->select($mailquery);
                if ($mailcheck != false) {
                    $msg = "<span class='error'>This E-mail already exist </span> ";
                    return $msg;
                }else{
        $query= "INSERT INTO
table_customer(name,address,city,country,zip,phone,email,password)
VALUES('$name','$address','$city','$country','$zip','$phone','$email','$password')";
            $productinsert = $this->db->insert($query);
                    if ($productinsert) {
                    $msg = "<span class='success'>Customer data Inserted Successfully</span>";
                       return $msg;
                        }else{
             $msg ="<span class='error'>Customer data not Inserted</span>";
                return $msg;
            } } }
        public function customerLogin($data){
        $email = mysqli_real_escape_string($this->db->link,$data['email']);
        $password = mysqli_real_escape_string($this->db->link,md5($data['password']));
        if (empty($email) || empty($password)) {
            $msg = "<span class='error'>Fields must not be empty </span> ";
                    return $msg;   }
        $query = "SELECT *FROM table_customer WHERE email = '$email' AND password =
'$password'";
        $result = $this->db->select($query);
        if ($result != false) {
            $value = $result->fetch_assoc();
```

```
            Session::set("cuslogin",true);
            Session::set("cmrId",$value['customerId']);
            Session::set("cmrName",$value['name']);
            header("Location:cart.php");
        }else{
            $msg = "<span class='error'> E-mail or password not match ! </span>";
                    return $msg;
        }
    }
    public function getCustomerData($id){
            $query = "SELECT * FROM table_customer WHERE customerId  = '$id' ";
                    $result = $this->db->select($query);
                    return $result;
            }
        public function customerUpdate($data, $cmrId){
        $name = mysqli_real_escape_string($this->db->link,$data['name']);
        $address = mysqli_real_escape_string($this->db->link,$data['address']);
        $city = mysqli_real_escape_string($this->db->link,$data['city']);
        $country = mysqli_real_escape_string($this->db->link,$data['country']);
        $zip = mysqli_real_escape_string($this->db->link,$data['zip']);
        $phone = mysqli_real_escape_string($this->db->link,$data['phone']);
        $email = mysqli_real_escape_string($this->db->link,$data['email']);
    if ($name =="" || $address== "" || $city== "" || $country== "" || $zip== "" || $phone== "" ||
$email== "") {
            $msg = "<span class='error'>Fields must not be empty </span> ";
                    return $msg;
              }else{
            $query = "UPDATE table_customer SET name = '$name', address = '$address', city =
'$city', country = '$country', zip = '$zip', phone = '$phone', email = '$email' WHERE customerId
= '$cmrId' ";
            $updated_row = $this->db->update($query);
                    if ($updated_row) {
                    $msg = "<span class='success'>Customer Profile Updated Successfully</span>";
                       return $msg;
                      }else{
              $msg ="<span class='error'>Customer Profile not Updated </span>";
               return $msg;
              }   }    }    } ?>
```

Product.php

```
<?php
```

```
$filepath = realpath(dirname(__FILE__));
include_once ($filepath.'/../lib/Database.php');
include_once ($filepath.'/../helpers/Format.php');
?>
<?php
class Product{
  private $fm;
        private $db;
        public function __construct(){
    $this->fm= new Format();
                $this->db= new Database();  }
   public function productInsert($data,$file){
       $productName = mysqli_real_escape_string($this->db->link,$data['productName']);
       $catId = mysqli_real_escape_string($this->db->link,$data['catId']);
         $brandId = mysqli_real_escape_string($this->db->link,$data['brandId']);
         $body = mysqli_real_escape_string($this->db->link,$data['body']);
         $price = mysqli_real_escape_string($this->db->link,$data['price']);
         $type = mysqli_real_escape_string($this->db->link,$data['type']);
   $permited  = array('jpg', 'jpeg', 'png', 'gif');
   $file_name = $file['image']['name'];
   $file_size = $file['image']['size'];
   $file_temp = $file['image']['tmp_name'];
   $div = explode('.', $file_name);
   $file_ext = strtolower(end($div));
   $unique_image = substr(md5(time()), 0, 10).'.'.$file_ext;
   $uploaded_image = "upload/".$unique_image;
   if ($productName == "" || $catId== "" || $brandId== "" || $body== "" || $price== "" ||
$file_name== "" || $type== "") {
        $msg = "<span class='error'>Fields must not be empty </span> ";
                return $msg;
                  }elseif ($file_size >1048567) {
                  echo "<span class='error'>Image Size should 1be less then 1MB!
                  </span>";
         } elseif (in_array($file_ext, $permited) === false) {
                  echo "<span class='error'>You can upload only:-"
                  .implode(', ', $permited)."</span>";
                 } else{
       move_uploaded_file($file_temp, $uploaded_image);
$query = "INSERT INTO table_product(productName,catId,brandId,body,price,image,type)
VALUES('$productName','$catId','$brandId','$body','$price','$uploaded_image','$type')";
    $productinsert = $this->db->insert($query);
```

```
            if ($productinsert) {
            $msg = "<span class='success'>Product inserted Successfully</span>";
               return $msg;
                }else{
        $msg ="<span class='error'>Product Not inserted</span>";
           return $msg;
       } } }
    public function getAllProduct(){
  $query  = "SELECT table_product.*, table_category.catName, table_brand.brandName FROM
table_product INNER JOIN table_category ON table_product.catId = table_category.catId
INNER JOIN table_brand ON table_product.brandId = table_brand.brandId  ORDER BY
table_product.productId DESC";
        $result = $this->db->select($query);
        return $result;
    }
    public function getProById($id){
   $query = "SELECT * FROM table_product WHERE productId = '$id' ";
        $result = $this->db->select($query);
        return $result;
     }
    public function productUpdate($data, $file, $id){
    $productName = mysqli_real_escape_string($this->db->link,$data['productName']);
    $catId = mysqli_real_escape_string($this->db->link,$data['catId']);
    $brandId = mysqli_real_escape_string($this->db->link,$data['brandId']);
   $body = mysqli_real_escape_string($this->db->link,$data['body']);
   $price = mysqli_real_escape_string($this->db->link,$data['price']);
   $type = mysqli_real_escape_string($this->db->link,$data['type']);
   $permited  = array('jpg', 'jpeg', 'png', 'gif');
           $file_name = $file['image']['name'];
           $file_size = $file['image']['size'];
           $file_temp = $file['image']['tmp_name'];
           $div = explode('.', $file_name);
           $file_ext = strtolower(end($div));
           $unique_image = substr(md5(time()), 0, 10).'.'.$file_ext;
           $uploaded_image = "upload/".$unique_image;
 if ($productName =="" ||$catId== ""|| $brandId== "" || $body== "" || $price=="" || $type== ""){
        $msg = "<span class='error'>Fields must not be empty </span> ";
                return $msg;
                    }else{
                    if (!empty($file_name)) {
                     if ($file_size >1048567) {
```

```
                echo "<span class='error'>Image Size should be less then 1MB!
                </span>";
            } elseif (in_array($file_ext, $permited) === false) {
                echo "<span class='error'>You can upload only:-"
                .implode(', ', $permited)."</span>";
               } else{
        move_uploaded_file($file_temp, $uploaded_image);
        $query = "UPDATE table_product SET productName = '$productName',  catId = '$catId',
brandId = '$brandId',  body = '$body', price = '$price',  image = '$uploaded_image',
         type = '$type' WHERE productId ='$id' ";
        $productupdated = $this->db->update($query);
                if ($productupdated) {
                $msg = "<span class='success'>Product updated Successfully</span>";
                  return $msg;
                   }else{
         $msg ="<span class='error'>Product Not updated</span>";
           return $msg;
           }   }
    } else{
  $query = "UPDATE table_product SET  productName = '$productName', catId = '$catId',
  brandId = '$brandId', body = '$body', price = '$price', type = '$type' WHERE productId ='$id' ";
  $productupdate = $this->db->update($query);
                if ($productupdate) {
                $msg = "<span class='success'>Product updated Successfully</span>";
                 return $msg;
                    }else{
        $msg ="<span class='error'>Product Not updated</span>";
         return $msg;
        }  }  }  }
    public function delProById($id){
        $query = "SELECT * FROM table_product WHERE productId = '$id'";
        $getData = $this->db->select($query);
        if ($getData) {
                while ($delImg = $getData->fetch_assoc()){
                        $dellink = $delImg['image'];
                        unlink($dellink);
                }   }
        $delquery = "DELETE FROM table_product WHERE productId = '$id' ";
        $deldata = $this->db->delete($delquery);
        if ($deldata) {
        $msg = "<span class='success'>Product deleted successfully</span>";
```

```
            return $msg;
           }else{
          $msg = "<span class='Error'>Product not deleted</span>";
                return $msg;
           }   }
  public function getFeaturedProduct(){
        $query = "SELECT * FROM table_product WHERE type='0' ORDER BY productId
DESC LIMIT 4 ";
        $result = $this->db->select($query);
        return $result;
   }
   public function getNewProduct(){
        $query = "SELECT * FROM table_product ORDER BY productId DESC LIMIT 4 ";
        $result = $this->db->select($query);
        return $result;
     }
   public function getSingleProduct($id){
        $query = "SELECT p.*, c.catName, b.brandName FROM table_product as p,
table_category as c, table_brand as b WHERE p.catId= c.catId AND p.brandId = b.brandId AND
p.productId= '$id' ";
        $result = $this->db->select($query);
        return $result;
      }
  public function latestFromIphone(){
    $query = "SELECT * FROM table_product WHERE brandId='3' ORDER BY productId
DESC LIMIT 1 ";
    $result = $this->db->select($query);
    return $result;
  }
  public function latestFromSamsung(){
    $query = "SELECT * FROM table_product WHERE brandId='1' ORDER BY productId
DESC LIMIT 1 ";
    $result = $this->db->select($query);
    return $result;
  }
   public function latestFromWalton(){
    $query = "SELECT * FROM table_product WHERE brandId='5' ORDER BY productId
DESC LIMIT 1 ";
    $result = $this->db->select($query);
    return $result;
      }
```

```
    public function latestFromRfl(){
    $query = "SELECT * FROM table_product WHERE brandId='6' ORDER BY productId
DESC LIMIT 1 ";
    $result = $this->db->select($query);
    return $result;
        }
    public function productByCat($id){
    $catId = mysqli_real_escape_string($this->db->link,$id);
    $query = "SELECT * FROM table_product WHERE catId = '$catId' ";
    $result = $this->db->select($query);
    return $result;
      }
     public function insertCompareData($cmprid,$cmrId){
    $cmrId = mysqli_real_escape_string($this->db->link,$cmrId);
    $productId = mysqli_real_escape_string($this->db->link,$cmprid);
 $cquery="SELECT *FROM table_compare  WHERE cmrId='$cmrId'AND productId =
'$productId' ";
     $check = $this->db->select($cquery);
     if ($check) {
       $msg = "<span class='Error'>Already Added</span>";
      return $msg;
     }
   $query="SELECT *FROM table_product  WHERE productId='$productId' ";
          $result = $this->db->select($query)->fetch_assoc();
          if ($result) {
              $productId = $result['productId'];
              $productName = $result['productName'];
              $price = $result['price'];
   $image = $result['image'];
   $query= "INSERT INTO table_compare(cmrId,productId,productName,price,image)
     VALUES('$cmrId','$productId','$productName','$price','$image')";
    $inserted_row = $this->db->insert($query);
    if ($inserted_row) {
        $msg = "<span class='success'>Added to Compare</span>";
      return $msg;
        }else{
        $msg = "<span class='Error'>Not Added</span>";
     return $msg;
       }  }  }
  public function getCompareData($cmrId){
 $query = "SELECT *FROM table_compare WHERE cmrId = '$cmrId' ORDER BY id DESC";
```

```
    $result = $this->db->select($query)
return $result;
    }
    public function delCompareData($cmrId){
     $query = "DELETE FROM table_compare WHERE cmrId = '$cmrId' ";
     $deldata = $this->db->delete($query);
    }
    public function saveWishListData($id,$cmrId){
     $cquery="SELECT *FROM table_wishlist  WHERE cmrId='$cmrId'AND productId = '$id' ";
     $check = $this->db->select($cquery);
      if ($check) {
        $msg = "<span class='Error'>Already Added</span>";
       return $msg;
      }
     $query = "SELECT * FROM table_product WHERE productId  = '$id' ";
           $result = $this->db->select($query)->fetch_assoc();
           if ($result) {
               $productId = $result['productId'];
               $productName = $result['productName'];
               $price = $result['price'];
               $image = $result['image'];
   $query= "INSERT INTO table_wishlist(cmrId,productId,productName,price,image)
         VALUES('$cmrId','$productId','$productName','$price','$image')";
         $inserted_row = $this->db->insert($query);
         if ($inserted_row)
              {
         $msg = "<span class='success'>Added to Wishlist Page</span>";
   return $msg;
     }else{
         $msg = "<span class='Error'>Not Added</span>";
     return $msg;
          }   }   }
   public function getWlistData($cmrId){
   $query = "SELECT *FROM table_wishlist WHERE cmrId = '$cmrId' ORDER BY id DESC";
     $result = $this->db->select($query);
     return $result;
   }
   public function delWlistData($cmrId, $productId){
  $query = "DELETE FROM table_wishlist WHERE cmrId = '$cmrId' AND productId =
$productId";
     $deldata = $this->db->delete($query);
```