A PROJECT REPORT

ON

“ONLINE BUS PASS SYSTEM”

(Submitted in the partial fulfilment of the requirements for the award of the degree)

Bachelor of Computer Applications [BCA]

Submitted By

Y.Gnaneswar-173665

Under the Guidance of

MR.Y.SRINIVAS RAO

Submitted to

Department of Computer Science & Applications



KAKARAPARTI BHAVANARAYANA COLLEGE (Autonomous)

(Sponsored by S.K.P.V.V Hindu High School Committee)

Kothapeta, Vijayawada.

2017-2020



Department of Computer Science & Applications (Autonomous)



This is to certify that this project work titled “ONLINE BUS PASS SYSTEM” is the bonafide work done by “Y.Gnaneswar(K1710065)” in the partial fulfilment of the requirements for the award of the degree Bachelor of Computer Applications (BCA) at KBN College (Autonomous) affiliated to Krishna University during 2017-2020.

PROJECT GUIDE HEAD OF THE DEPARTMENT

EXTERNAL EXAMINER

VIJAYAWADA-1.

Date:

**DECLARATION**

We hereby declare that this project work titled **“Online Bus Pass System”** under the guidance of “Mr.Y.Srinivas Rao Sir” submitted to Department of Computer Application in partial fulfilment of the award of the degree of “BACHELOR OF COMPUTER APPLICATIONS [B.C.A]”, KBN COLLEGE (Autonomous), Vijayawada is done by us and have not been submitted to any other institution or published else where.

Place: Vijayawada

Date:

|  |  |  |
| --- | --- | --- |
| **Roll No** | **Regd. No** | **Name** |
| 173665 | K1710065 | Y.Gnaneswar |

ACKNOWLEDGEMENT

First of all, we are grateful to the Almighty God for establishing me to complete this project.

We express our sincere thanks to the Management and our beloved Principal

Dr.V. Narayana Rao for providing such wonderful facilities required encouragement in completion of this entire project work.

We place on Record, Our sincere gratitude to Sri. P. Ravindra, HOD of Computer Applications for his constant encouragement. He monitored the progress and arranged all facilities to make this project easier.

We owe our profound gratitude to our Project Guide “Mr.Y.Srinivas Rao” who took keen interest on this project work and guided us all along and whose patience we have probably used to the limit. He was always so involved in the entire process, shared his knowledge, and encouraged to think.

We would like to thank all the other faculty members, technical staff and supporting staff who have provided their contribution in the completion this project work.

Last but not Least, our heartful thanks to the Team Members, our parents and friends for their continuous encouragement and support in the completion these projects work

|  |  |  |
| --- | --- | --- |
| **Roll No** | **Regd. No** | **Name** |
| 173665 | K1710065 | Y.Gnaneswar |

.

**CONTENTS**

1. INTRODUCTION

* Title
* Aim

1. DESCRIPTION

* Existing System
* Proposed System
* Modules description

1. SYSTEM ENVIRONMENT

* Hardware Requirement
* Software Requirement
* About Tools Used

1. DESIGNING DOCUMENTS

* UML Diagrams
* Use Case Diagram
* Sequence Diagram
* Class Diagram
* State Chart Diagram
* Activity Diagram
* E.R Diagrams

1. DATA DICTONARY
2. CODING
3. OUTPUT SCREENS
4. CONSLUSION
5. FUTURE SCOPE

**INTRODUCTION**

It is useful for those who are facing problems with the current manual work of bus pass registration and renewal. The user needs to register by submitting their details through online. The administrator will verify the student details and if they are satisfied they will approve bus pass. The pass will be generated and sent to user’s mail.

**Title : “ONLINE BUS PASS SYSTEM”**

**Aim**: The main aim of the project is to provide safe, reliable, time-saving and affordable services to the user, where the user can generate and renewal the bus pass online.

**DESCRIPTION :**

**EXISTING SYSTEM :**

In this system the person as to visit the counter and have to submit the details and then they have to wait for approval of the bus pass.

**DEMERITS:**

* It is time consuming.
* It consumes lot of manpower to get better results.
* It lacks of data security.
* Reports take time to produce.

**PROPOSED SYSTEM :**

In this system user register their details through online and get there unique username and password. The bus pass for a particular person will be sent to his mail and he can also renewal the buspass.

**MERITS:**

* User can find all the bus pass related information online without going to the bus station.
* This online bus pass software system will help students and passengers get bus passes online and eliminate the need of standing in queues for passes or collecting a ticket for each journey.
* The payment can be done online via credit or master card.

**Module Description**

**ADMINISTRATORMODULE:**  
**Manage profile:** Manage profile of user  
**Manage Pass Scheme:** Manage the different pass schemes  
**Verify Passes:**  Verification of online registration for passes.  
**Pass History:**  He can check the history of pass of any user for verification.

**USERMODULE:**  
**Member Registration:** User can create his account for pass registration or renewal.  
**Login:**  User can login for pass registration or renewal or for manage of account.  
**Manage profile:** User can manage or update in his profile.  
**Change password:** User can change his password after login into profile.  
**Pass Schemes:** User can select pass scheme related to him.  
**Pass Registration:** User can register for pass.  
**Generate Pass:** After Complete registration user can generate his pass.

**PAYMENT MODULE**:   
  
**Credit/debit:** Entering the card number to pay.  
**UPI:** Paying the money through Paytm,Tez etc.  
**Net banking:** Paying through bank A/c online.

**SYSTEM ENVIRONMENT**

**Hardware requirements:**

**Processor :**Intel Pentium-IV

**Hard Disk :**250 GB

**Memory :** 1GB RAM

**Software requirements:**

**Operating System:** Windows 7

**Front End:** ASP.NET

**Back End:** SQL Server

**Framework:** **.**NET Framework

**About Tools Used**

**INTRODUCTION TO .NET Framework**

The .NET Framework is a new computing platform that simplifies application development in the highly distributed environment of the Internet. The .NET Framework is designed to fulfill the following objectives.

* To provide a consistent object-oriented programming environment whether object code is stored and executed locally, executed locally but Internet-distributed, or executed remotely.
* To provide a code-execution environment that minimizes software deployment and versioning conflicts.
* To provide a code-execution environment that guarantees safe execution of code, including code created by an unknown or semi-trusted third party.
* To provide a code-execution environment that eliminates the performance problems of scripted or interpreted environments.
* To make the developer experience consistent across widely varying types of applications, such as Windows-based applications and Web-based applications.
* To build all communication on industry standards to ensure that code based on the .NET Framework can integrate with any other code.

The .NET Framework has two main components: the common language runtime and the .NET Framework class library. The common language runtime is the foundation of the .NET Framework. You can thick of the runtime as an agent that manages code at execution time, providing core services such as memory management, thread management, and remoting, while also enforcing strict type safety and other forms of code accuracy that ensure security and robustness. In fact, the concept of code management is a fundamental principle of the runtime. Code that targets the runtime is known as a managed code, while code that does not target the runtime is known as unmanaged code. The class library, the other main component of .NET Framework, is a comprehensive, object-oriented collection of reusable types that you can use to develop applications ranging from traditional command-line or graphical user interface (GUI) applications to applications based on the latest innovations provided by ASP.NET such as Web Forms and XML Web services.

The .NET Framework can he hosted by unmanaged components that load the common language runtime into their processes and initiate the execution of managed code, thereby creating a software environment that can exploit both managed and unmanaged features. The .NET Framework not only provides several runtime hosts, but also supports the development of third-party runtime hosts.

For example, ASP.NET hosts the runtime to provide a scalable, server-side environment for managed code. ASP.NET works directly with the runtime to enable Web Forms applications and XML Web services, both of which are discussed later in this topic.

Internet Explorer is an example of an unmanaged application that hosts the runtime (in the form of a MIME type extension). Using Internet Explorer to host the runtime enables you to embed managed components or Windows Forms controls in HTML documents.

**FEATURES OF THE COMMON LANGUAGE RUNTIME**

The common language runtime manages memory, thread execution, code execution, code safety verification, compilation, and other system services. These features are intrinsic to the managed code that runs on the common language runtime.

With regards to security, managed components are awarded varying degrees of trust, depending on a number of factors that include their origin (such as the Internet, enterprise network, or local computer). This means that a managed component might or might not be able to perform file-access operations, registry-access operations, or other sensitive functions, even if it is being used in the same active application.

The runtime enforces code access security. For example, users can trust that an executable embedded in a Web page can play an animation on screen or sing a song, but cannot access their personal data, file system, or network. The security features of the runtime thus enable legitimate Internet-deployed software to be exceptionally featuring rich.

The runtime also enforces code robustness by implementing a strict type and codeverification infrastructure called the Common Type System (CTS). The CTS ensures that all managed code is self-describing. The various Microsoft and third-party language compilers.

Generate managed code that conforms to the CTS. This means that managed code can consume other managed types and instances, while strictly enforcing type fidelity and type safety.

In addition, the managed environment of the runtime eliminates many common software issues. For example, the runtime automatically handles object layout and manages references to objects, releasing them when they are no longer being used.

**.NET FRAMEWORK CLASS LIBRARY**

The .NET Framework class library is a collection of reusable types that tightly integrate with the common language runtime. The class library is object oriented, providing types from which your own managed code can dive functionality. This not only makes the .NET Framework types easy to use, but also reduces the time associated with learning new features of the .NET Framework. In addition, third-party components can integrate seamlessly with classes in the .NET Framework.

As would be expect from an object-oriented class library, the .NET Framework types enable you to accomplish a range of common programming tasks, including such as string management, data collection, database connectivity, and file access. In addition to these common tasks, the class library includes types that support a variety of specialized development scenarios. For example, you can use the .NET Framework to develop the following types of applications and services.

* Console applications.
* Scripted or hosted applications.
* Windows GUI applications (Windows Forms).
* ASP.NET applications.
* XML Web services.
* Windows Services.

For example, the Windows Form classes are a comprehensive set of reusable types that vastly simplify Windows GUI development. If you write an ASP.NET Web Form application, you can use the Web Forms classes.

**CLIENT APPLICATION DEVELOPMENT :**

Client applications are the closest to a traditional style of application in Windows-based programming. These are the types of applications that display windows or forms on the desktop,

Enabling a user to perform a task. Client applications include applications such as word processors and spread sheets, as well as custom business applications such as data-entry tools, reporting tools, and so on. Client applications usually employ windows, menus, buttons, and other GUI elements, and they likely access local resources such as the file system and peripherals such as printers. Another kind of client application is the traditional ActiveX control (now replaced by the managed Windows Forms control) deployed over the Internet as a Web page. This application is much like other client applications: it is executed natively, has access to local resources, and includes graphical elements.

In the past, developers created such applications using C/C++ in conjustion with the Microsoft Foundation Classes (MFC) or with a Rapid Application Development (RAD) environment such as Microsoft® Visual Basic®. The .NET Framework incorporates aspects of these existing products into a single, consistent development environment that drastically simplifies the development of client applications.

The Windows Forms classes contained in the .NET Framework are designed to be used for GUI development. You can easily create command windows, buttons, menus, toolbars, and other screen elements with the flexibility necessary to accommodate shifting business needs.

For example, the .NET Framework provides simple properties to adjust visual attributes associated with forms. In some cases the underlying operating system does not support changing these attributes directly, and in these cases the .NET Framework automatically recreates the forms. This is one of many ways in which the .NET Framework integrates the developer interface, making coding simpler and more consistent.

**ASP.NET**

**Server Application Development**

Server-side applications in the managed world are implemented through runtime hosts. Unmanaged applications host the common language runtime, which allows your custom managed code to control the behavior of the server. This model provides you with all the features of the common language runtime and class library while gaining the performance and scalability of the host server.

The following illustration shows a basic network schema with managed code running in different server environments. Servers such as IIS and SQL Server can

perform standard operations while your application logic executes through the managed code.

**SERVER-SIDE MANAGED CODE**

ASP.NET is the hosting environment that enables developers to use the .NET Framework to target Web-based applications. However, ASP.NET is more than just a runtime host; it is a complete architecture for developing Web sites and Internet-distributed objects using managed code. Both Web Forms and XML Web services use IIS and ASP.NET as the publishing mechanism for applications, and both have a collection of supporting classes in the .NET Framework.

XML Web services, an important evolution in Web-based technology, are distributed, server-side application components similar to common Web sites. However, unlike Web-based applications, XML Web services components have no UI and are not targeted for browsers such as Internet Explorer and Netscape

Navigator. Instead, XML Web services consist of reusable software components designed to be consumed by other applications, such as traditional client applications, Web-based applications, or even other XML Web services. As a result, XML Web services technology is rapidly moving application development and deployment into the highly distributed environment of the Internet. If you have used earlier versions of ASP technology, you will immediately notice the improvements that ASP.NET and Web Forms offers. For example, you can develop Web Forms pages in any language that supports the .NET Framework. In addition, your code no longer needs to share the same file with your HTTP text (although it can continue to do so if you prefer). Web Forms pages execute in native machine language because, like any other managed application, they take full advantage of the runtime. In contrast, unmanaged ASP pages are always scripted and interpreted. ASP.NET pages are faster, more functional, and easier to develop than unmanaged ASP pages because they interact with the runtime like any managed application. The .NET Framework also provides a collection of classes and tools to aid in development and consumption of XML Web services applications. XML Web services are built on standards such as SOAP (a remote procedure-call protocol), XML (an extensible data format), and WSDL (the Web Services Description Language). The .NET Framework is built on these standards to promote interoperability with non-Microsoft solutions.

For example, the Web Services Description Language tool included with the .NET Framework SDK can query an XML Web service published on the Web, parse its WSDL description, and produce C# or Visual Basic source code that your application can use to become a client of the XML Web service. The source code can create classes derived from classes in the class library that handle all the underlying communication using SOAP and XML parsing. Although you can use the class library to consume XML Web services directly, the Web Services

Description Language tool and the other tools contained in the SDK facilitate your development efforts with the .NET Framework.

If you develop and publish your own XML Web service, the .NET Framework provides a set of classes that conform to all the underlying communication standards, such as SOAP, WSDL, and XML. Using those classes enables you to focus on the logic of your service, without concerning yourself with the communications infrastructure required by distributed software development.

Finally, like Web Forms pages in the managed environment, your XML Web service will run with the speed of native machine language using the scalable communication of IIS.

**ACTIVE SERVER PAGES.NET**

ASP.NET is a programming framework built on the common language runtime that can be used on a server to build powerful Web applications. ASP.NET offers several important advantages over previous Web development models:

* **Enhanced Performance:** ASP.NET is compiled common language runtime code running on the server. Unlike its interpreted predecessors, ASP.NET can take advantage of early binding, just-in-time compilation, native optimization, and caching services right out of the box. This amounts to dramatically better performance before you ever write a line of code.
* **World-Class Tool Support:** The ASP.NET framework is complemented by a rich toolbox and designer in the Visual Studio integrated development environment. WYSIWYG editing, drag- and-drop server controls, and automatic deployment are just a few of the features this powerful tool provides.
* **Power and Flexibility:** Because ASP.NET is based on the common language runtime, the power and flexibility of that entire platform is available to Web application developers. The .NET Framework class library, Messaging, and Data.

Access solutions are all seamlessly accessible from the Web. ASP.NET is also language independent, so you can choose the language that best applies to your application or partition your application across many languages. Further, common language runtime interoperability guarantees that your existing investment in COM-based development is preserved when migrating to ASP.NET.

* **Simplicity:** ASP.NET makes it easy to perform common tasks, from simple form submission and client authentication to deployment and site configuration. For example, the ASP.NET page framework allows you to build user interfaces that cleanly separate application logic from presentation code and to handle events in a simple, Visual Basic - like forms processing model. Additionally, the common language runtime simplifies development, with managed code services such as automatic reference counting and garbage collection.
* **Manageability:** ASP.NET employs a text-based, hierarchical configuration system, which simplifies applying settings to your server environment and Web applications. Because configuration information is stored as plain text, new settings may be applied without the aid of local administration tools. This "zero local administration" philosophy extends to deploying ASP.NET Framework applications as well. An ASP.NET Framework application is deployed to a server simply by copying the necessary files to the server. No server restart is required, even to deploy or replace running compiled code.
* **Scalability and Availability:** ASP.NET has been designed with scalability in mind, with features specifically tailored to improve performance in clustered and multiprocessor environments. Further, processes are closely monitored and managed by the ASP.NET runtime, so that if one misbehaves (leaks, deadlocks), a new process can be created in its place, which helps keep your application constantly available to handle requests.
* **Customizability and Extensibility:** ASP.NET delivers well-factored architecture that allows developers to "plug-in" their code at the appropriate level. In fact, it is possible to extend or replace any subcomponent of the ASP.NET runtime with your own customwritten component. Implementing custom authentication or state services has never been easier.
* **Security:** With built in Windows authentication and per-application configuration, you can be assured that your applications are secure.

**LANGUAGE SUPPORT**

The Microsoft .NET Platform currently offers built-in support for three languages: C#, Visual Basic, and JavaScript.

**WHAT IS ASP.NET WEB FORMS?**

The ASP.NET Web Forms page framework is a scalable common language runtime programming model that can be used on the server to dynamically generate Web pages. Intended as a logical evolution of ASP (ASP.NET provides syntax compatibility with existing pages), the ASP.NET Web Forms framework has been specifically designed to address a number of key deficiencies in the previous model. In particular, it provides:

* The ability to create and use reusable UI controls that can encapsulate common functionality and thus reduce the amount of code that a page developer has to write.
* The ability for developers to cleanly structure their page logic in an orderly fashion (not "spaghetti code").
* The ability for development tools to provide strong WYSIWYG design support for pages (existing ASP code is opaque to tools).

ASP.NET Web Forms pages are text files with an .aspx file name extension. They can be deployed throughout an IIS virtual root directory tree. When a browser client requests.aspx resources, the ASP.NET runtime parses and compiles the target file into a .NETFramework class. This class can then be used to dynamically process incoming requests. (Note that the .aspx file is compiled only the first time it is accessed; the compiled type instance is then reused across multiple requests).

ASP.NET Web Forms pages are text files with an .aspx file name extension. They can be deployed throughout an IIS virtual root directory tree. When a browser client requests.aspx resources, the ASP.NET runtime parses and compiles the target file into a .NETFramework class. This class can then be used to dynamically process incoming requests.

ASP.NET provides syntax compatibility with existing ASP pages. This includes support for <% %> code render blocks that can be intermixed with HTML

**CODE-BEHIND WEB FORMS**

ASP.NET supports two methods of authoring dynamic pages. The first is the method shown in the preceding samples, where the page code is physically declared within the originating .aspx file. An alternative approach--known as the code-behind method—enables the page code to be more cleanly separated from the HTML content into an entirely separate file.

**INTRODUCING TO ASP.NET SERVER CONTROLS**

In addition to (or instead of) using <% %> code blocks to program dynamic content,ASP.NET page developers can use ASP.NET server controls to program Web pages. Server controls are declared within an .aspx file using custom tags or intrinsic HTML tags that contains a run at="server" attributes value. Intrinsic HTML tags are handled by one of the controls in the **System.Web.UI.HtmlControls** namespace. Any tag that doesn't explicitly map to one of the control is assigned the type of **System.Web.UI.HtmlControls.HtmlGenericControl.** Server controls automatically maintain any client-entered values between round trips to the server. This control state is not stored on the server (it is instead stored within an<input type="hidden"> form field that is round-tripped between requests). Note also that no client-side script is required. In addition to supporting standard HTML input controls, ASP.NET enables developers to utilize richer custom controls on their pages. For example, the following sample demonstrates how the **<asp: adrotator>** control can be used to dynamically display rotating ads on a page.

1. ASP.NET Web Forms provide an easy and powerful way to build dynamic Web UI.

2. ASP.NET Web Forms pages can target any browser client (there are no script library or cookie requirements).

3. ASP.NET Web Forms pages provide syntax compatibility with existing ASP pages.

4. ASP.NET server controls provide an easy way to encapsulate common functionality.

5. ASP.NET ships with 45 built-in server controls. Developers can also use controls built by third parties.

6. ASP.NET server controls can automatically project both up level and down-level HTML.

7. ASP.NET templates provide an easy way to customize the look and feel of list server controls.

8. ASP.NET validation controls provide an easy way to do declarative client or server data validation.

**C#.NET**

**ADO.NET OVERVIEW**

ADO.NET is an evolution of the ADO data access model that directly addresses user requirements for developing scalable applications. It was designed specifically for the web with scalability, statelessness, and XML in mind.

ADO.NET uses some ADO objects, such as the Connection and Command objects, and also introduces new objects. Key new ADO.NET objects include the DataSet, **DataReader,** and **DataAdapter.**

The important distinction between this evolved stage of ADO.NET and previous data architectures is that there exists an object -- the **DataSet** -- that is separate and distinct for many data stores. Because of that, the **DataSet** functions as a standalone entity. You can think of the **DataSet** as an always disconnected record set that knows nothing about the source or destination of the data it contains. Inside a **DataSet,** much like in a database, there are tables, columns, relationships, constraints, views, and so forth. A **DataAdapter** is the object that connects to the database to fill the **DataSet.** Then, it connects back to the database to update the data there, based on operations performed while the **DataSet** held the data. In the past,data processing has been primarily connection-based. Now, in an effort to make multi-tiered apps more efficient, data processing is turning to a message-based approach that revolves around chunks of information. At the center of this approach

is the **DataAdapter**, which provides a bridge to retrieve and save data between a **DataSet** and its source data store. It accomplishes this by means of requests to the appropriate SQL commands made against the data store.

The XML-based **DataSet** object provides a consistent programming model that works with all models of data storage: flat, relational, and hierarchical. It does this by having no 'knowledge' of the source of its data, and by representing the data that it holds as collections and data types. No matter what the source of the data within the **DataSet** is, it is manipulated through the same set of standard APIs exposed through the **DataSet** and its subordinate objects.

While the **DataSet** has no knowledge of the source of its data, the managed provider has detailed and specific information. The role of the managed provider is to connect, fill, and persist the **DataSet** to and from data stores. The OLE DB and SQL Server .NET Data Providers **(System.Data.OleDb** and **System.Data.SqlClient)** that are part of the .NetFramework provide four basic objects: the **Command**, **Connection**, **DataReader** and **DataAdapter.** In the remaining sections of this document, we'll walk through each part of the **DataSet** and the OLE DB/SQL Server .NET Data Providers explaining what they are, and how to program against them.

The following sections will introduce you to some objects that have evolved, and some that are new. These objects are:

* **Connections**. For connection to and managing transactions against a database.
* **Commands**. For issuing SQL commands against a database.
* **DataReaders**. For reading a forward-only stream of data records from a SQL Server data source.
* **DataSets**. For storing, Remoting and programming against flat data, XML data and relational data.
* **DataAdapters**. For pushing data into a DataSet, and reconciling data against a database

When dealing with connections to a database, there are two different options: SQLServer .NET Data Provider (System.Data.SqlClient) and OLE DB .NET Data Provider(System.Data.OleDb). In these samples we will use the SQL Server .NET

Data Provider. These are written to talk directly to Microsoft SQL Server. The OLE DB .NET Data Provider is used to talk to any OLE DB provider (as it uses OLE DB underneath).

**Connections**

Connections are used to 'talk to' databases, and are represented by provider-specific classes such as **SqlConnection**. Commands travel over connections and result sets are returned in the form of streams which can be read by a **DataReader** object, or pushed into a **DataSet** object.

**Commands**

Commands contain the information that is submitted to a database, and are represented by provider-specific classes such as **SqlCommand**. A command can be a stored procedure call, an UPDATE statement, or a statement that returns results. You can also use input and output parameters, and return values as part of your command syntax. The example below shows how to issue an **INSERT** statement against the **North** **wind** **database**.

**Data** **Readers**:

The **Data** **Reader** object is somewhat synonymous with a read-only/forward-only cursor over data. The **Data** **Reader** API supports flat as well as hierarchical data. A **Data Reader** object is returned after executing a command against a database. The format of the returned **Data** **Reader** object is different from a record set. For example, you might use the Data Reader to show the results of a search list in a web page.

**DATA SETS AND DATA ADAPTERS:**

**DataSets:**

The **DataSet** object is similar to the ADO **Recordset** object, but more powerful, and with one other important distinction: the **DataSet** is always disconnected. The **DataSet** object represents a cache of data, with database-like structures such as tables, columns, relationships, and constraints. However, though a **DataSet** can and does behave much like a database, it is important to remember that **DataSet** objects do not interact directly with databases, or other source data. This allows the developer to work with a programming model that is always consistent, regardless of where the source data resides. Data coming from a database, an XML file, from code, or user input can all be placed into **DataSet** objects. Then, as changes are made to the **DataSet** they can be tracked and verified before updating the source data.

Get Changes method of the **DataSet** object actually creates a second **DataSet** that contains only the changes to the data. This **DataSet** is then used by a **DataAdapter** (or other objects) to update the original data source.

The **DataSet** has many XML characteristics, including the ability to produce and consume XML data and XML schemas. XML schemas can be used to describe

schemas interchanged via **WebServices**. In fact, a **DataSet** with a schema can actually be compiled for type safety and statement completion.

**DATAADAPTERS (OLEDB/SQL)**

The **DataAdapter** object works as a bridge between the **DataSet** and the source data. Using the provider-specific **SqlDataAdapter** (along with its associated **SqlCommand** and **SqlConnection**) can increase overall performance when working with a Microsoft **SQLServer** databases. For other OLE DB-supported databases, you would use the **OleDbDataAdapter** object and its associated **OleDbCommand** and **OleDbConnection** objects.

The **DataAdapter** object uses commands to update the data source after changes have been made to the **DataSet**. Using the Fill method of the **DataAdapter** calls the SELECT command; using the Update method calls the INSERT, UPDATE or DELETE command for each changed row. You can explicitly set these commands in order to control the statements used at runtime to resolve changes, including the use of stored procedures. For ad-hocscenarios, a **CommandBuilder** object can generate these at run-time based upon a select statement. However, this run-time generation requires an extra round-trip to the server.

In order to gather required metadata, so explicitly providing the INSERT, UPDATE, and DELETE commands at design time will result in better run-time performance.

1. ADO.NET is the next evolution of ADO for the .Net Framework.

2. ADO.NET was created with n-Tier, statelessness and XML in the forefront. Two new objects, the **DataSet** and **DataAdapter**, are provided for these scenarios.

3. ADO.NET can be used to get data from a stream, or to store data in a cache for updates.

4. ADO.NET can be used to get data from a stream, or to store data in a cache for updates.

5. Remember, you can execute a command directly against the database in order to do inserts, updates, and deletes. You don't need to first put data into a DataSet in order to insert, update, or delete it.

6. Also, you can use a **DataSet** to bind to the data, move through the data, and navigate data relationships.

**SQL SERVER**

A database management or DBMS gives the user access to their data and helps them transform the data into information. Such database management systems include dBase, paradox, IMS, SQL Server and SQL Server. These systems allow users to create, update and extract information from their database.

A database is a structured collection of data. Data refers to the characteristics of people things and events. SQL server stores each data item its own fields. In SQL Server, the fields relating to a participating person, thing or even6t are bundled together to form a single complete unit of data, called a record .(it can also be referred to as raw or an occurrence). Each record is made up of a number of fields. No two fields in a record can have the same field name.

During an SQL Server Database design project, the analysis of your business needs identifies all the fields or attributes of interest. If your business needs change over time, you define any additional fields or change the definition of existing fields.

**PRIMARY** **KEY**

Every table in SQL Server has a field or a combination of fields that uniquely identifies each record in the table. The unique identifier is called the primary key, or simply the key. The primary key provides the means to distinguish one record from all other in a table. It allows the user and the database system to identity, locate and refer to one particular record in the database.

**RELATIONAL** **DATABASE**

Sometimes all the information of interest to a business operation can be stored in one table. SQL Server makes it very easy to link the data in multiple tables. Matching an employee into the department in which they work in one example. This

is what makes SQL server a Relational Database Management System or RDBMS stored in a two or more tables and enables to define relationship between the tables and enables you to define relationships between the tables.

**FOREIGN** **KEY**

When a field is one table matches the primary key of anther field is referred to as a foreign key. A foreign key is a field or a group of fields in one table whose values match those of the primary key of another table.

**REFERENTIAL** **INTEGRITY**

Not only does SQL Server allow you to link multiple tables, it also maintains consistency between them. Ensuring that the data among related tables is correctly matched is referred to as maintaining referential integrity

**DATA ABSTRACTION**

A major purpose of a database system is to provide users with an abstract view of the data. This system hides certain details of how the data is stored and maintained. Data abstraction is divided into three levels.

* **Physical Level**: This is the lowest level of abstraction at which one describes how the data are actually stored.
* **Conceptual Level**: At this level of database abstraction all the attributed and what data are actually stored is described and entries and relationship among them.
* **View** **Level**: This is the highest level of abstraction at which one describes only part of the database.

**ADVANTAGES OF RDBMS**

* Redundancy can be avoided
* Inconsistency can be eliminated
* Data can be Shared
* Standards can be enforced
* Security restrictions can be applied
* Integrity can be maintained
* Conflicting requirements can be balanced
* Data independence can be achieved.

**LIMITATIONS OF DBMS**

A significant disadvantage of the DBMS system is cost. In addition to the cost of purchasing of developing the software, the hardware has to be upgraded to allow for the extensive programs and the workspace required for their execution and storage.

**FEATURES OF SQL SERVER (RDBMS)**

SQL SERVER is one of the leading database management system (DBMS) because it is the only Database that meets the uncompromising requirements of today’s most demanding information systems. From complex decision support systems (DSS) to the most rigorous online transaction processing (OLTP) application, even application that require simultaneous DSS and OLTP access to the same critical data, SQL Server leads the industry in both performance and capability

. SQL SERVER is a truly portable, distributed, and open DBMS that delivers unmatched performance, continuous operation and support for every database.

SQL SERVER RDBMS is high performance fault tolerant DBMS which is specially designed for online transactions processing and for handling larger database application.

**ENTERPRISE** **WIDE** **DATA** **SHARING**

The unrivaled portability and connectivity of the SQL SERVER DBMS enables all the systems in the organization to be linked into a singular, integrated computing resource.

**PORTABILITY**

SQL SERVER is fully portable to more than 80 distinct hardware and operating systems platforms, including UNIX, MSDOS, OS/2, Macintosh and dozens of proprietary platforms. This portability gives complete freedom to choose the database server platform that meets the system requirements.

**OPEN** **SYSTEMS**

SQL SERVER offers a leading implementation of industry-standard SQL. SQL Server’s open architecture integrates SQL SERVER and non-SQL SERVER DBMS with industries most comprehensive collection of tools, application, and third party software products SQL Server’s Open architecture provides transparent access to data from other relational database and even non-relational database.

**DESIGNING:**

Design is the first step in the development phase for any techniques and principles for the purpose of defining a device, a process or system in sufficient detail to permit its physical realization. Once the software requirements have been analysed and specified the software design involves three technical activities-designs, coding, implementation and testing that are required to build and verify the software. The design activities are of main importance in this phase, because in this activity, decisions ultimately affecting the success of the software implementation and its ease of maintenance are made. These decisions have the final bearing upon reliability and maintainability of the system. Design is the only way to accurately translate the customer’s requirements into finished software or a system. Design is the place where quality is fostered in development. Software design is a process through which requirements are translated into a representation of software. Software design is conducted in two steps. Preliminary design is concerned with the transformation of requirements into data.

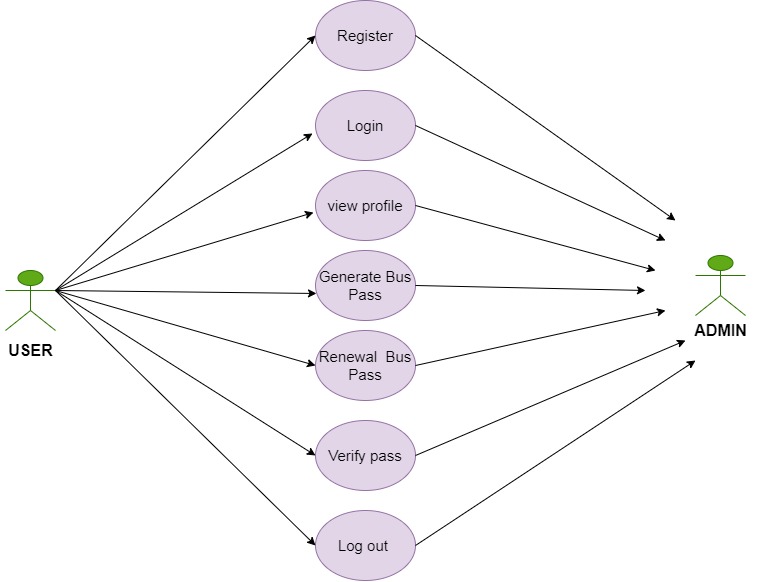
**UML Diagrams**

UML stands for Unified Modeling Language. It provides a different diagrams to visualize the design of the system.

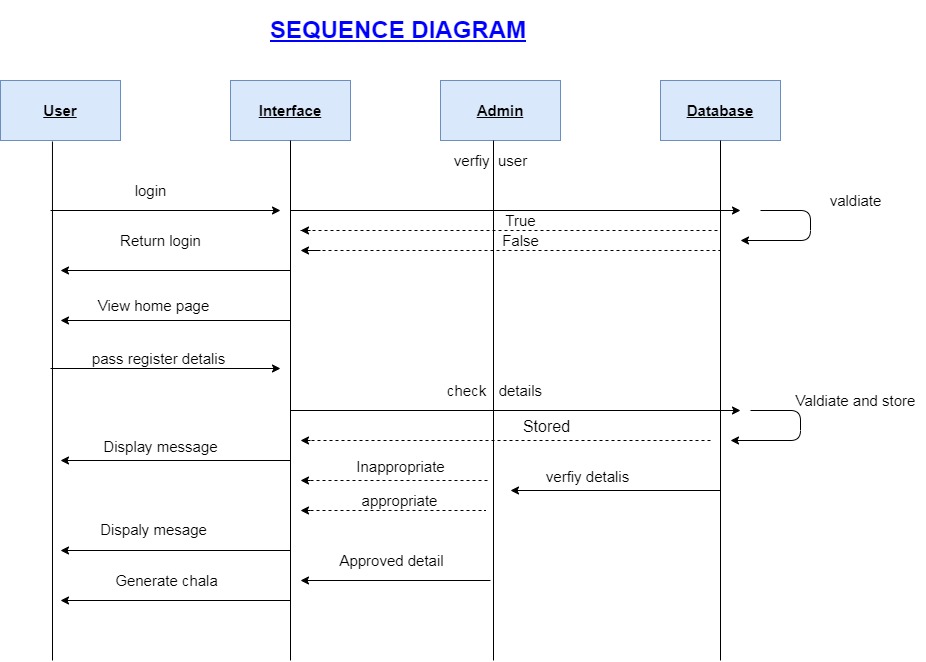
**Types of UML diagrams:**

* Use Case Diagram
* Sequence Diagram
* Class Diagram
* State Chart Diagram
* Activity Diagram

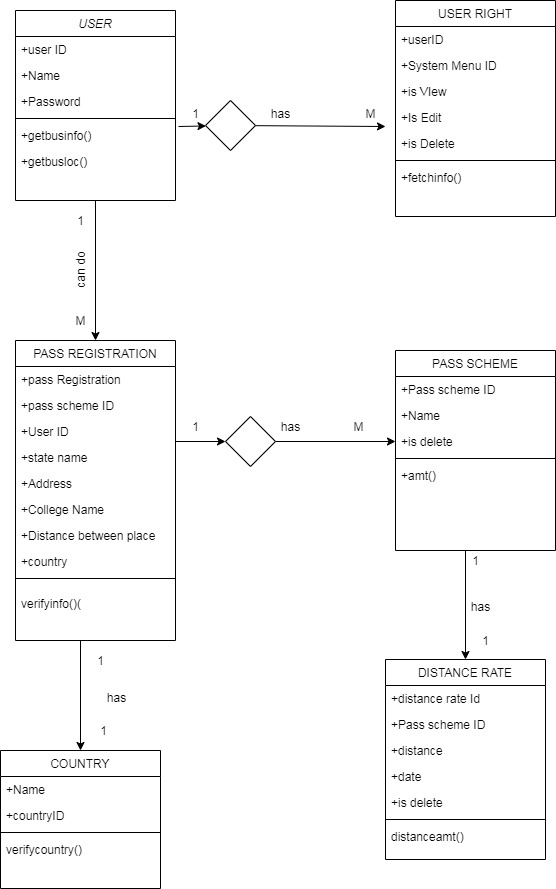
**Use Case Diagram:** Representation of a user interaction with the system.



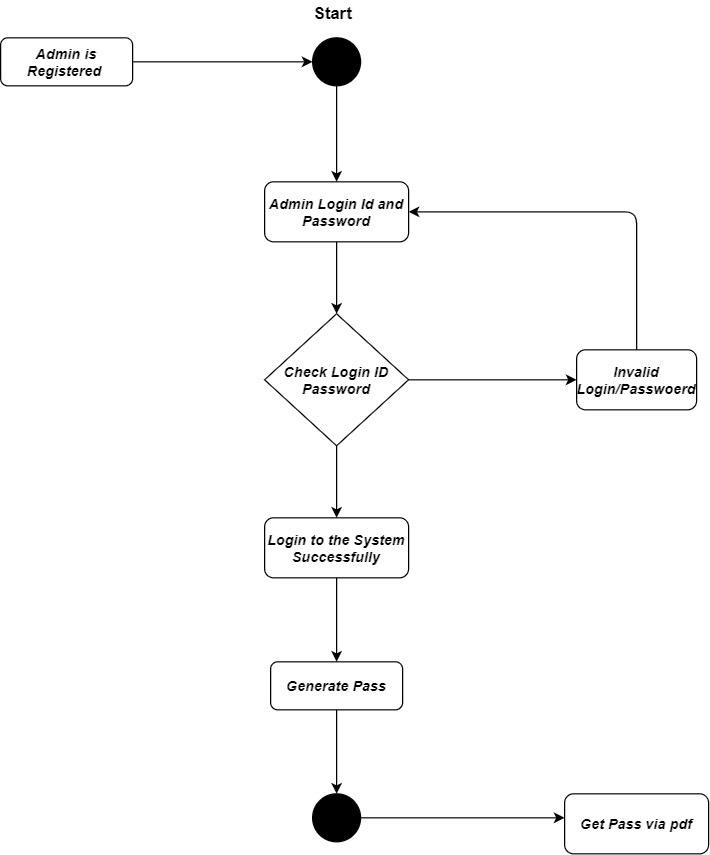
**Sequence diagram:** It describes how the order a group of objects work together.



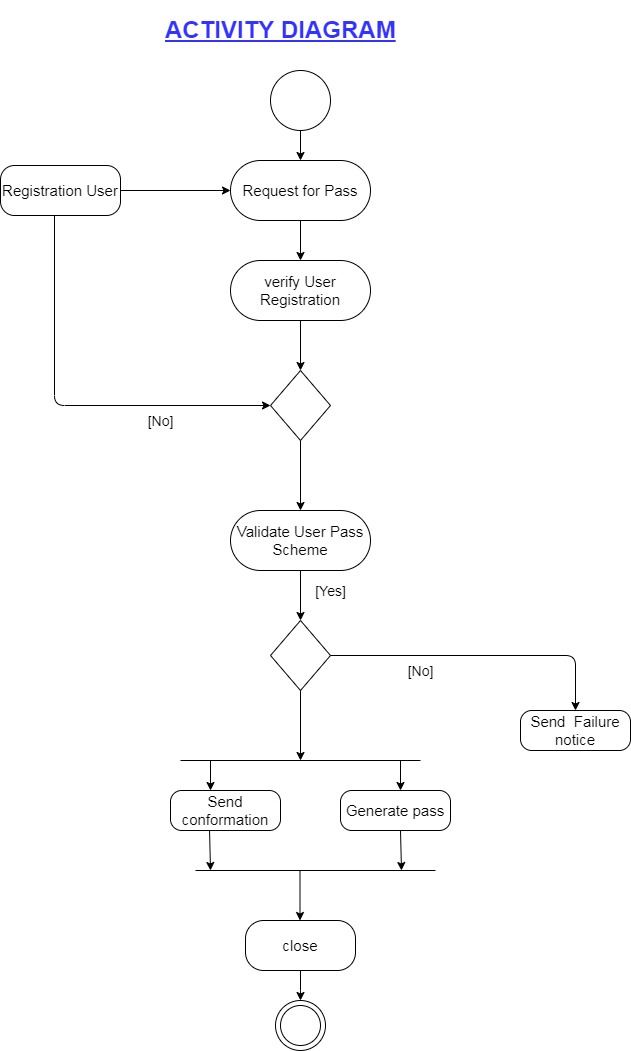
**Class Diagram:** It describes the structure of a system.



**State Chart Diagram:**It describes the behaviour of systems.



**Activity diagram :** It describes the workflows of stepwise activities and actions.



**E-R Diagrams:**

The Entity-Relationship (ER) model was originally proposed by Peter in 1976 [Chen76] as a way to unify the network and relational database views. Simply stated the ER model is a conceptual data model that views the real world as entities and relationships. A basic component of the model is the Entity-Relationship diagram which is used to visually represent data objects. Since Chen wrote his paper the model has been extended and today it is commonly used for database design for the database designer, the utility of the ER model is: It maps well to the relational model. The constructs used in the ER model can easily be transformed into relational tables. It is simple and easy to understand with a minimum of training. Therefore, the model can be used by the database designer to communicate the design to the end user. In addition, the model can be used as a design plan by the database developer to implement a data model in specific database management software.

**Connectivity and Cardinality**:

The basic types of connectivity for relations are: one-toone, one-to-many, and many-to-many. A one-to-one (1:1)relationship is when at most one instance of an entity A is associated with one instance of entity B. For example, "employees in the company are each assigned their own office. For each employee there exists a unique office and for each office there exists a unique employee. A one-to-many (1:N)relationships is when for one instance of entity A, there are zero, one, or many instances of entity B, but for one instance of entity B, there is only one instance of entity A. An example of a 1:N relationships is a department has many employees each employee is assigned to one department. A many-to-many (M:N)relationship, sometimes called non-specific, is when for one instance of entity A, there are zero, one, or many instances of entity B and for one instance of entity B there are zero, one, or many instances of entity A. The connectivity of a relationship describes the mapping of associated.

**ER** **Notation**:

There is no standard for representing data objects in ER diagrams. Each modeling methodology uses its own notation. The original notation used by Chen is widely used in academics texts and journals but rarely seen in either CASE tools or publications by nonacademics. Today, there are a number of notations used; among the more common are Bachman, crow's foot, and IDEFIX. All notational styles represent entities as rectangular boxes and relationships as lines connecting boxes. Each style uses a special set of symbols to represent the cardinality of a connection.

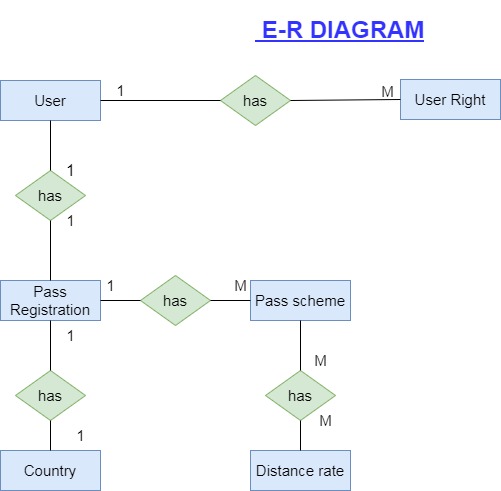
The notation used in this document is from Martin. The symbols used for the basic ER constructs are:

**Entities** are represented by labelled rectangles. The label is the name of the entity. Entity names should be singular nouns.

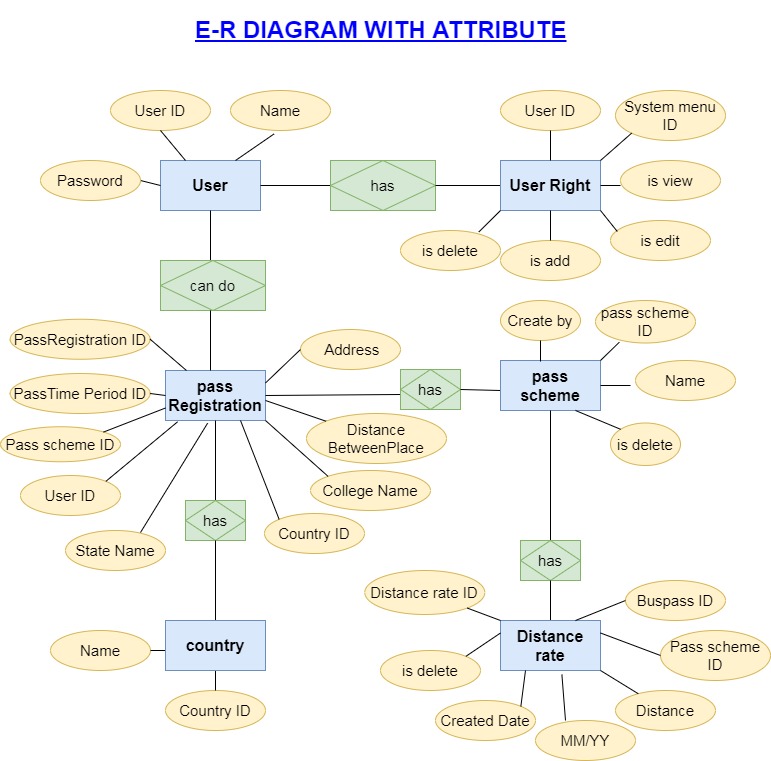
**Relationships** are represented by a solid line connecting two entities. The name of the relationship is written above the line. Relationship names should be verbs.

**Attributes**, when included, are listed inside the entity rectangle. Attributes which are identifiers are underlined. Attribute names should be singular nouns.

**E.R Diagram :**



**E.R diagrams with attributes:**



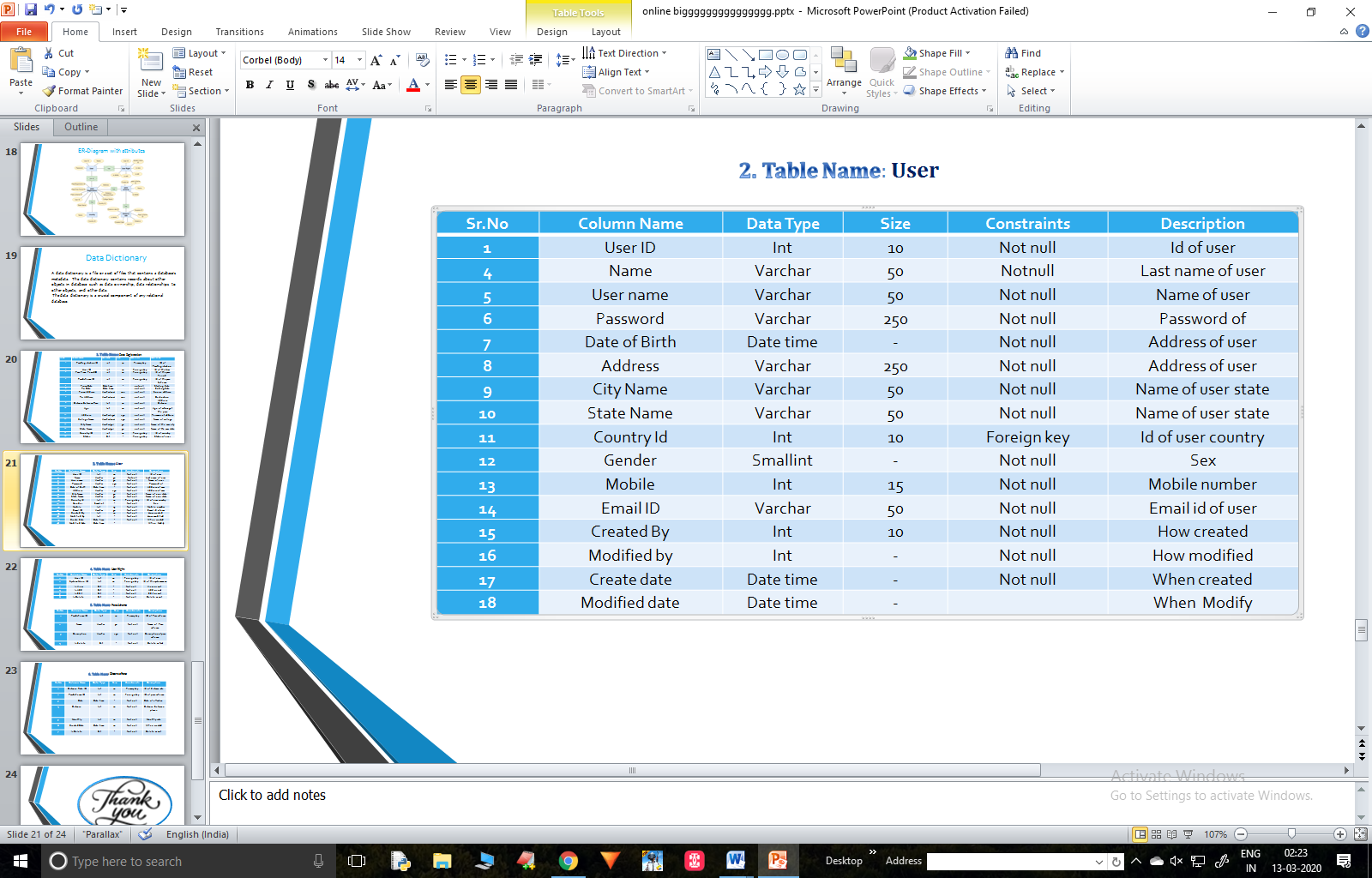
**Data Dictionary :**

A data dictionary is a file or a set of files that contains a database's metadata. The data dictionary contains records about other objects in database such as data ownership, data relationships to other objects, and other data.

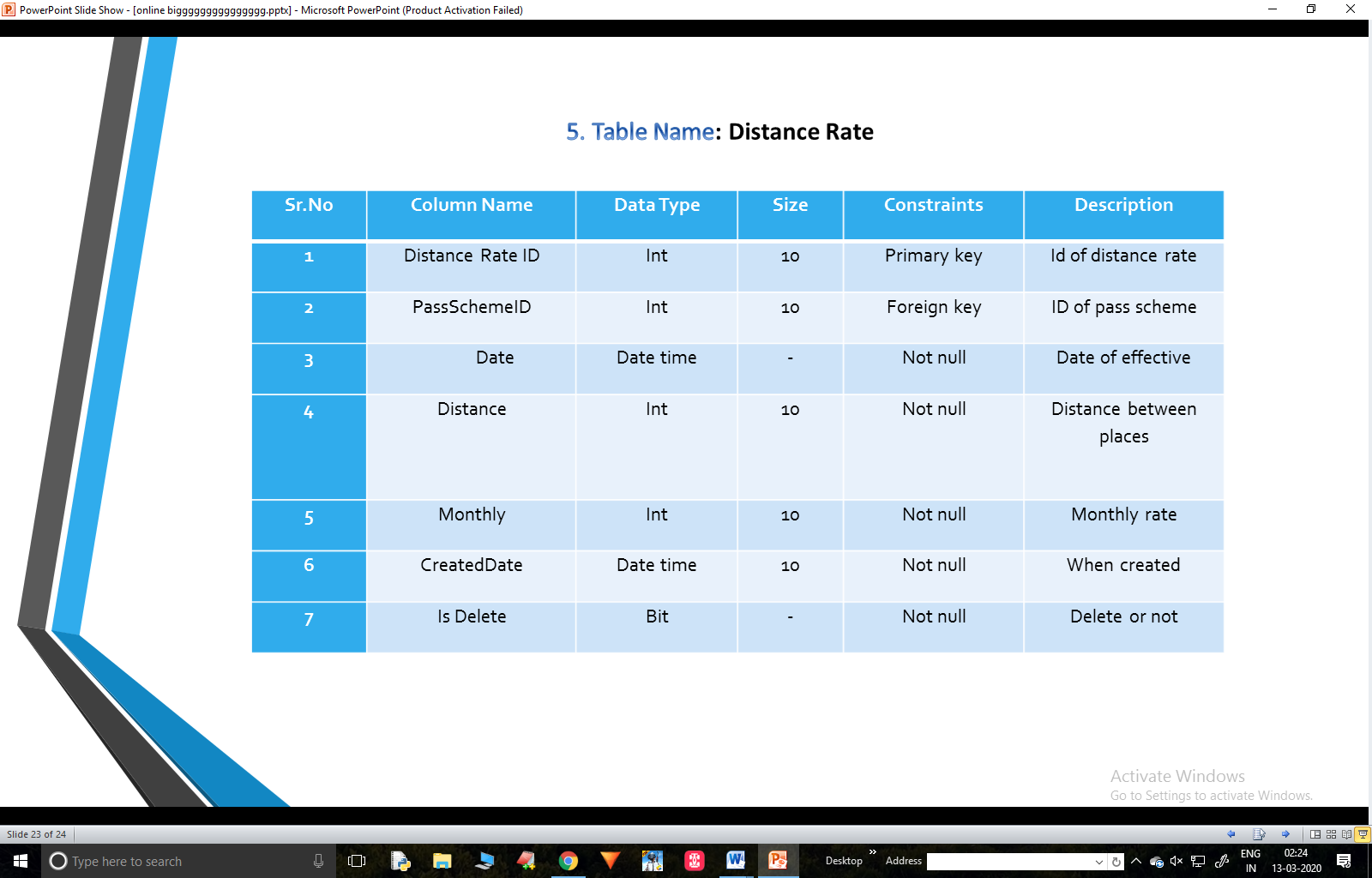
The data dictionary is a crucial component of any relational database.

A data dictionary is a file or a set of files that contains a database's metadata. The data dictionary contains records about other objects.









**CODING:**

**asp:Content ID="Content1" ContentPlaceHolderID="head" Runat="Server">**

**<style type="text/css">**

**.style1**

**{**

**height: 340px;**

**width: 826px;**

**}**

**.style2**

**{**

**width: 132px;**

**}**

**.style3**

**{**

**width: 132px;**

**height: 35px;**

**color: #FF9900;**

**}**

**.style4**

**{**

**height: 35px;**

**}**

**.style5**

**{**

**width: 132px;**

**color: #FF9900;**

**}**

**.style6**

**{**

**width: 132px;**

**height: 38px;**

**color: #FF9900;**

**}**

**.style7**

**{**

**height: 38px;**

**}**

**</style>**

**</asp:Content>**

**CREATE PASS :**

**<html xmlns="http://www.w3.org/1999/xhtml">**

**<head id="Head1" runat="server">**

**<title></title>**

**<style type="text/css">**

**.newStyle15**

**{**

**background-color: #000080;**

**font-size: xx-large;**

**color: #FF0000;**

**}**

**.newStyle16**

**{**

**background-color: #FF9900;**

**color: #000080;**

**}**

**.newStyle17**

**{**

**font-size: x-large;**

**}**

**.style1**

**{**

**text-align: center;**

**height: 77px;**

**width: 434px;**

**margin-left: 232px;**

**}**

**.newStyle18**

**{**

**font-size: x-large;**

**}**

**.newStyle19**

**{**

**background-color: #FF9900;**

**font-size: x-large;**

**color: #000080;**

**font-family: "Franklin Gothic Book";**

**}**

**.newStyle20**

**{**

**font-size: x-large;**

**font-weight: bold;**

**color: #FF9900;**

**background-color: #000080;**

**}**

**.style2**

**{**

**background-color: #000080;**

**font-size: xx-large;**

**color: #FF0000;**

**font-weight: bold;**

**}**

**.newStyle21**

**{**

**font-size: xx-large;**

**}**

**.newStyle22**

**{**

**font-size: xx-large;**

**color: #FF9900;**

**text-decoration: blink;**

**}**

**.style3**

**{**

**font-size: xx-large;**

**color: #FF9900;**

**text-decoration: blink;**

**}**

**#form1**

**{**

**height: 416px;**

**width: 192px;**

**}**

**</style>**

**</head>**

**<body bgcolor="#000066" text="#ff0000">**

**<form id="form1" runat="server" style="width: auto; height: auto">**

**<div style="height: 77px; width: 463px; z-index: 1; left: 186px; top: 62px; position: absolute;"**

**<img src="pictures/right.png"**

**style="width: 312px; height: 211px; z-index: 1; left: 443px; top: 71px; position: absolute" /></div>**

**<p>**

**<asp:Login ID="Login1" runat="server" BackColor="Navy" BorderColor="Orange"**

**BorderWidth="1px" Font-Names="microsoft" Font-Size="15pt"**

**onauthenticate="Login1\_Authenticate"**

**style="z-index: 0; left: 326px; top: 133px; position: absolute; height: 216px; width: 315px"**

**CssClass="newStyle15" ForeColor="Orange"**

**LoginButtonImageUrl="~/pictures/reverse.jpg" PasswordLabelText="Password"**

**RememberMeText="Remember me next time" oninit="Login1\_Init"**

**onloggedin="Login1\_LoggedIn" onloggingin="Login1\_LoggingIn">**

**<TitleTextStyle BackColor="orange" Font-Bold="True" ForeColor="navy" />**

**</asp:Login>**

**<asp:LinkButton ID="LinkButton1" runat="server" ForeColor="#FF9900"**

**onclick="LinkButton1\_Click"**

**style="z-index: 1; left: 406px; top: 356px; position: absolute; height: 22px; width: 176px">REGISTER HERE</asp:LinkButton>**

**</p>**

**<div class="style1">**

**<span class="style2">**

**</span><span**

**class="style3">&nbsp; Online BusPass System</span></div>**

**<img src="pictures/left.png"**

**style="width: 411px; height: 265px; margin-right: 0px; z-index: 1; left: -22px; top: 108px; position: absolute; right: 415px;" /><asp:ImageButton**

**ID="admin" runat="server" ImageUrl="~/pictures/login\_icon.png"**

**onclick="ImageButton1\_Click"**

**style="z-index: 1; left: 439px; position: absolute; height: 53px; width: 78px; bottom: 45px;" />**

**</form>**

**</body>**

**</html>**

**LOGIN.aspx :**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.UI;

using System.Web.UI.WebControls;

using System.Data.SqlClient;

public partial class Login : System.Web.UI.Page

{

protected void Page\_Load(object sender, EventArgs e)

{

if (!this.IsPostBack)

ViewState["LoginErrors"] = 0;

}

protected void Login1\_Authenticate(object sender, AuthenticateEventArgs e)

{

if (YourValidationFunction(Login1.UserName, Login1.Password))

{

// e.Authenticated = true;

Login1.Visible = false;

}

else

{

e.Authenticated = false;

}

}

protected void Login1\_LoginError(object sender, EventArgs e)

{

if (ViewState["LoginErrors"] == null)

ViewState["LoginErrors"] = 0;

int ErrorCount = (int)ViewState["LoginErrors"] + 1;

ViewState["LoginErrors"] = ErrorCount;

if ((ErrorCount > 3) && (Login1.PasswordRecoveryUrl != string.Empty))

Response.Redirect(Login1.PasswordRecoveryUrl);

}

private bool YourValidationFunction(string UserName, string Password)

{

bool boolReturnValue = false;

string strConnection = "Data Source=BCASYSTEM065-PC;Initial Catalog=buspass;User ID=sa;Password=kbn;";

SqlConnection con = new SqlConnection(strConnection);

String SQLQuery = "SELECT UserName, Password FROM Login";

SqlCommand command = new SqlCommand(SQLQuery, con);

SqlDataReader Dr;

con.Open();

Dr = command.ExecuteReader();

while (Dr.Read())

{

if ((UserName == Dr["UserName"].ToString()) & (Password == Dr["Password"].ToString()))

{

boolReturnValue = true;

}

Dr.Close();

return boolReturnValue;

}

return boolReturnValue;

}

}

Firstpage.aspx

﻿using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.UI;

using System.Web.UI.WebControls;

using System.Data.SqlClient;

public partial class firstpage : System.Web.UI.Page

{

protected void Page\_Load(object sender, EventArgs e)

{

if (!this.IsPostBack)

ViewState["LoginErrors"] = 0;

}

protected void Login1\_Authenticate(object sender, AuthenticateEventArgs e)

{

if (YourValidationFunction(Login1.UserName, Login1.Password))

{

// e.Authenticated = true;

Login1.Visible = false;

}

else

{

e.Authenticated = false;

}

}

protected void Login1\_LoginError(object sender, EventArgs e)

{

if (ViewState["LoginErrors"] == null)

ViewState["LoginErrors"] = 0;

int ErrorCount = (int)ViewState["LoginErrors"] + 1;

ViewState["LoginErrors"] = ErrorCount;

if ((ErrorCount > 3) && (Login1.PasswordRecoveryUrl != string.Empty))

Response.Redirect(Login1.PasswordRecoveryUrl);

}

private bool YourValidationFunction(string UserName, string Password)

{

bool boolReturnValue = false;

string strConnection = "Data Source=BCASYSTEM065-PC;Initial Catalog=buspass;User ID=sa;Password=kbn;";

SqlConnection con = new SqlConnection(strConnection);

String SQLQuery = "SELECT UserName, Password FROM Login";

SqlCommand command = new SqlCommand(SQLQuery, con);

SqlDataReader Dr;

con.Open();

Dr = command.ExecuteReader();

while (Dr.Read())

{

if ((UserName == Dr["UserName"].ToString()) & (Password == Dr["Password"].ToString()))

{

boolReturnValue = true;

Response.Redirect("first.master");

}

Dr.Close();

return boolReturnValue;

}

return boolReturnValue;

}

protected void LinkButton1\_Click(object sender, EventArgs e)

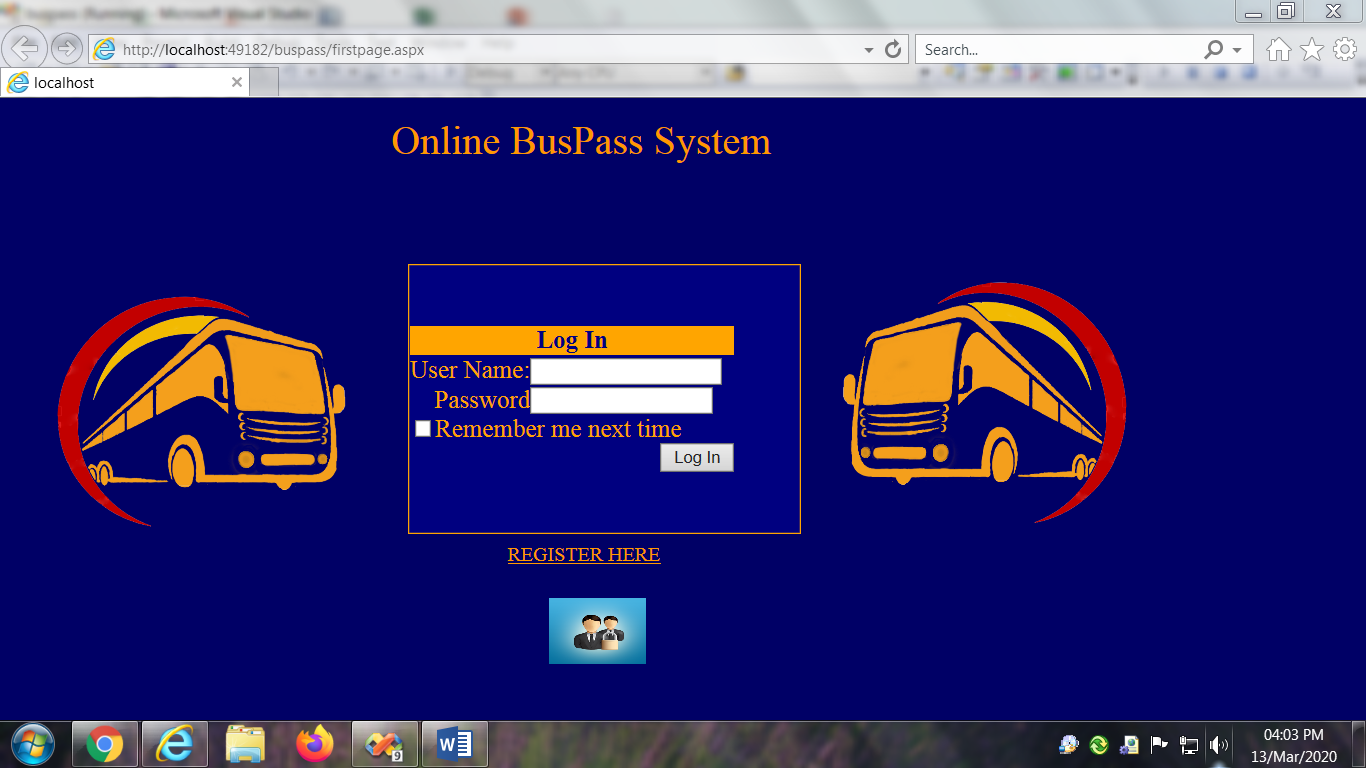
{

Response.Redirect("Register.aspx");

}

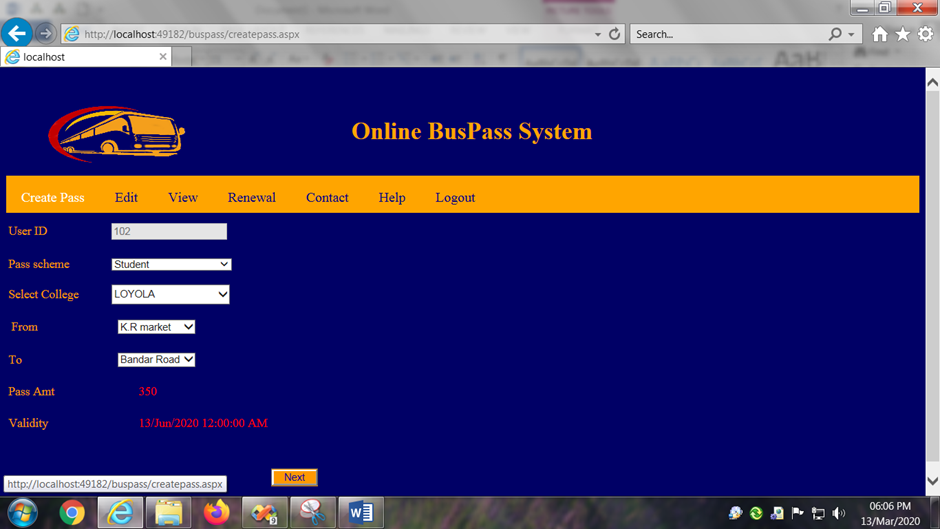
}

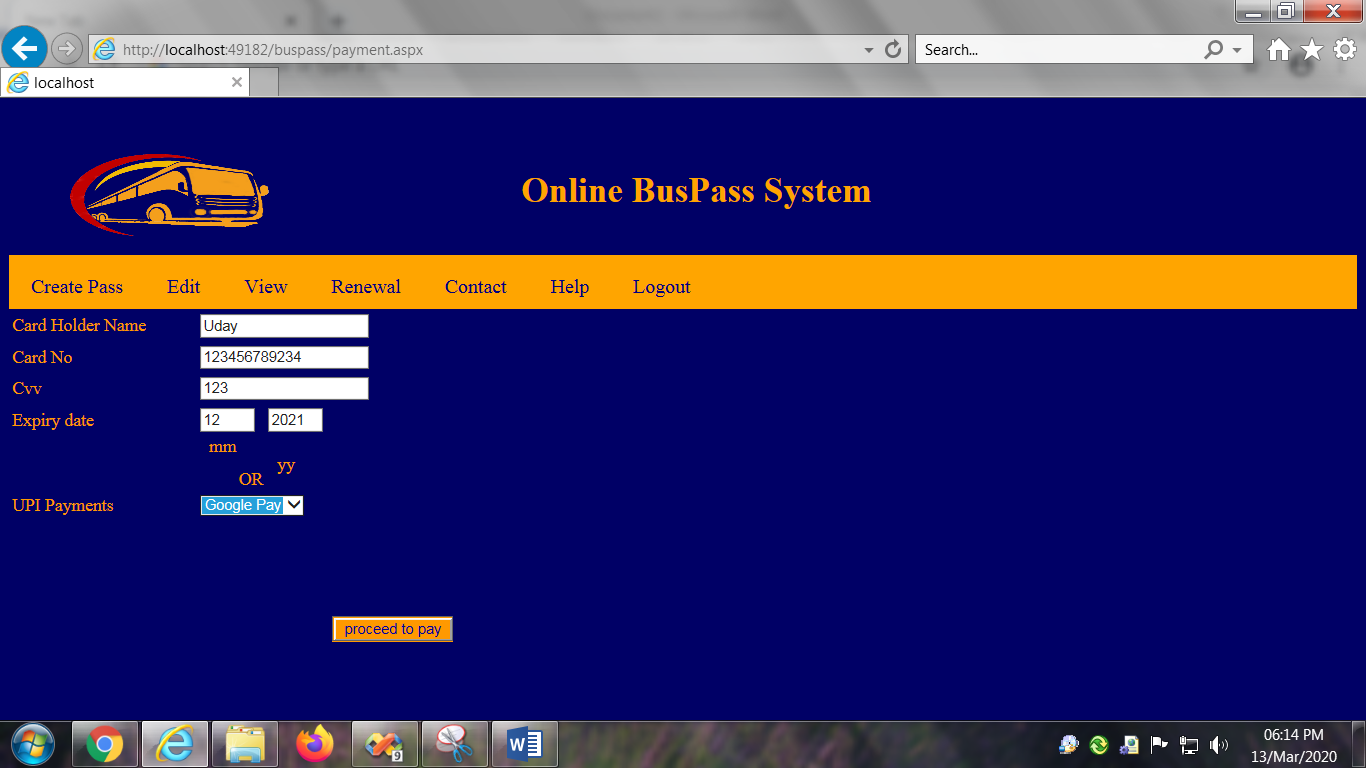
**OUTPUT SCREENS :**

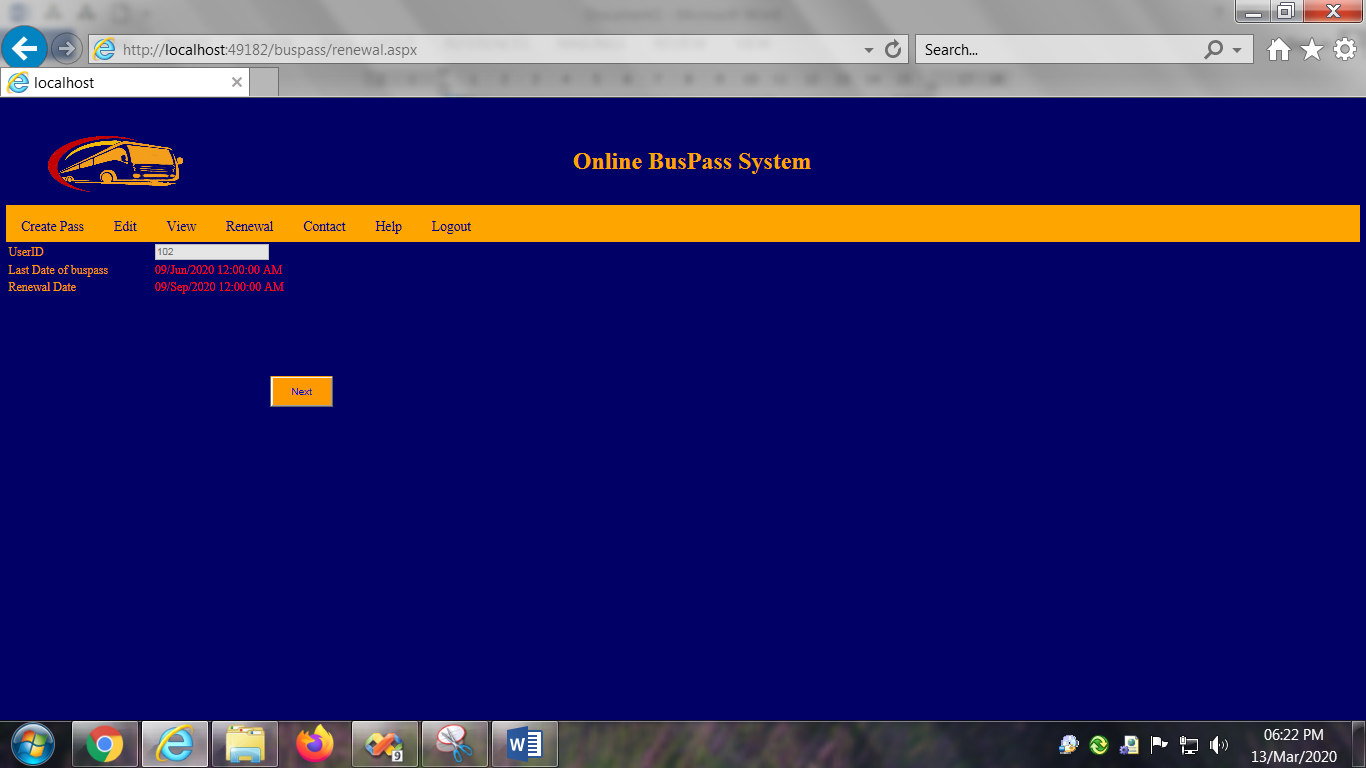


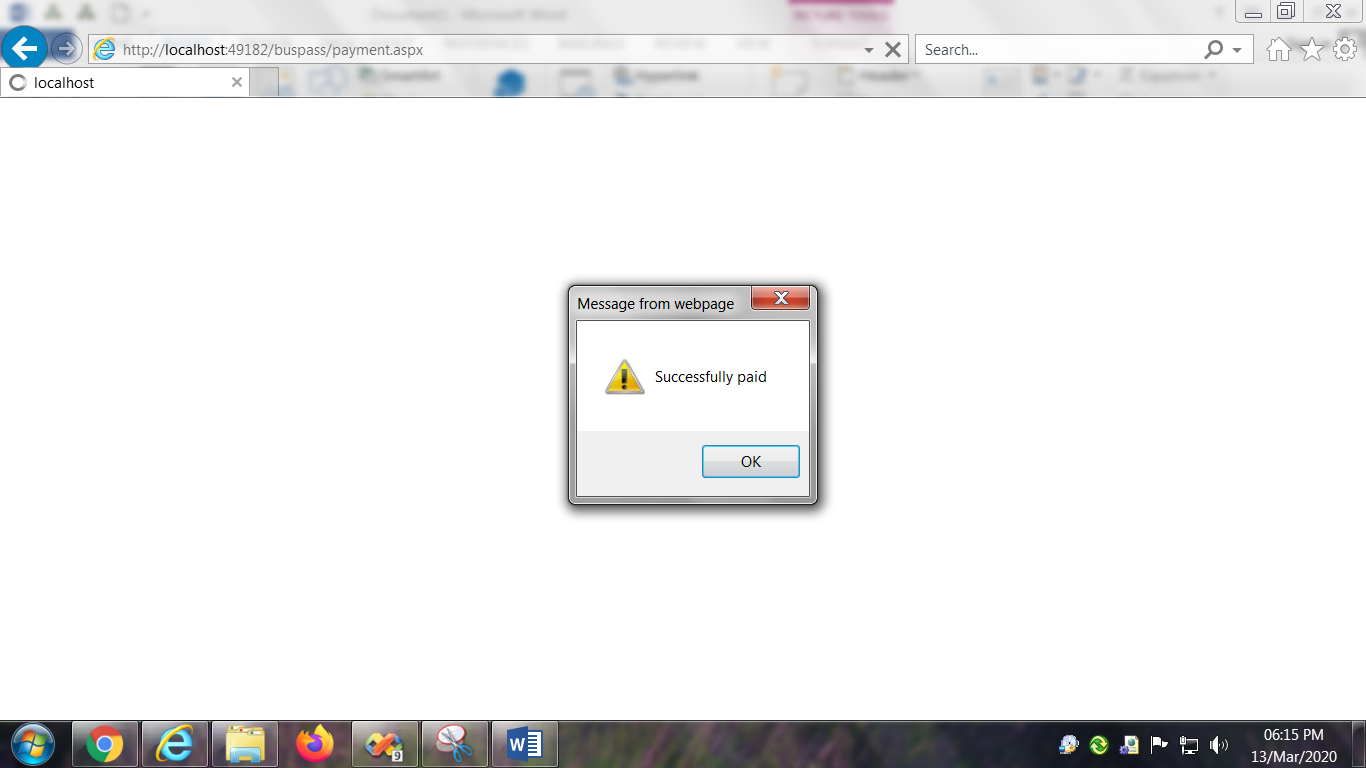


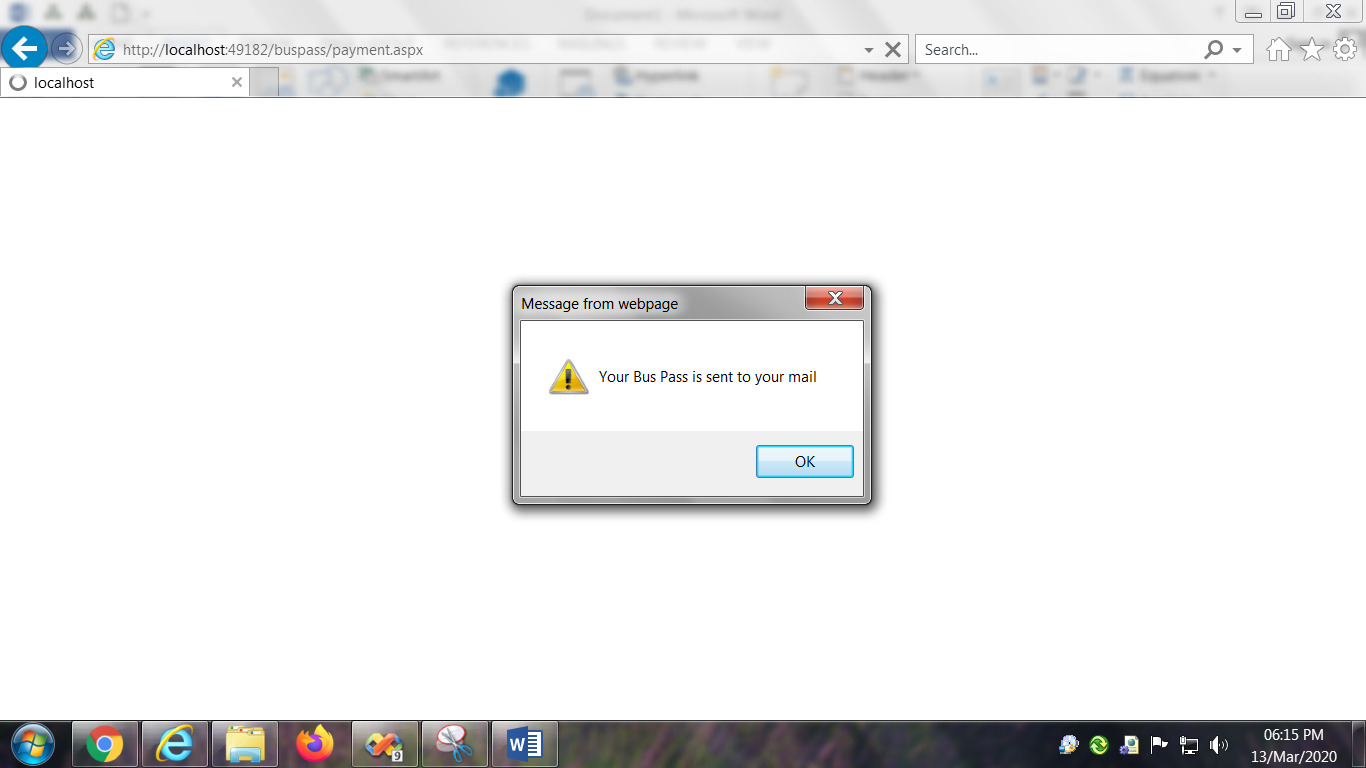












**CONCLUSION :**

At last, We conclude that using this online bus pass system the people can get there bus pass in a very easy way without standing in a queue in bus station.

**FUTURE SCOPE :**

The system is intended to develop an application to perform functions like accessing basic information for authentication and provide passes without the need to wait in a queue in the bus station. Online bus pass  system is for students and other daily travellers to get pass through online.