CHAPTER-1

INTRODUCTION

1.1SYNOPSIS

My project entitled as “Online Salon and Spa Booking System”. Salon and Spa is a Website developed for a Salon and Spa in order to book appointments, pay, get feedback from Customers. This System is provided with proper registration form with date and time slot and types of services the user needs.

This system helps the Salon and Spa to overcome the difficulty of managing their customers and to boost up their business through online booking as well as, Customers to book their appointment by their desired date, time slot and types of services which they can pay early through online or by cash.

The home page contains various links to proceed like men booking, women booking, Admin, and Feedback to report for improvements or a message for Salon.

Both men and women modules have common features like to choose the desired type of services as they need and provide feedback.

This package is developed in Windows platform. The programming Language used is ASP.NET . SQL Server is used as an backend database for details.

1.2 MODULE DESCRIPTION

The "ONLINE SALON AND SPA BOOKING SYSTEM" has four modules.

- MEN BOOKING
- WOMEN BOOKING
- ADMIN
- FEEDBACK

MEN BOOKING :

This module help the men customers to book their desired type of services and book their appointment in their suitable Date and Time Slot.

WOMEN BOOKING :

This module help the women customers to book their desired type of services and book their appointment in their suitable Date and Time Slot.

ADMIN :

This module help the admin to check today's appointments, customer's details, payment details and their valuable feedbacks.

FEEDBACK :

This module helps salon to improve the service by the feedbacks which they receive from customers.

CHAPTER-2

SYSTEM ANALYSIS

2.1 EXISTING SYSTEM

Visiting a salon or spa without a prior appointment can keep a person wait for hours, it takes a lot of precious time of customers. Queue time can be longer than customer expected. Sometimes customers gets frustrated by waiting for a day to get a haircut.

2.1.1 DISADVANTAGE OF EXISTING SYSTEM

- Waiting Time
- Long Queue
- Cannot choose desired Time Slot
- Absence of salon
- Inability of Online Payment

2.2 PROPOSED SYSTEM

"ONLINE SALON AND SPA BOOKING SYSTEM"

The proposed system explains what you are going to do in this project which explains procees and how the project is working. This proposed System mainly used to time consuming. This system is mainly used to solve the drawbacks of the existing system.

2.2.1 ADVANTAGES

- Customer can easily book them appointment by their desired date, time slot and type of services.
- The payment can be done easily through Online.
- Computerized System
- Consume less time
- Data can be retrieved easily.

2.3 SYSTEM SPECIFICATION

2.3.1 HARDWARE SPECIFICATION

- Processor : AMD A-8-6410 APU Radeon R5 Grahics 2.00 GHz
- RAM : 4GB
- Hard disk : 1TB
- Monitor : 18.5”LED Monitor

2.3.2 SOFTWARE SPECIFICATION

- Operating system : Windows 10
- Front end : ASP.NET
- Back end : SQL

CHAPTER-3

SOFTWARE DESCRIPTION

3.1 FRONT-END

ASP.NET

ASP.NET is a web application framework designed and developed by Microsoft**.** ASP.NET is open source and a subset of the .NET Framework and successor of the classic ASP (Active Server Pages). Version 1.0 of the .NET Framework was first released in January 2002.

.NET Framework is used to create a variety of applications and services like Console, Web, Windows, etc. But ASP.NET is only used to create web applications and web services. That's why we termed ASP.NET as a subset of the .NET Framework.

The ASP.NET application codes can be written in any of the following languages:

- C#
- Visual Basic.Net
- Jscript
- J#

ASP.NET is used to produce interactive, data-driven web applications over the internet. It consists of a large number of controls such as text boxes, buttons, and labels for assembling, configuring, and manipulating code to create HTML pages.

APS.NET WEB FORMS MODELS

ASP.NET web forms extend the event-driven model of interaction to web applications. The browser submits a web form to the web server and the server returns a full markup page or HTML page in response.

THE ASP.NET COMPONENT MODEL

The ASP.NET component model provides various building blocks of ASP.NET pages. It is an object model, which describes:

- Server-side counterparts of almost all HTML elements or tags, such as <form> and <input>.
- Server controls, which help in developing complex user interfaces. For example, the Calendar control or the Grid view control.

ASP.NET is a technology, which works on the .Net framework that contains all web-related functionalities. The .Net framework is made of an object-oriented hierarchy. An ASP.NET web application is made of pages. When a user requests an ASP.NET page, the IIS delegates the processing of the page to the ASP.NET runtime system.

COMPONENT OF .NET FRAMEWORK 3.5

Before going to the next session on Visual Studio.Net, let us go through the various components of the .Net framework 3.5. The following table describes the components of the .Net framework 3.5 and the job they perform:

ASP.NET life cycle specifies, how:

- ASP.NET processes pages to produce dynamic output
- The application and its pages are instantiated and processed
- ASP.NET compiles the pages dynamically

The ASP.NET life cycle could be divided into two groups:

- Application Life Cycle
- Page Life Cycle

APS.NET APPLICATION LIFE CYCLE

The application life cycle has the following stages:

- User requests accessing application resource, a page. The browser sends this request to the webserver.
- A unified pipeline receives the first request and the following events take place:
    - An object of the class Application Manager is created.
    - An object of the class Hosting Environment is created to provide information regarding the resources.
    - Top-level items in the application are compiled.
- The request is processed by the HTTP Application class. Different events are raised by this class for processing the request.
- Response objects are created. The application objects such as HTTP Context, HTTP Request, and HTTP Response are created and initialized.
- An instance of the HTTP Application object is created and assigned to the request.

ASP.NET LIFE CYCLE EVENTS

At each stage of the page life cycle, the page raises some events, which could be coded. An event handler is a function or subroutine, bound to the event, using declarative attributes such as Onclick or handle.

Following are the page's life cycle events:

- **PreInit** - PreInit is the first event in the page life cycle. It checks the IsPostBack property and determines whether the page is a postback. It sets the themes and master pages, creates dynamic controls, and gets and sets profile property values.

This event can be handled by overloading the OnPreInit method or creating a Page_PreInit handler.

- **Init** - Init event initializes the control property and the control tree is built. This event can be handled by overloading the OnInit method or creating a Page_Init handler.
- **InitComplete** - InitComplete event allows tracking of view state. All the controls turn on view-state tracking.
- **LoadViewState** - LoadViewState event allows loading view state information into the controls.
- **LoadPostData** - During this phase, the contents of all the input fields are defined with the <form> tag processed.
- **PreLoad** - PreLoad occurs before the postback data is loaded in the controls. This event can be handled by overloading the OnPreLoad method or creating a Page_PreLoad handler.

- **Load** - The Load event is raised for the page first and then recursively for all child controls. The controls in the control tree are created. This event can be handled by overloading the OnLoad method or creating a Page_Load handler.
- **LoadComplete** - The loading process is completed, control event handlers are run, and page validation takes place. This event can be handled by overloading the OnLoadComplete method or creating a Page_LoadComplete handler
- **PreRender** - The PreRender event occurs just before the output is rendered. By handling this event, pages and controls can perform any updates before the output is rendered.
- **PreRenderComplete** - As the PreRender event is recursively fired for all child controls, this event ensures the completion of the pre-rendering phase.
- **SaveStateComplete** - State of control on the page is saved. Personalization, control state, and view state information are saved. The HTML markup is generated. This

stage can be handled by overriding the Render method or creating a Page_Render handler.

- **UnLoad** - The UnLoad phase is the last phase of the page life cycle. It raises the UnLoad event for all controls recursively and lastly for the page itself. Final cleanup is done and all resources and references, such as database connections, are freed. This event can be handled by modifying the OnUnLoad method or creating a Page_UnLoad handler.

THE ASP.NET MVC

It allows for building web pages using the **MODEL-VIEW-CONTROLLER** design pattern

THE ASP.NET WEB PAGES

A lightweight syntax for adding dynamic code and data access directly inside **HTML** Markup.

ASP.NET LIFE CYCLE SPECIFIES,

- ASP.NET process pages to produce dynamic output
- The application and its pages are instantiated and processed
- ASP.NET complies with the pages dynamically

3.2 BACK-END

SQL SERVER

SQL is a language to operate databases; it includes database creation, deletion, fetching rows, modifying rows, etc.

SQL Server is an **ANSI** (American National Standard Institute) standard language, but there are many different versions of **SQL** language

What is SQL?

SQL is Structured Query Language, which is a computer language for storing, manipulating, and retrieving data stored in a relational database.

SQL is the standard language for relational database systems. All the Relational Database Management systems **(RDMS)** like MySQL, MS-ACCESS, Oracle, Sybase, Informix, Postgres, and SQL servers use SQL as their standard database language

Also, they are using different dialects, such as,

- **MS SQL** server using **T-SQL**,
- Oracle using **PL/SQL,**
- MS ACCESS version of SQL is called JET SQL (native format) etc.

SQL IS WIDELY POPULAR BECAUSE IT OFFERS THE FOLLOWING ADVANTAGES,

- Allows users to access data in the relational database management system.
- Allows the users to describe the data.
- Allows the users to define the data in a database and manipulate the data.

A BRIEF HISTORY OF SQL

- **1970** – Dr. Edgar F. "Ted" Codd of IBM is known as the father of relational databases, He described a relational model for databases.
- **1974** – Structured Query Language appeared.
- **1978** – IBM worked to develop Codd's idea and released a product named System/R.
- **1986** – IBM developed the prototype of a relational database standardized by ANSI. The First relational database was released by Relational Software

SQL Commands

The standard SQL commands to interact with the relational databases are CREATE, SELECT, INSERT, UPDATE, DELETE and DROP. These commands can be classified into the following groups based on their nature.

NULL Value

A NULL value in a table is a value in a field that appears to be blank, which means a field with a NULL value is a field with no value. It is very important to understand that a NULL value is different than a zero value or a field that contains spaces. A field with a NULL value is the one that has been left blank during record creation.

SQL Constraints

Constraints are the rules enforced on data columns on a table. These are used to limit the type of data that can go into a table. Constraints are the rules enforced on the data columns of a table. These are used to limit the type of data that can go into a table. This ensures the accuracy and reliability of the data in the database.

- Following are some of the most commonly used constraints available in SQL. These constraints have already been discussed in the SQL -RDBMS Concept chapter, but it's worth revising them at this point.

- **NOT NULL Constraints** − Ensures that a column cannot have a NULL value.

- **DEFAULT Constraints** − Provides a default value for a column when none is specified.

- **UNIQUE Constraints** − Ensures that all values in a column are different.

- **PRIMARY KEY** − Uniquely identifies each row/record in a database table.

- **FOREIGN KEY** − Uniquely identifies a row/record in any of the given database tables.
- **CHECK Constraints** − The CHECK constraint ensures that all the values in a column satisfy certain conditions.

- **INDEX** − Used to create and retrieve data from the database very quickly.

SQL Server

SQL Server is a relational database management system, or RDBMS, developed and marketed by Microsoft.

Similar to other RDBMS software, SQL Server is built on top of SQL, a standard programming language for interacting with relational databases. SQL Server is tied to Transact-SQL, or T-SQL, the Microsoft's implementation of SQL that adds a set of proprietary programming constructs.

SQL Server works exclusively in the Windows environment for more than 20 years. In 2016, Microsoft made it available on Linux. SQL Server 2017 became generally available in October 2016 that ran on both Windows and Linux.

3.3 DATABASE

CONNECTIVITY ADO.NET

ADO.NET provides a bridge between the Front-End controls and the Back-End database. The ADO.NET object encapsulates all the data access operations and the controls interact with these objects to display, thus hiding the details of the movement of data

DATABASE CONNECTIVITY

ASP.NET allows the following sources of data to be accessed and used:

- Databases (e.g., Access, SQL Server, Oracle, MySQL)
- XML documents
- Business Objects
- Flat files

ASP.NET hides the complex processes of data access and provides much higher level of classes and objects through which data is accessed easily. These classes hide all complex coding for connection, data retrieving, data querying, and data manipulation.

ADO.NET is the technology that provides the bridge between various ASP.NET control objects and the backend data source. In this tutorial, we will look at data access and working with the data in brief.

Retrieve and display data

It takes two types of data controls to retrieve and display data in ASP.NET:

- **A data source control** - It manages the connection to the data, selection of data, and other jobs such as paging and caching of data etc.
- **A data view control** - It binds and displays the data and allows data manipulation. We will discuss the data binding and data source controls in detail later. In this section, we will use a SqlDataSource control to access data and a Grid View control to display and manipulate data in this chapter.

- We will also use an Access database, which contains the details about .Net books available in the market. Name of our database is ASPDotNetStepByStep.mdb and we will use the data table DotNetReferences.

The table has the following columns: ID, Title, AuthorFirstName, AuthorLastName, Topic, and Publisher.

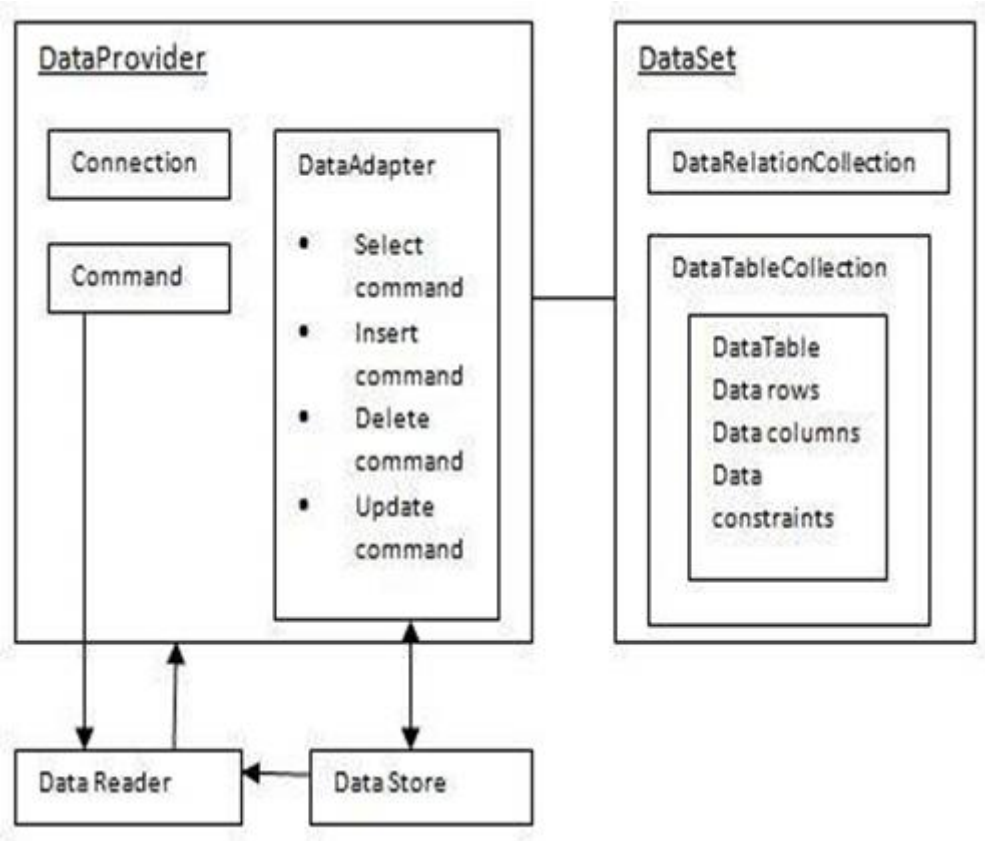
Let us directly move to action, take the following steps:

1. Create a web site and add a SqlDataSourceControl on the web form.
2. Click on the Configure Data Source option.
3. Click on the New Connection button to establish connection with a database.
4. Once the connection is set up, you may save it for further use. At the next step, you are asked to configure the select statement:
5. Select the columns and click next to complete the steps. Observe the WHERE, ORDER BY, and the Advanced buttons. These buttons allow you to provide the where clause, order by clause, and specify the insert, update, and delete commands of SQL respectively. This way, you can manipulate the data.
6. Add a GridView control on the form. Choose the data source and format the control using AutoFormat option.
7. After this the formatted GridView control displays the column headings, and the application is ready to execute.
8. Finally execute the application.

ADO.NET

ADO.NET provides a bridge between the front end controls and the back end database. The ADO.NET objects encapsulate all the data access operations and the controls interact with these objects to display data, thus hiding the details of movement of data.

The following figure shows the ADO.NET objects at a glance:



The DataSet Class

The dataset represents a subset of the database. It does not have a continuous connection to the database. To update the database a reconnection is required. The DataSet contains DataTable objects and DataRelation objects. The DataRelation objects represent the relationship between two tables.

Following table shows some important properties of the DataSet class:

Properties	Description
CaseSensitive	Indicates whether string comparisons within the data tables are case-sensitive.
Container	Gets the container for the component.

DataSetName	Gets or sets the name of the current data set.
DefaultViewManager	Returns a view of data in the data set.
DesignMode	Indicates whether the component is currently in design mode.
EnforceConstraints	Indicates whether constraint rules are followed when attempting any update operation.
Events	Gets the list of event handlers that are attached to this component.
ExtendedProperties	Gets the collection of customized user information associated with the DataSet.
HasErrors	Indicates if there are any errors.
IsInitialized	Indicates whether the DataSet is initialized.
Locale	Gets or sets the locale information used to compare strings within the table.
Namespace	Gets or sets the namespace of the DataSet.
Prefix	Gets or sets an XML prefix that aliases the namespace of the DataSet.
Relations	Returns the collection of DataRelation objects.
Tables	Returns the collection of DataTable objects.

The following table shows some important methods of the DataSet class:

Methods	**Description**
AcceptChanges	Accepts all changes made since the DataSet was loaded or this method was called.
BeginInit	Begins the initialization of the DataSet. The

	initialization occurs at run time.
Clear	Clears data.
Clone	Copies the structure of the DataSet, including all DataTable schemas, relations, and constraints. Does not copy any data.
Copy	Copies both structure and data.
CreateDataReader()	Returns a DataTableReader with one result set per DataTable, in the same sequence as the tables appear in the Tables collection.
CreateDataReader(DataTable[])	Returns a DataTableReader with one result set per DataTable.
EndInit	Ends the initialization of the data set.
Equals(Object)	Determines whether the specified Object is equal to the current Object.
Finalize	Free resources and perform other cleanups.
GetChanges	Returns a copy of the DataSet with all changes made since it was loaded or the AcceptChanges method was called.
GetChanges(DataRowState)	Gets a copy of DataSet with all changes made since it was loaded or the AcceptChanges method was called, filtered by DataRowState.
GetDataSetSchema	Gets a copy of XmlSchemaSet for the DataSet.
GetObjectData	Populates a serialization information object with the data needed to serialize the DataSet.
GetType	Gets the type of the current instance.
GetXML	Returns the XML representation of the data.

GetXMLSchema	Returns the XSD schema for the XML representation of the data.
HasChanges()	Gets a value indicating whether the DataSet has changes, including new, deleted, or modified rows.
HasChanges(DataRowState)	Gets a value indicating whether the DataSet has changes, including new, deleted, or modified rows, filtered by DataRowState.
IsBinarySerialized	Inspects the format of the serialized representation of the DataSet.
Load(IDataReader, LoadOption, DataTable[])	Fills a DataSet with values from a data source using the supplied IDataReader, using an array of DataTable instances to supply the schema and namespace information.
Load(IDataReader, LoadOption, String[])	Fills a DataSet with values from a data source using the supplied IDataReader, using an array of strings to supply the names for the tables within the DataSet.
Merge()	Merges the data with data from another DataSet. This method has different overloaded forms.
ReadXML()	Reads an XML schema and data into the DataSet. This method has different overloaded forms.
ReadXMLSchema(0)	Reads an XML schema into the DataSet. This method has different overloaded forms.
RejectChanges	Rolls back all changes made since the last

	call to AcceptChanges.
WriteXML()	Writes an XML schema and data from the DataSet. This method has different overloaded forms.
WriteXMLSchema()	Writes the structure of the DataSet as an XML schema. This method has different overloaded forms.

The Data Table Class

The Data Table class represents the tables in the database. It has the following important properties; most of these properties are read only properties except the Primary Key property:

Properties	**Description**
ChildRelations	Returns the collection of child relationship.
Columns	Returns the Columns collection.
Constraints	Returns the Constraints collection.
DataSet	Returns the parent DataSet.
DefaultView	Returns a view of the table.
ParentRelations	Returns the ParentRelations collection.
PrimaryKey	Gets or sets an array of columns as the primary key for the table.
Rows	Returns the Rows collection.

The following table shows some important methods of the Data Table class:

Methods	**Description**
AcceptChanges	Commits all changes since the last Accept Changes.

Clear	Clears all data from the table.
GetChanges	Returns a copy of the Data Table with all changes made since the Accept Changes method was called.
GetErrors	Returns an array of rows with errors.
ImportRows	Copies a new row into the table.
LoadDataRow	Finds and updates a specific row, or creates a new one, if not found any.
Merge	Merges the table with another Data Table.
NewRow	Creates a new Data Row.
RejectChanges	Rolls back all changes made since the last call to Accept Changes.
Reset	Resets the table to its original state.
Select	Returns an array of Data Row objects.

The DataRow Class

The DataRow object represents a row in a table. It has the following important properties:

Properties	**Description**
HasErrors	Indicates if there are any errors.
ItemArrays	Gets or sets all the values for the row.
Table	Returns the parent table.

The following table shows some important methods of the DataRow class:

Methods	Description
AcceptChanges	Accepts all changes made since this method was called.
BeginEdit	Begins edit operation.
CancelEdit	Cancels edit operation.
Delete	Deletes the Data Row.
EndEdit	Ends the edit operation.
GetChildRows	Gets the child rows of this row.
GetParentRow	Gets the parent row.
GetParentRows	Gets parent rows of Data Row object.
Reject Changes	Rolls back all changes made since the last call to Accept Changes.

The Data Adapter Object

The Data Adapter object acts as a mediator between the DataSet object and the database. This helps the Dataset to contain data from multiple databases or other data source.

The Data Reader Object

The Data Reader object is an alternative to the DataSet and Data Adapter combination. This object provides a connection oriented access to the data records in the database. These objects are suitable for read-only access, such as populating a list and then breaking the connection.

DB Command and DB Connection Objects

The DB Connection object represents a connection to the data source. The connection could be shared among different command objects.

The DB Command object represents the command or a stored procedure sent to the database from retrieving or manipulating data.

CHAPTER-4

SYSTEM DESIGN AND DEVELOPMENT PROCESS

4.1 DFD DIAGRAM

A data flow diagram is graphical tool used to describe and analyze movement of data through a system. These are the central tool and the basis from which the other components are developed. The transformation of data from input to output, through processed, may be described logically and independently of physical components associated with the system. These are known as the logical data flow diagrams. The physical data flow diagrams show the actual implements and movement of data between people, departments and workstations. A full description of a system actually consists of a set of data flow diagrams. The development of DFD'S is done in several levels. Each process in lower level diagrams can be broken down into a more detailed DFD in the next level

The idea behind the explosion of a process into more process is that understanding at one level of detail is exploded into greater detail at the next level. This is done until further explosion is necessary and an adequate amount of detail is described for analyst to understand the process.

3.2 DFD SYMBOLS

In the DFD, there are four symbols

- A square defines a source(originator) or destination of system data
- An arrow identifies data flow. It is the pipeline through which the information flows
- A circle or a bubble represents a process that transforms incoming data flow into outgoing data flows.
- An open rectangle is a data store, data at rest or a temporary repository of data

= Source or Destination of data

= Data Store

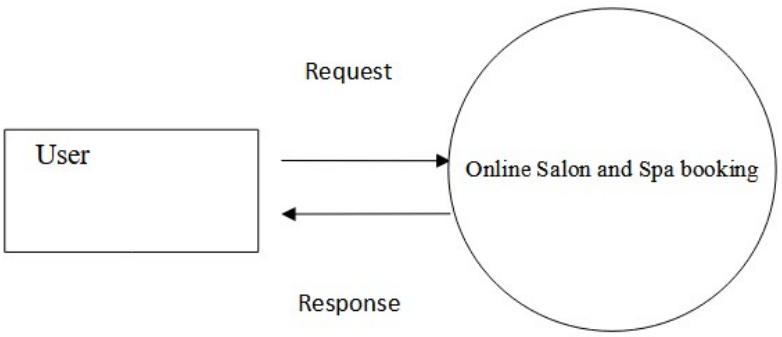
= Data Flow

= Process

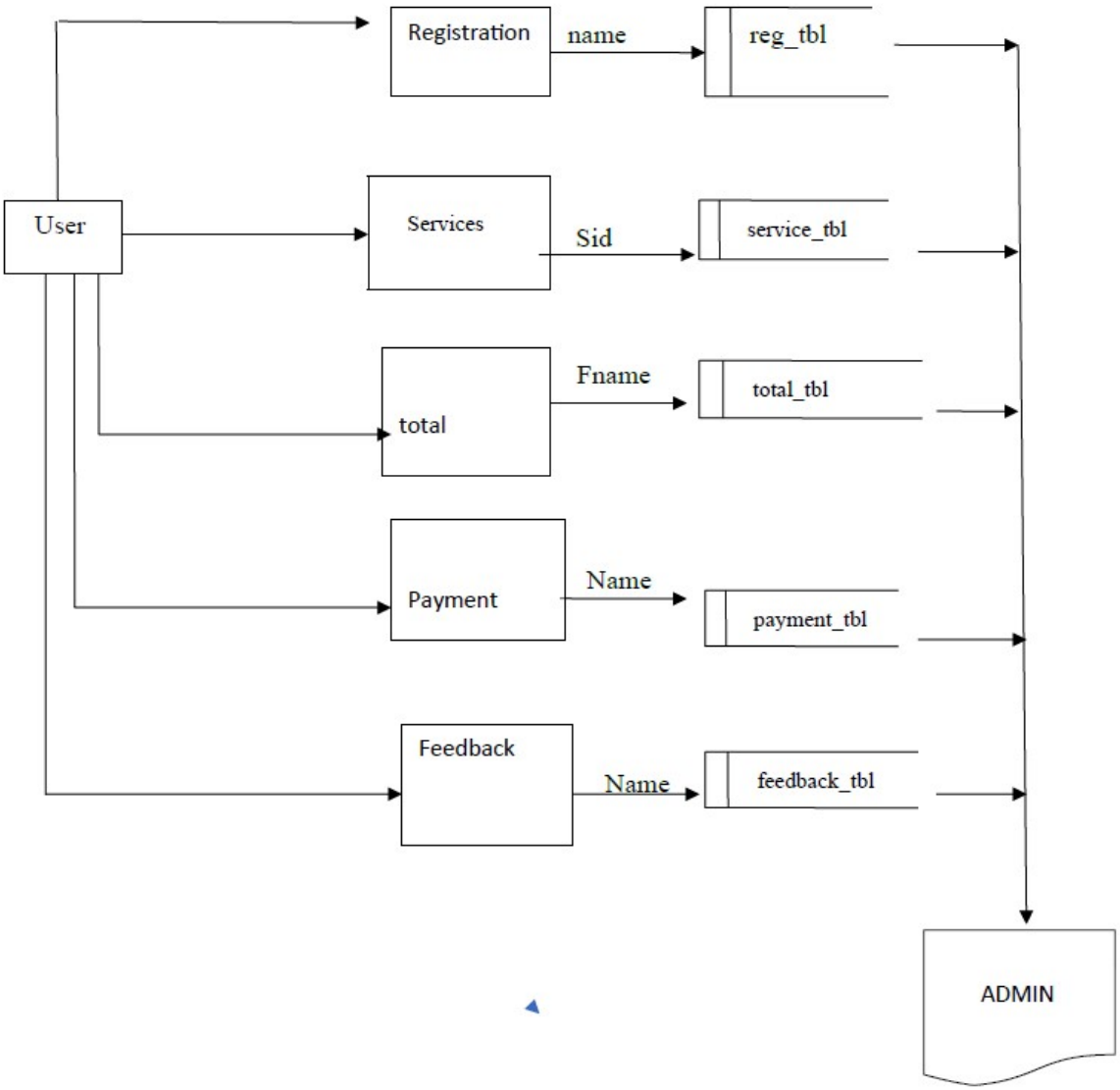
DFD'S

- The DFD shows flow of data, not of control loops and decision are controlled considerations do not appear on a DFD.
- The DFD does not indicate the time factor involved in any process whether the dataflow take place daily, weekly, monthly or yearly.

LEVEL 0 :



LEVEL 1


Registration
name
reg_tbl
User
Services
Sid
service_tbl
Fname
total_tbl
total
Payment
Name
payment_tbl
Feedback
Name
feedback_tbl
ADMIN

CHAPTER-5

DESIGN PROCESS

System design is a process of planning a new system to complement or all together replace the old system. The purpose of the design phase is to plan a solution for the problem. The phase is the first step in moving from the problem domain to the solution domain. The design of the system is the critical aspect that affects the quality of the software. System design is also called top level design. The design phase translates the logical aspects of the system into physical aspects of the system.

The system design consists of:

- Input Design
- Database Design
- Output Design

5.1. INPUT DESIGN

Input Design is the process of converting a user oriented description of the inputs to a computer-based business system into a programmer-oriented specification. Input data were found to be available for establishing and maintaining master and transaction files and for creating output records.

The most suitable types of input media, for either off-line or on-line devices, where selected after a study of alternative data capture techniques. The field length must be documented.

The Sequence of field should match the sequence of the field on the source document. The data format must be identified to the data entry operator. VDT-Visual Display Terminals. Effective screen design can not only reduce errors but can increase productivity and user satisfaction. Data entered is displayed on the screen.

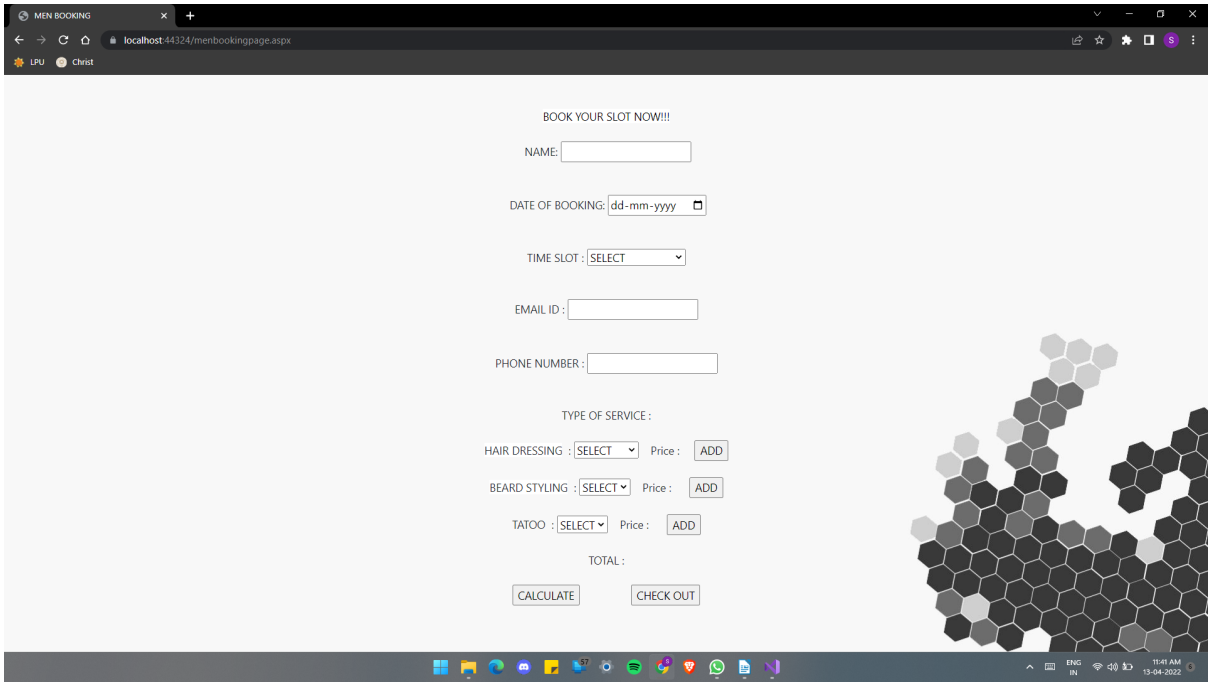
Before sending the data to the computer system, modification can be made. Each display station has its own memory called buffer for storing data. Common size display screen is 24 rows of 80 characters each.

Keeping in view the above description of the input types and input media, it can be said that most of the inputs are of the form of internal and interactive. As Input data is to be the directly keyed in by the user, the keyboard can be considered to be the most suitable input devices

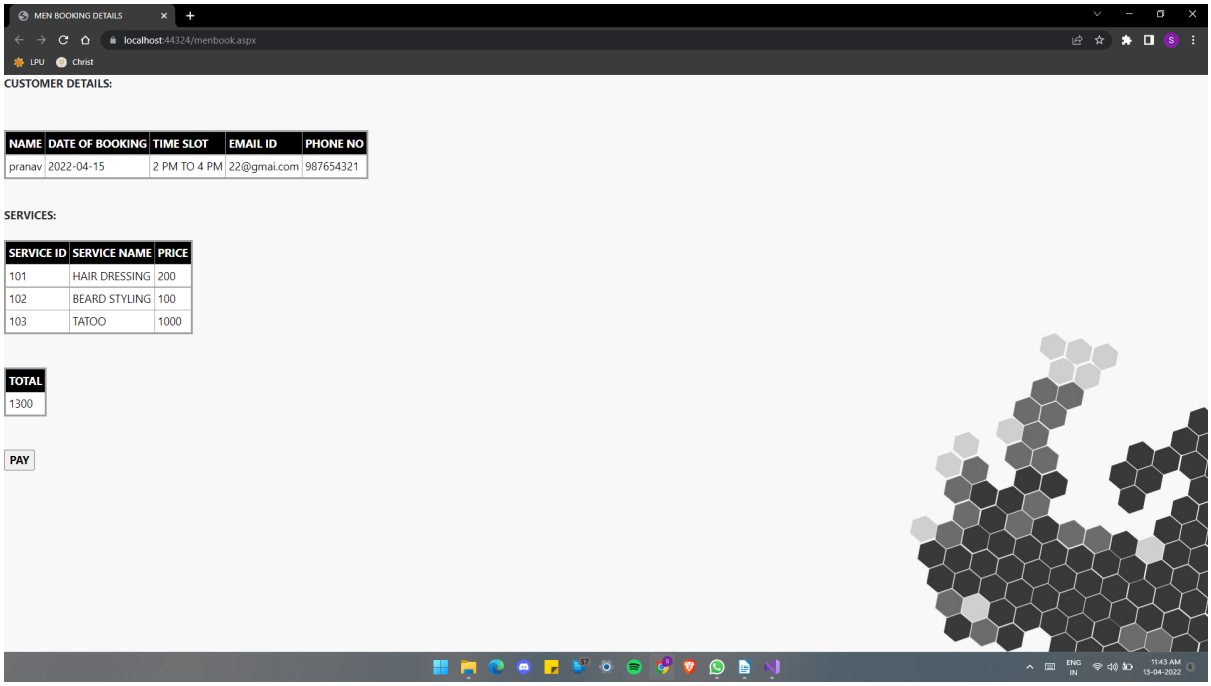
Input Design in the project are :

- Customer Details and Booking Form Design
- Payment Summary Form Design
- Payment Form Design
- Admin Form Design
- Feedback Details Form Design

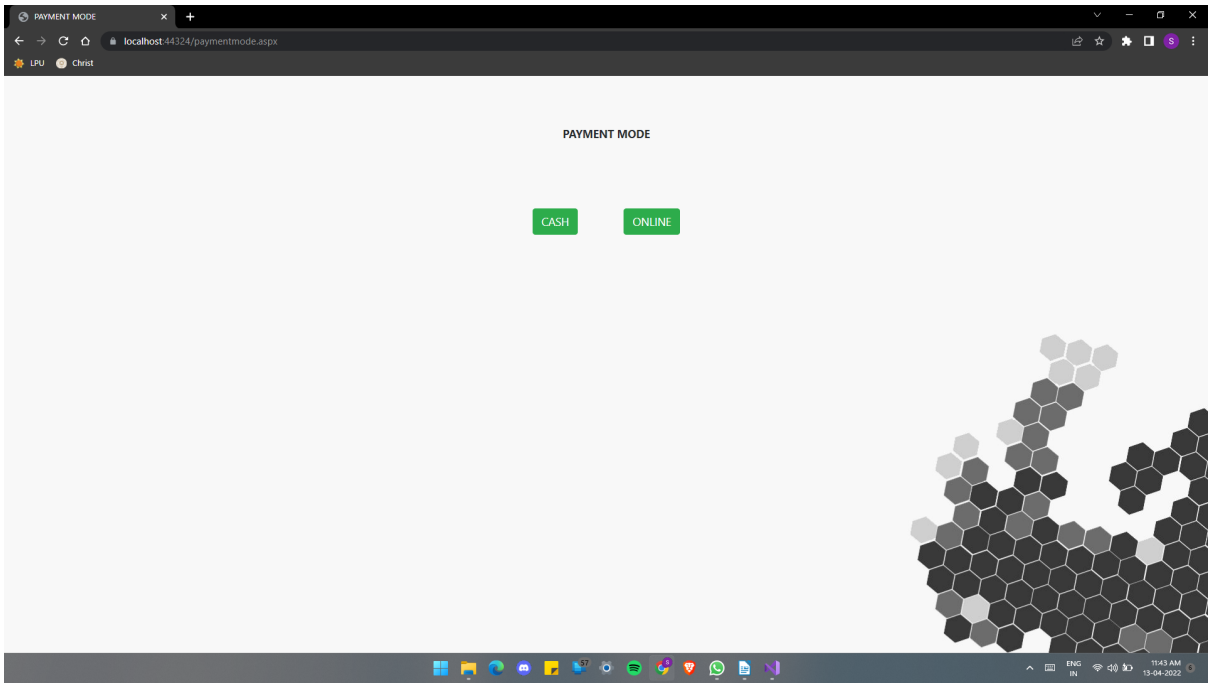
CUSTOMER AND BOOKING FORM :

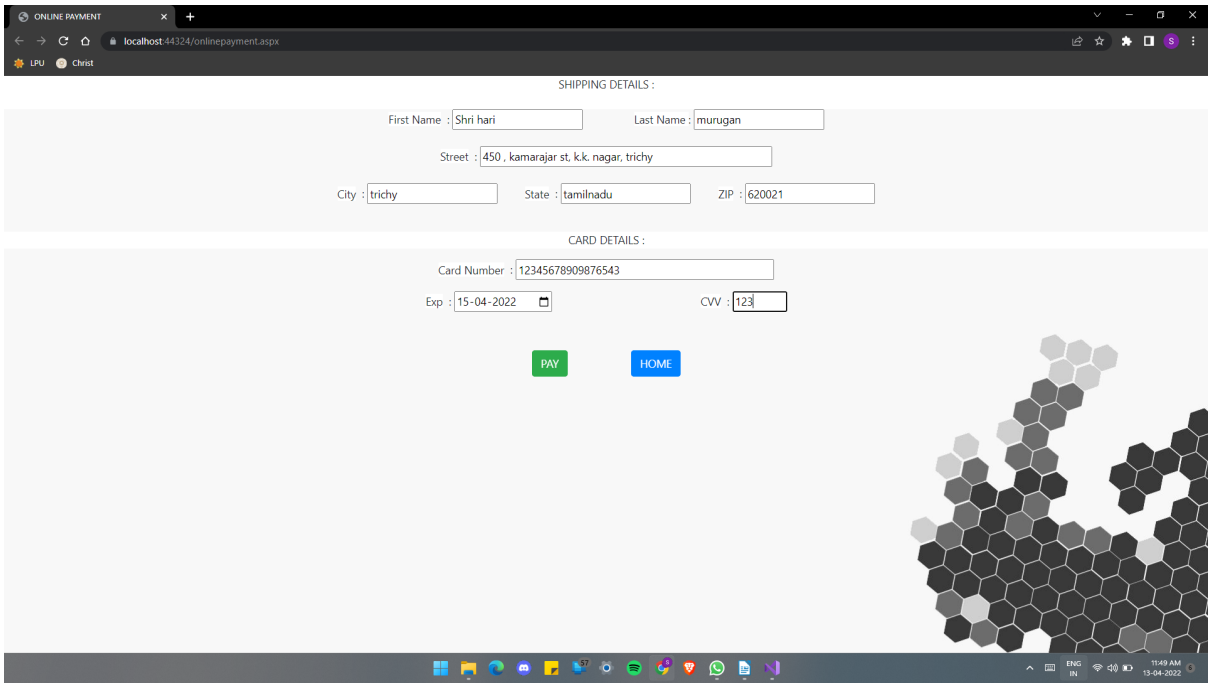


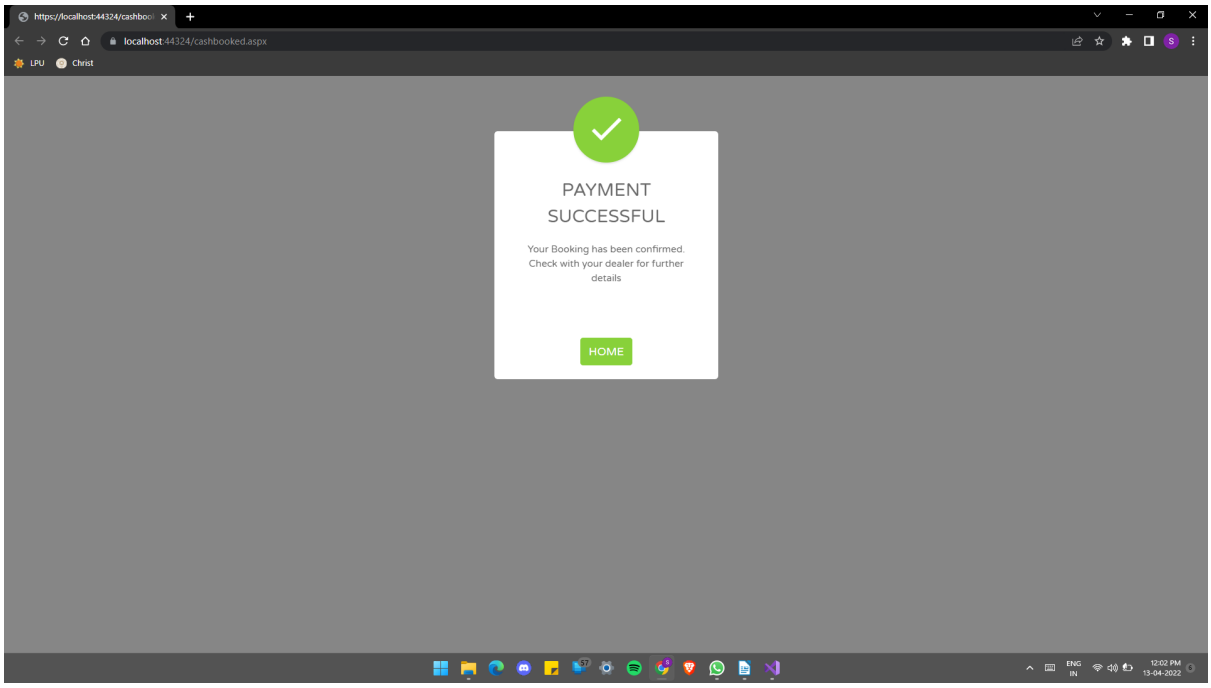
PAYMENT SUMMARY FORM :



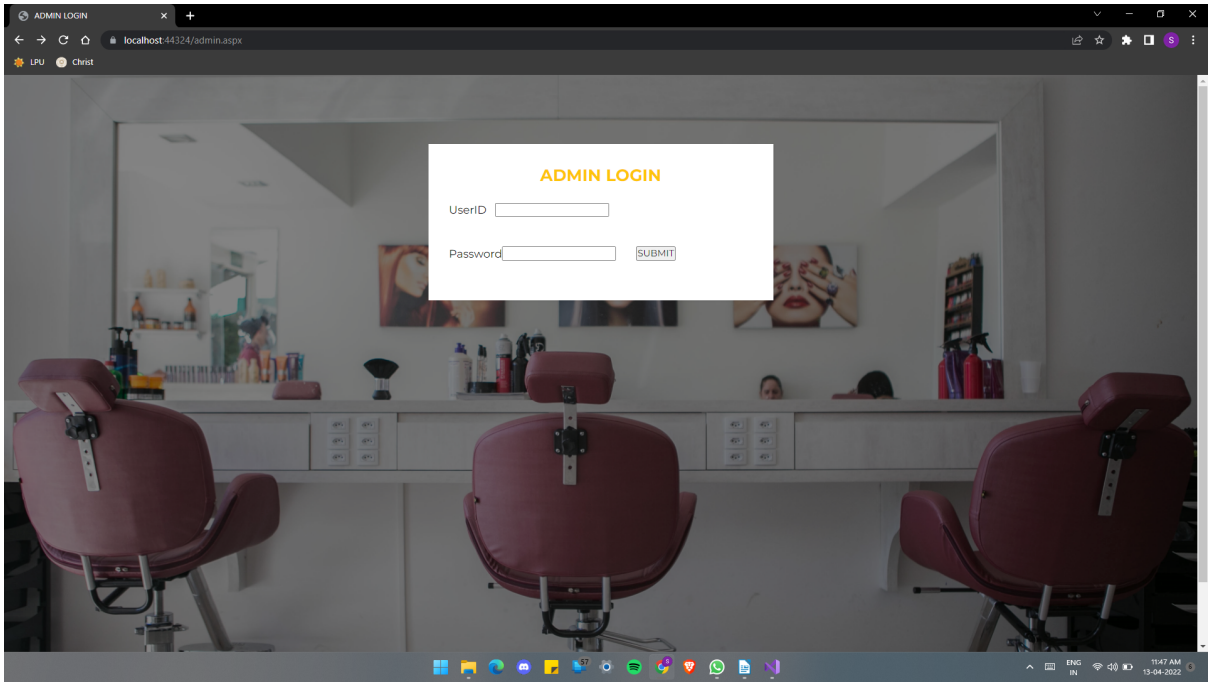
PAYMENT FORM :

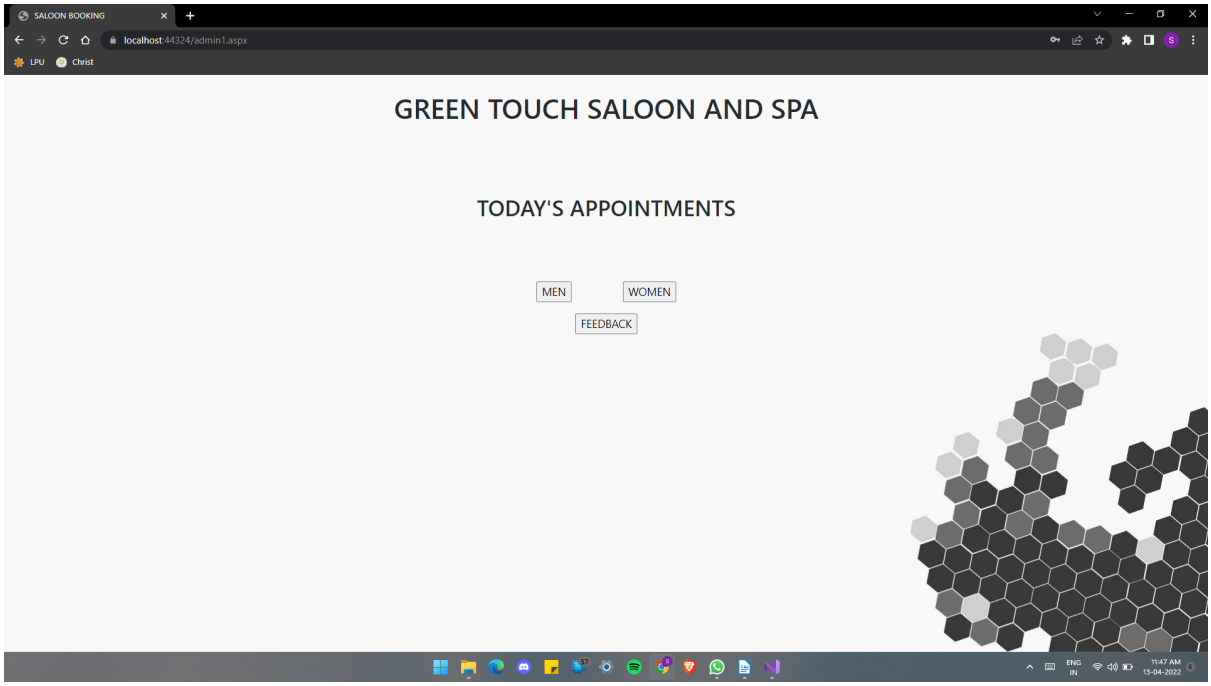


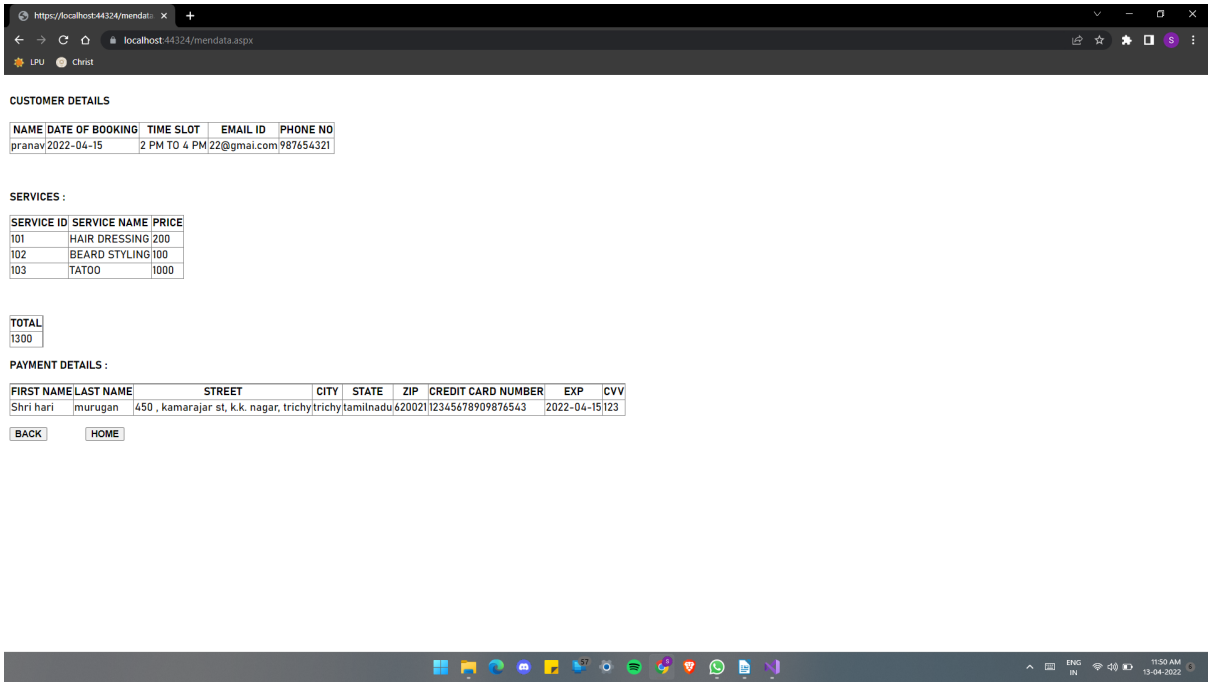

ONLINE PAYMENT
localhost:44324/onlinepayment.aspx
LPU
Christ
SHIPPING DETAILS :
First Name : Shri hari
Last Name : murugan
Street : 450 , kamarajar st, k.k. nagar, trichy
City : trichy
State : tamilnadu
ZIP : 620021
CARD DETAILS :
Card Number : 12345678909876543
Exp : 15-04-2022
CVV : 123
PAY
HOME


localhost:44324/cashbooked.aspx
LPU
Christ
PAYMENT SUCCESSFUL
Your Booking has been confirmed. Check with your dealer for further details
HOME

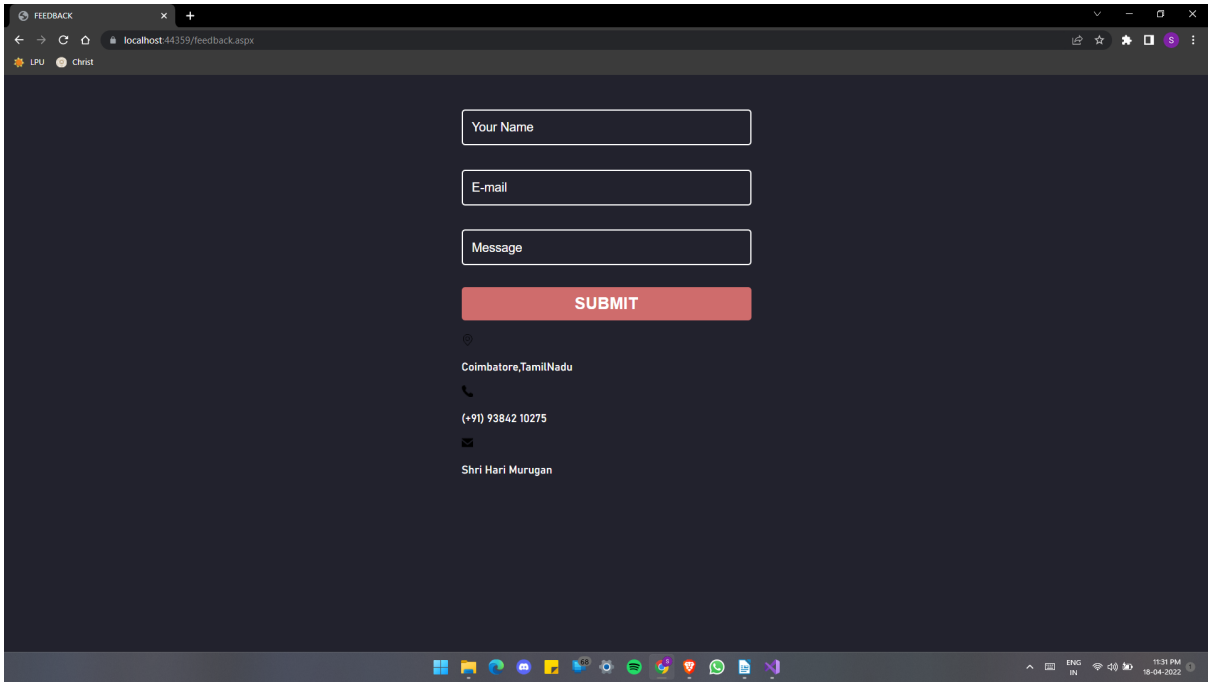
ADMIN FORM :


ADMIN LOGIN
localhost:44324/admin.aspx
LPU
Christ
ADMIN LOGIN
UserID
Password
SUBMIT


SALOON BOOKING
localhost:44324/admin1.aspx
LPU
Christ
GREEN TOUCH SALOON AND SPA
TODAY'S APPOINTMENTS
MEN
WOMEN
FEEDBACK



FEEDBACK BACK PAGE:



5.2. DATABASE DESIGN

Database design is the process of producing a detailed data model of a 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.

The term database design can be used to describe many different parts of the design of an overall database system. Principally, and most correctly, it can be thought of as the logical design of the base data structures used to store the data. In the relational model these are the tables and views. In an object database the entities and relationships map directly to object classes and named relationships. However, the term database design could also be used to apply to the overall process of designing, not just the base data structures, but also the forms and queries used as part of the overall database application within the database management system.

The process of doing database design generally consists of a number of steps which will be carried out by the database designer. Usually, the designer must:

- Determine the data to be stored in the database.
- Determine the relationships between the different data elements.
- Superimpose a logical structure upon the data on the basis of these relationships.

OBJECTIVES

- Ease of use
- Control of data integrity
- Control of redundancy
- Control of security
- System performance
- System compatibility

Table Name: MEN_Details

Primary Key: NAME

Update Script File: dbo.Mendetails.sql

Name	Data Type	Allow Nulls	Default
NAME	varchar(50)	☐	
DATE OF BOOKING	varchar(50)	☐	
TIME SLOT	varchar(50)	☐	
EMAIL	varchar(50)	☐	
PHONE NUMBER	varchar(50)	☐	
		☐	

Table Name: Service

Primary Key: sid

Update Script File: dbo.Servicesmen.sql

Name	Data Type	Allow Nulls	Default
SERVICE ID	varchar(50)	☐	
SERVICE NAME	varchar(50)	☐	
PRICE	varchar(50)	☐	
		☐	

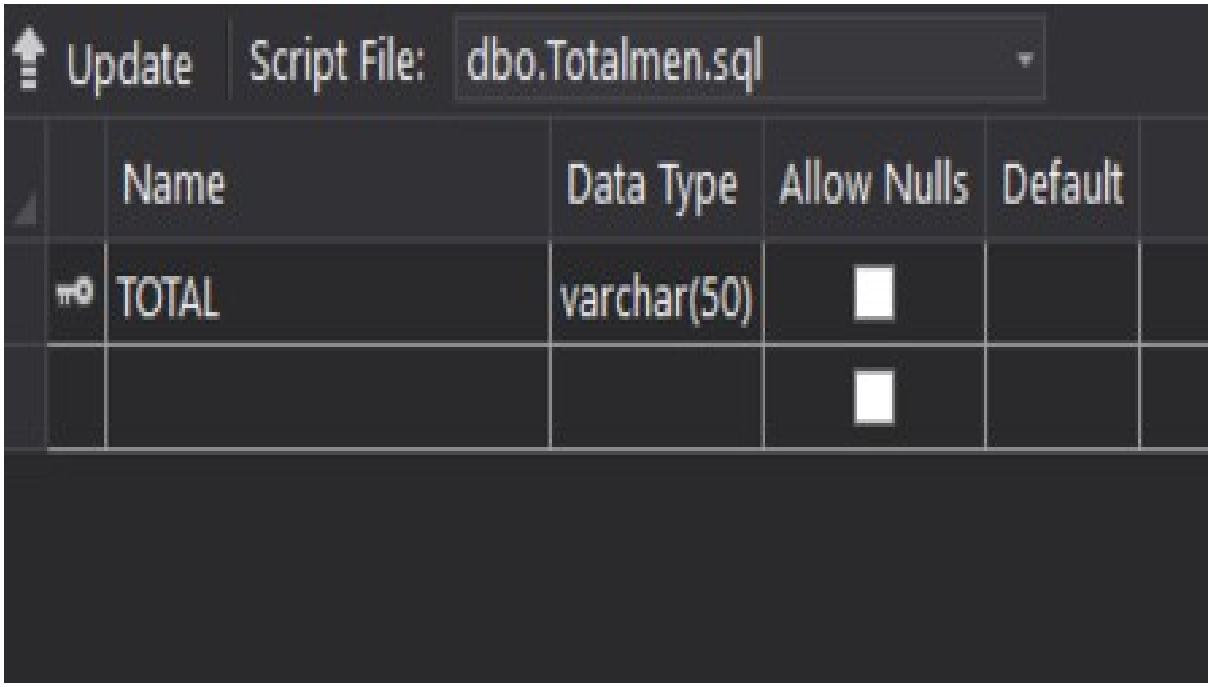
Table Name: payment

Primary Key: fname

Name	Data Type	Allow Nulls	Default
FIRST NAME	varchar(50)	☐	
LAST NAME	varchar(50)	☐	
STATE	varchar(50)	☐	
CITY	varchar(50)	☐	
PIN CODE	varchar(50)	☐	
CARD NUMBER	varchar(50)	☐	
CARD NAME	varchar(50)	☐	
EXPIRE	varchar(50)	☐	
CVV	varchar(50)	☐	
		☐	

Table Name : total

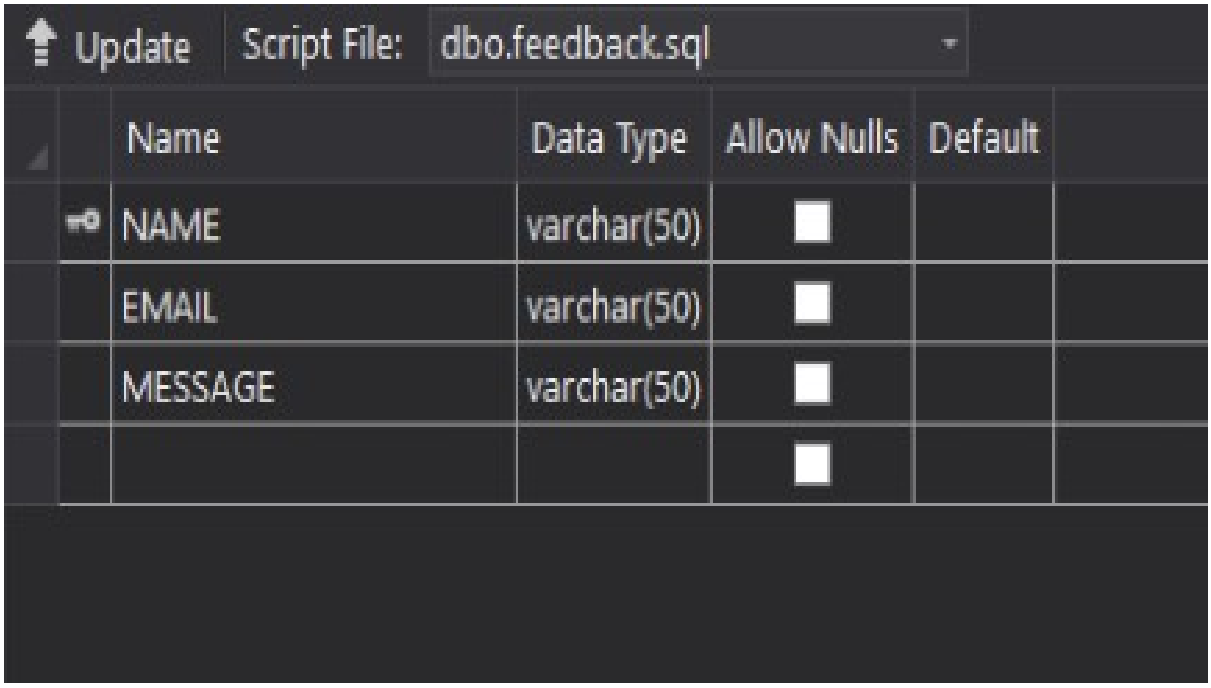
Primary key: TOTAL



Name	Data Type	Allow Nulls	Default
TOTAL	varchar(50)	☐	
		☐	

Table Name : feedback

Primary key: Name



5.3 OUTPUT DESIGN

To give you a head start, the ASP.NET source code for each pattern is provided in 2 forms: structural and real-world. Structural code uses type names as defined in the pattern definition and UML diagrams. Real-world code provides real-world programming situations where you may use these patterns.

A third form, C# optimized, demonstrates design patterns that fully exploit built-in .NET 4.5 features, such as, generics, attributes, delegates, reflection, and more. These and much more are available in our .NET Design Pattern Framework 4.5 You can see the Singleton page for a .NET 4.5 Optimized.

CHAPTER-6

SYSTEM TESTING AND IMPLEMENTATION

6.1 TESTING METHODOLOGIES

Testing is the process of evaluating a system or its component(s) with the intent to find whether it satisfies the specified requirements or not. In simple words, testing is executing a system in order to identify any gaps, errors, or missing requirements in contrary to the actual requirements.

According to ANSI/IEEE 1059 standard, Testing can be defined as - A process of analyzing a software item to detect the differences between existing and required conditions (that is defects/errors/bugs) and to evaluate the features of the software item.

Who does Testing?

It depends on the process and the associated stakeholders of the project(s). In the IT industry, large companies have a team with responsibilities to evaluate the developed software in context of the given requirements. Moreover, developers also conduct testing which is called **Unit Testing**. In most cases, the following professionals are involved in testing a system within their respective capacities:

- Software Tester
- Software Developer
- Project Lead/Manager
- End User

Different companies have different designations for people who test the software on the basis of their experience and knowledge such as Software Tester, Software Quality Assurance Engineer, QA Analyst, etc.

It is not possible to test the software at any time during its cycle. The next two sections state when testing should be started and when to end it during the SDLC.

When to Start Testing?

An early start to testing reduces the cost and time to rework and produce error-free software that is delivered to the client. However in Software Development Life Cycle (SDLC), testing can be started from the Requirements Gathering phase and continued till the deployment of the software. It also depends on the development model that is being used. For example, in the Waterfall model, formal testing is conducted in the testing phase; but in the incremental model, testing is performed at the end of every increment/iteration and the whole application is tested at the end.

Testing is done in different forms at every phase of SDLC:

- ❖ During the requirement gathering phase, the analysis and verification of requirements are also considered as testing.
- ❖ Reviewing the design in the design phase with the intent to improve the design is also considered as testing.
- ❖ Testing performed by a developer on completion of the code is also categorized as testing.

When to Stop Testing?

It is difficult to determine when to stop testing, as testing is a never-ending process and no one can claim that a software is 100% tested. The following aspects are to be considered for stopping the testing process:

- Testing Deadlines
- Completion of test case execution
- Completion of functional and code coverage to a certain point
- Bug rate falls below a certain level and no high-priority bugs are identified
- Management decision

Verification & Validation

These two terms are very confusing for most people, who use them interchangeably. The following table highlights the differences between verification and validation.

S.NO	VERIFICATION	VALIDATION
1	Verification addresses the concern: "Are you building it right	Validation addresses the concern: "Are you building the right thing?"
2	Ensures that the software system meets all the functionality.	Ensures that the functionalities meet the intended behavior.
3	Verification takes place first and includes the checking documentation, code, etc.	Validation occurs after verification for and mainly involves the checking of the overall product
4	Done by developers.	Done by testers.
5	It has static activities, as it includes collecting reviews, walkthroughs, and inspections to verify software.	It has dynamic activities, as it includes executing the software against the requirements.
6	It is an objective process and no subjective decision should be needed to verify software.	It is a subjective process and involves subjective decisions on how well a software works.

Manual Testing

Manual testing includes testing a software manually, i.e., without using any automated tool or any script. In this type, the tester takes over the role of an end-user and tests the software to identify any unexpected behavior or bug. There are different stages for manual testing such as unit testing, integration testing, system testing, and user acceptance testing.

Testers use test plans, test cases, or test scenarios to test a software to ensure the completeness of testing. Manual testing also includes exploratory testing, as testers explore the software to identify errors in it.

Automation Testing

Automation testing, which is also known as Test Automation, is when the tester writes scripts and uses software to test the product. This process involves automation of a manual process. Automation Testing is used to re-run the test scenarios that were performed manually, quickly, and repeatedly.

Apart from regression testing, automation testing is also used to test the application from load, performance, and stress point of view. It increases the test coverage, improves accuracy, and saves time and money in comparison to manual testing.

What is Automate?

It is not possible to automate everything in a software. The areas at which a user can make transactions such as the login form or registration forms, any area where large number of users can access the software simultaneously should be automated.

Furthermore, all GUI items, connections with databases, field validations, etc. can be efficiently tested by automating the manual process.

When to Automate?

Test Automation should be used by considering the following aspects of software:

- Large and critical projects
- Projects that require testing the same areas frequently
- Requirements not changing frequently
- Accessing the application for load and performance with many virtual users
- Stable software with respect to manual testing
- Availability of time

How to Automate?

Automation is done by using a supportive computer language like VB scripting and an automated software application. There are many tools available that can be used to write automation scripts. Before mentioning the tools, let us identify the process that can be used to automate the testing process:

- Identifying areas within a software for automation
- Selection of appropriate tool for test automation
- Writing test scripts
- Development of test suits
- Execution of scripts
- Create result reports
- Identify any potential bug or performance issues

Software Testing Tools

The following tools can be used for automation testing:

- HP Quick Test Professional

- Selenium
- IBM Rational Functional Tester
- SilkTest
- TestComplete
- Testing Anywhere
- WinRunner
- LoadRunner
- Visual Studio Test Professional
- WATIR

There are different methods that can be used for software testing. This chapter briefly describes the methods available.

Black-Box Testing

The technique of testing without having any knowledge of the interior workings of the application is called black-box testing. The tester is oblivious to the system architecture and does not have access to the source code. Typically, while performing a black-box test, a tester will interact with the system's user interface by providing inputs and examining outputs without knowing how and where the inputs are worked upon.

The following table lists the advantages and disadvantages of black-box testing.

ADVANTAGES	DISADVANTAGES
Well suited and efficient for large code segments.	Limited coverage, since only a selected number of test scenarios is actually performed.
Code access is not required.	Inefficient testing, due to the fact that the tester only has limited knowledge about an application.

Clearly separates user's perspective from the developer's perspective through visibly defined roles.	Blind coverage, since the tester cannot target specific code segments or error-prone areas.
Large numbers of moderately skilled testers can test the application with no knowledge of implementation, programming language, or OS	

White-Box Testing

White-box testing is the detailed investigation of internal logic and structure of the code. White-box testing is also called **glass testing** or **open-box testing**. In order to perform **white-box** testing on an application, a tester needs to know the internal workings of the code.

The tester needs to have a look inside the source code and find out which unit/chunk of the code is behaving inappropriately.

The following table lists the advantages and disadvantages of white-box testing.

Advantages	**Disadvantages**
• As the tester has knowledge of the source code, it becomes very easy to find out which type of data can help in testing the application effectively.	• Due to the fact that a skilled tester is needed to perform white-box testing, the costs are increased.
• It helps in optimizing the code.	• Sometimes it is impossible to look into every nook and

	corner to find out hidden errors that may create problems, as many paths will go untested.
• Extra lines of code can be removed which can bring in hidden defects.	• It is difficult to maintain white-box testing, as it requires specialized tools like code analysers and debugging tools.
• Due to the tester's knowledge about the code, maximum coverage is attained during test scenario writing.	

Grey-Box Testing

Grey-box testing is a technique to test the application with having a limited knowledge of the internal workings of an application. In software testing, the phrase the more you know, the better carries a lot of weight while testing an application.

Mastering the domain of a system always gives the tester an edge over someone with limited domain knowledge. Unlike black-box testing, where the tester only tests the application's user interface; in grey-box testing, the tester has access to design documents and the database. Having this knowledge, a tester can prepare better test data and test scenarios while making a test plan.

Advantages	Disadvantages
• Offers combined benefits of black-box and white-box testing wherever possible.	• Since the access to source code is not available, the ability to go over the code and test coverage is limited.
• Grey box testers don't rely on the source code; instead they rely on interface definition and functional specifications.	• The tests can be redundant if the software designer has already run a test case.
• Based on the limited information available, a grey-box tester can design excellent test scenarios especially around	• Testing every possible input stream is unrealistic because it would take an unreasonable amount of time; therefore, many
• Communication protocols and data type handling.	• Program paths will go untested.
• The test is done from the point of view of the user and not the designer.	

A Comparison of Testing Methods

The following table lists the points that differentiate black-box testing, grey-box testing, and white-box testing.

Black-Box Testing	Grey-Box Testing	White-Box Testing
The internal workings of an application need not be known.	The tester has limited knowledge of the internal workings of the application.	Tester has full knowledge of the internal workings of the application.
Also known as closed-box testing, data-driven testing, or functional testing.	Also known as translucent testing, as the tester has limited knowledge of the insides of the application.	Also known as clear-box testing, structural testing, or code-based testing.
Performed by end-users and also by testers and developers.	Performed by end-users and also by testers and developers.	Normally done by testers and developers.
Testing is based on external expectations - Internal behaviour of the application is unknown.	Testing is done on the basis of high-level database diagrams and data flow diagrams.	Internal workings are fully known and the tester can design test data accordingly.
It is exhaustive and the least time-consuming.	Partly time-consuming and exhaustive.	The most exhaustive and time-consuming type of testing.
Not suited for algorithm testing.	Not suited for algorithm testing.	Suited for algorithm testing.
This can only be done by trial-and-error method.	Data domains and internal boundaries can be tested, if known.	Data domains and internal boundaries can be better tested.

6.2 IMPLEMENTATION

In this chapter, we will study about programming methods, documentation and challenges in software implementation.

Structured Programming

In the process of coding, the lines of code keep multiplying, thus, size of the software increases. Gradually, it becomes next to impossible to remember the flow of program. If one forgets how software and its underlying programs, files, procedures are constructed it then becomes very difficult to share, debug and modify the program. The solution to this is structured programming. It encourages the developer to use subroutines and loops instead of using simple jumps in the code, thereby bringing clarity in the code and improving its efficiency Structured programming also helps programmer to reduce coding time and organize code properly.

Structured programming states how the program shall be coded. Structured programming uses three main concepts:

- **Top-down analysis** - software is always made to perform some rational work. This rational work is known as problem in the software parlance. Thus it is very important that we understand how to solve the problem. Under top-down analysis, the problem is broken down into small pieces where each one has some significance. Each problem is individually solved and steps are clearly stated about how to solve the problem.

- **Modular Programming** - While programming, the code is broken down into smaller group of instructions. These groups are known as modules, subprograms or subroutines. Modular programming based on the understanding of top-down analysis. It discourages jumps using ‘goto’ statements in the program, which often makes the program flow non-traceable. Jumps are prohibited and modular format is encouraged in structured programming.

- **Structured Coding** - In reference with top-down analysis, structured coding sub-divides the modules into further smaller units of code in the order of their execution. Structured programming uses control structure, which controls the flow of the program, whereas structured coding uses control structure to organize its instructions in definable patterns.

Functional Programming

Functional programming is style of programming language, which uses the concepts of mathematical functions. A function in mathematics should always produce the same result on receiving the same argument. In procedural languages, the flow of the program runs through procedures, i.e. the control of program is transferred to the called procedure. While control flow is transferring from one procedure to another, the program changes its state.

In procedural programming, it is possible for a procedure to produce different results when it is called with the same argument, as the program itself can be in different state while calling it. This is a property as well as a drawback of procedural programming, in which the sequence or timing of the procedure execution becomes important.

Functional programming provides means of computation as mathematical functions, which produces results irrespective of program state. This makes it possible to predict the behaviour of the program.

Functional programming uses the following concepts:

- **First class and High-order functions** - These functions have capability to accept another function as argument or they return other functions as results.
- **Pure functions** - These functions do not include destructive updates, that is, they do not affect any I/O or memory and if they are not in use, they can easily be removed without hampering the rest of the program.
- **Recursion** - Recursion is a programming technique where a function calls itself and repeats the program code in it unless some pre-defined condition matches. Recursion is the way of creating loops in functional programming.

- **Strict evaluation** - It is a method of evaluating the expression passed to a function as an argument. Functional programming has two types of evaluation methods, strict (eager) or non-strict (lazy). Strict evaluation always evaluates the expression before invoking the function. Non-strict evaluation does not evaluate the expression unless it is needed.
- **λ-calculus** - Most functional programming languages use λ-calculus as their type systems. λ-expressions are executed by evaluating them as they occur.

Programming style

Programming style is set of coding rules followed by all the programmers to write the code. When multiple programmers work on the same software project, they frequently need to work with the program code written by some other developer. This becomes tedious or at times impossible, if all developers do not follow some standard programming style to code the program.

An appropriate programming style includes using function and variable names relevant to the intended task, using well-placed indentation, commenting code for the convenience of reader and overall presentation of code. This makes the program code readable and understandable by all, which in turn makes debugging and error solving easier. Also, proper coding style helps ease the documentation and updation.

Coding Guidelines

Practice of coding style varies with organizations, operating systems and language of coding itself.

The following coding elements may be defined under coding guidelines of an organization:

- **Naming conventions** - This section defines how to name functions, variables, constants and global variables.
- **Indenting** - This is the space left at the beginning of line, usually 2-8 whitespace or single tab.
- **Whitespace** - It is generally omitted at the end of line.

- **Operators** - Defines the rules of writing mathematical, assignment and logical operators. For example, assignment operator '=' should have space before and after it, as in "x = 2".
- **Control Structures** - The rules of writing if-then-else, case-switch, while-until and for control flow statements solely and in nested fashion.
- **Line length and wrapping** - Defines how many characters should be there in one line, mostly a line is 80 characters long. Wrapping defines how a line should be wrapped, if is too long.
- **Functions** - This defines how functions should be declared and invoked, with and without parameters.
- **Variables** - This mentions how variables of different data types are declared and define

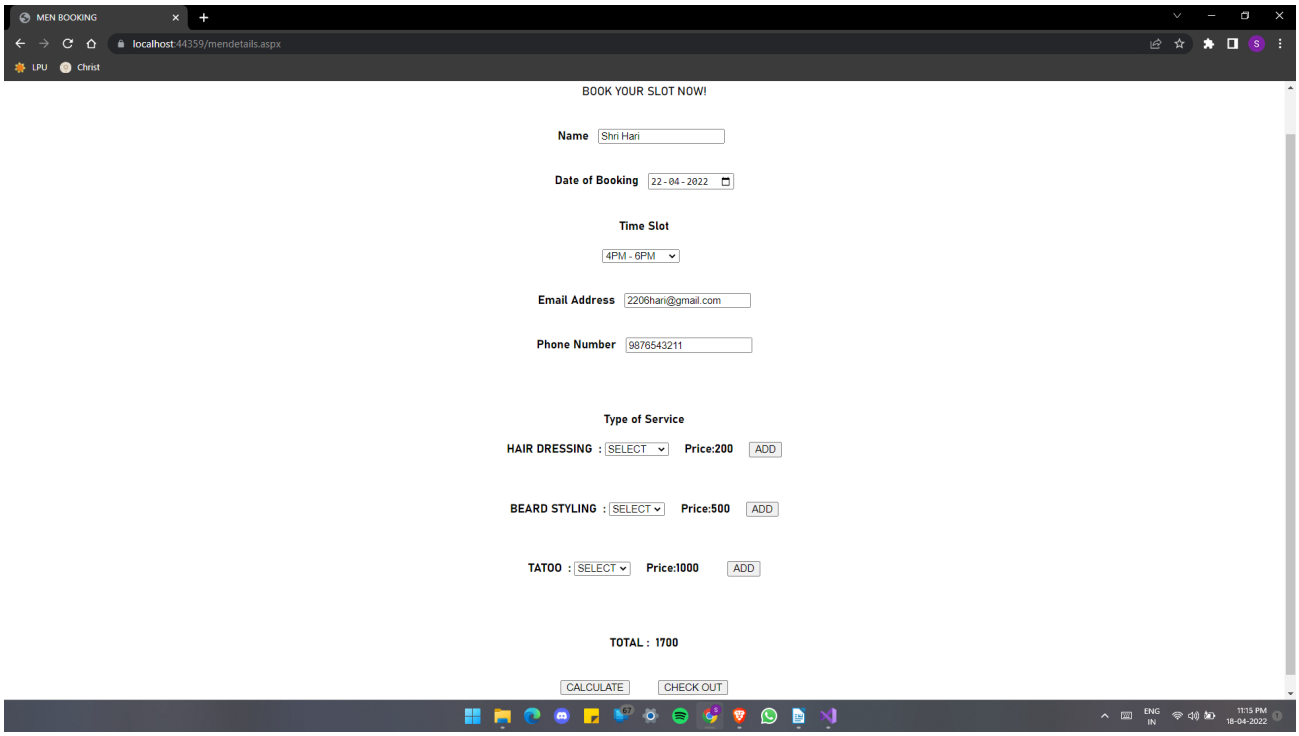
CHAPTER-7

CONCLUSION

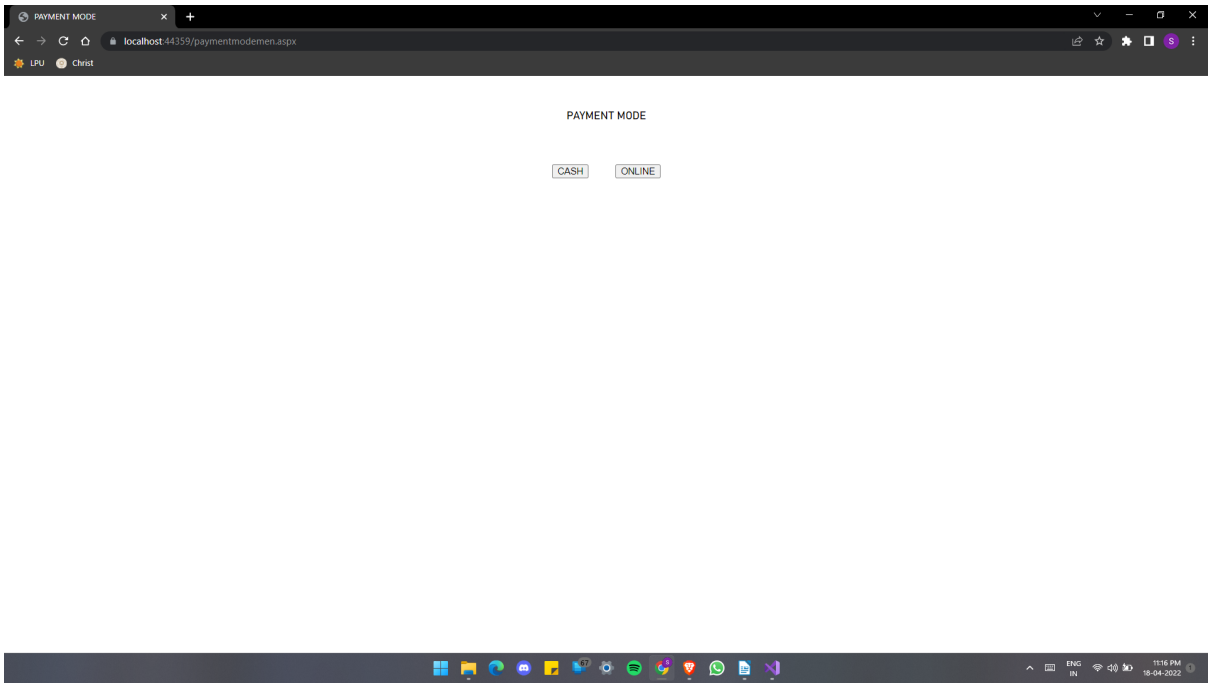
The system is "**ONLINE SALON AND SPA BOOKING SYSTEM**", it has a registration Form where the user can use the registration interface. The types of service details are shown in registration page. Once the user choose their services they needed, now the user is ready to pay. In the next form the checkout page it shows the user details and services that user selected. The next form is the payment page, here user gives their card details to further pay online. Also a form to get user's feedback to improve the services. This system also contains admin page which has all the data for men and women and feedback given by users.

FORM SCREENSHOTS

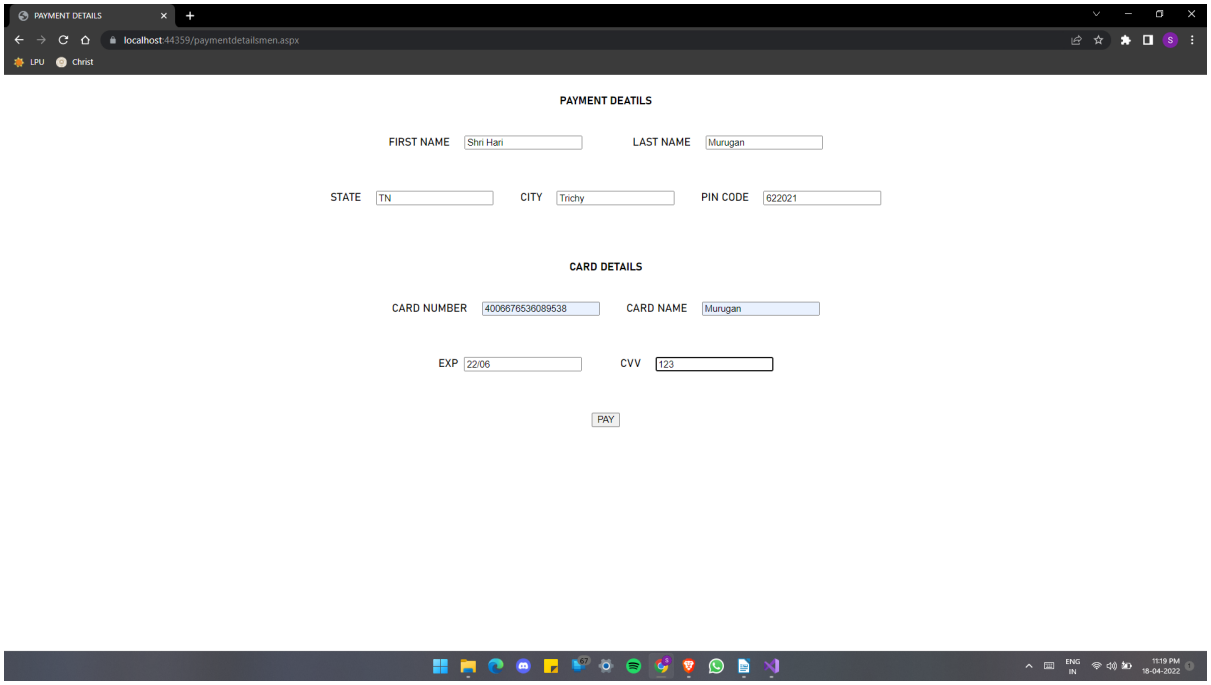
FORM NAME: REGISTRATION



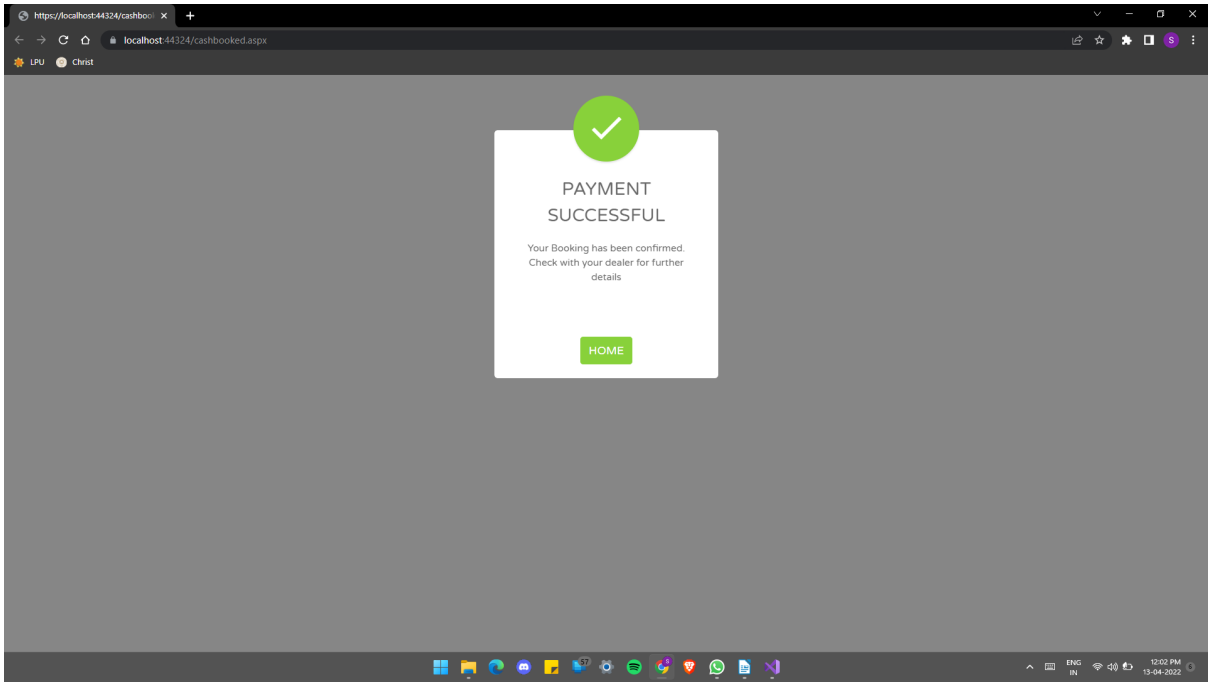
FORM NAME: PAYMENT MODE



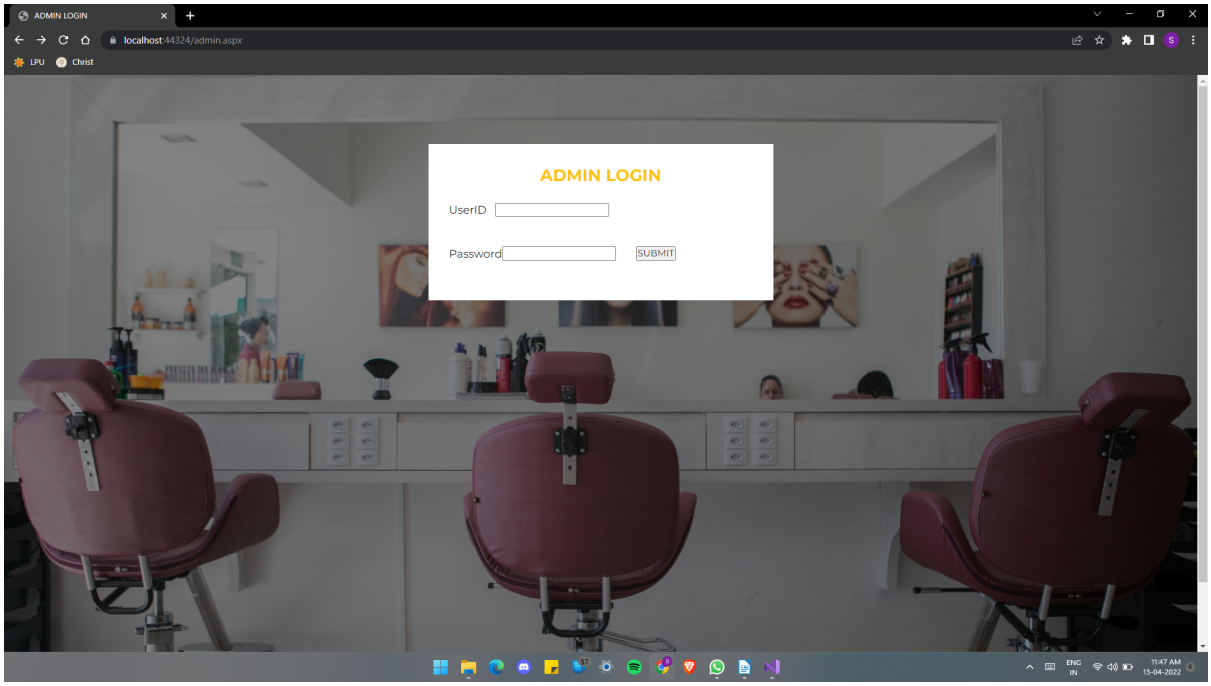
FORM NAME: ONLINE PAYMENT



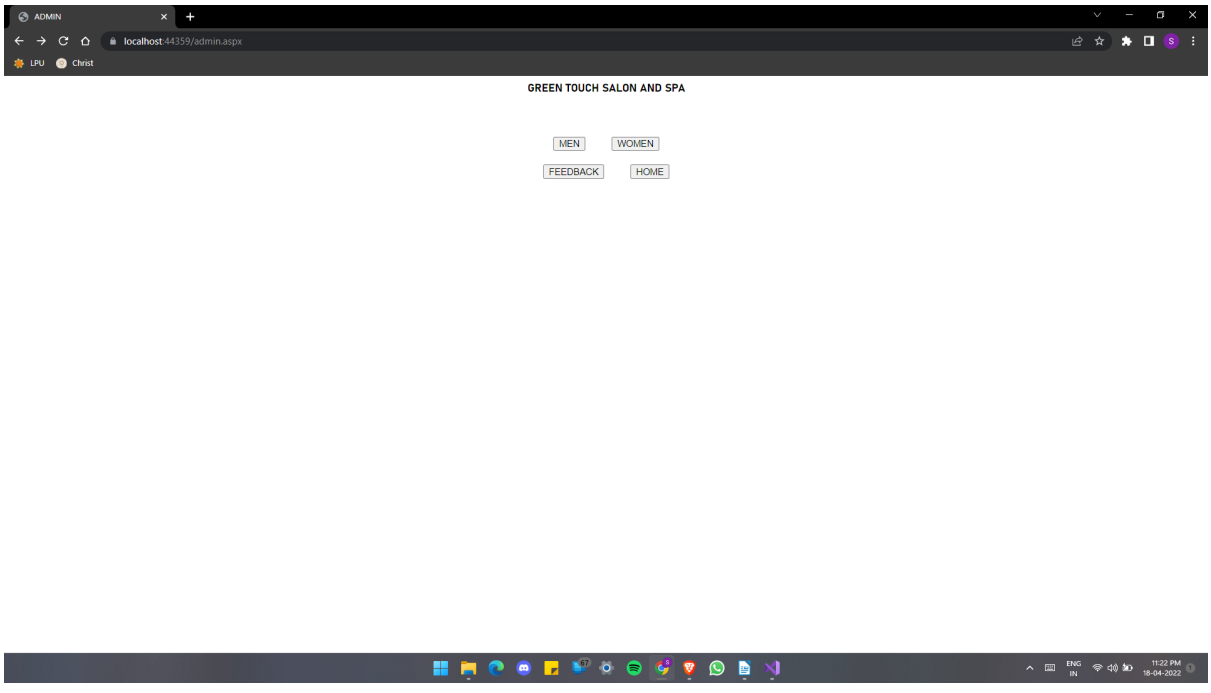
FORM NAME: PAYMENT SUCCESS (CASH AND ONLINE)



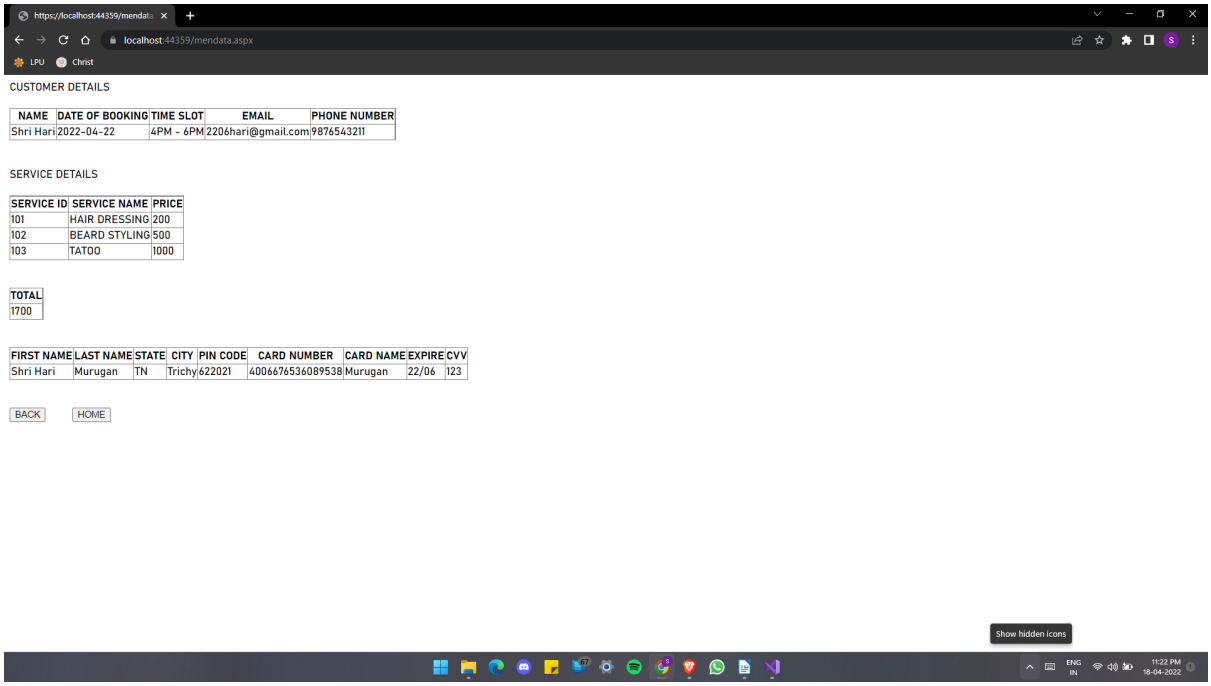
FORM NAME: ADMIN LOGIN



FORM NAME: ADMIN HOME



FORM NAME: ADMIN MEN DETAILS



CUSTOMER DETAILS

NAME	DATE OF BOOKING	TIME SLOT	EMAIL	PHONE NUMBER
Shri Hari	2022-04-22	4PM - 6PM	2206hari@gmail.com	9876543211

SERVICE DETAILS

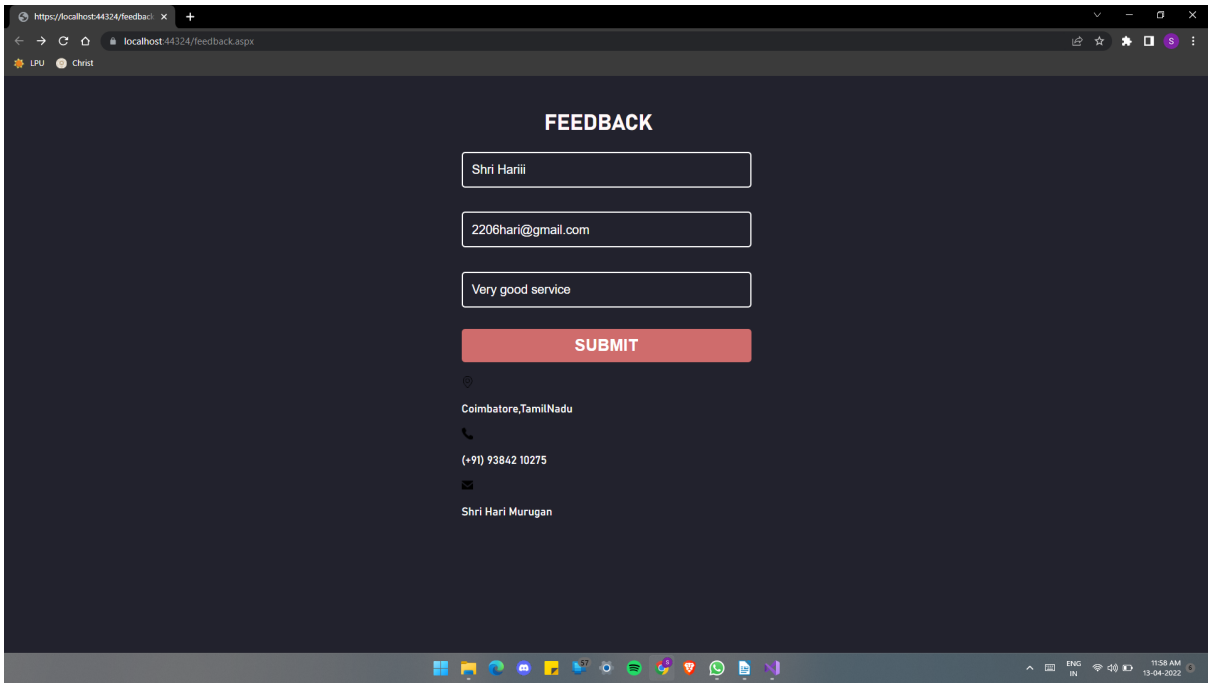
SERVICE ID	SERVICE NAME	PRICE
101	HAIR DRESSING	200
102	BEARD STYLING	500
103	TATOO	1000

TOTAL
1700

FIRST NAME	LAST NAME	STATE	CITY	PIN CODE	CARD NUMBER	CARD NAME	EXPIRE	CVV
Shri Hari	Murugan	TN	Trichy	622021	4006676536089538	Murugan	22/06	123

BACK HOME

FORM NAME: FEEDBACK



REFERENCE

1. “Object Oriented Programming in ASP.NET” Alistar MacMonnies, First Indian Reprint – 2004
2. “Optimizing SQL Server” Robert D. Schneider, Jetty R. Garbus,
3. “Pro ASP.NET Core MVC2” Adam Freeman Apress, now its Seventh edition
4. “The complete reference ASP.NET” Jittery R. Shapiro

WEBSITES

1. https://stackoverflow.com
2. https://codepen.io
3. https://www.w3schools.com
4. http://www.dotnet.microsoft.com