1. Introduction:

1.1 Literature Survey

The ATM System is the project which is used to access their bank accounts in order to make cash withdrawals. Whenever the user need to make cash withdraws, they can enter their PIN number (personal identification number) and it will display the amount to be withdrawn in the form of 100's 500's and 1000's. Once their withdrawn was successful, the amount will be debited in their account.

The ATM will service one customer at a time. A customer will be required to enter ATM Card number, personal identification number (PIN) – both of which will be sent to the database for validation as part of each transaction. The customer will then be able to perform one or more transactions. Also customer must be able to make a balance inquiry of any account linked to the card.

The ATM will communicate each transaction to the database and obtain verification that it was allowed by the database. In the case of a cash withdrawal, a second message will be sent after the transaction has been physically completed (cash dispensed or envelope accepted). If the database determines that the customer's PIN is invalid, the customer will be required to re-enter the PIN before a transaction can Now a day, there is no proper security for ATM machines. Robbery of the ATM machines has been increased widely. By using the existed technology ATM machines are not safe in order to provide proper security for money. So it is proposed a new technology which can overcome this problem. Vibration detection sensors, microcontroller and GSM modem are used here to make up the problem. proceed.
If a transaction fails for any reason other than an invalid PIN, the ATM will display an explanation of the problem, and will then ask the customer whether he/she wants to do another transaction.

The ATM will provide the customer with a printed receipt for each successful transaction, showing the date, time, machine location, type of transaction, account(s), amount, and ending and available balance(s) of the affected account ("to" account for transfers).

1.2 Existing System:

Present working scenario of the organization is something like this: the present working of the organization is outdated and the methods in the working of organization are not efficient. The information of the ATM’s are recorded on paper and any record books.

The present system is slow and involves much clerical again in any nearest ATM man power is required which is a time consuming process.

1.3 ProposedSystem:

The proposed system provides a user friendly interface and more secure. This system computerizes all the details of ATMs and its working conditions and the money status. Large volumes of data can be stored with case. Maintenance of file is flexible. Records stored are updated now and then. Stored data and procedures can be easily edited. Reports can be generated with case. Accurate calculations are made. Less manpower require.

SYSTEM REQUIREMENTS

Hardware Requirements:

Processor: Intel core 13 and above

RAM: 128 MB

Monitor: VGA monitor

Keyboard and mouse

Celeron or higher processor

2GB RAM

40GB hard disk

Software Requirements:

Operating system: Windows 11

Database: SQL server

Server side technology: ASP.Net

Server side scripting: ASP

Client side scripting: HTML

Web browser: Google chrome, Internet Explorer etc

2.REQUIREMENT SOFTWARE SPECIFICATION:

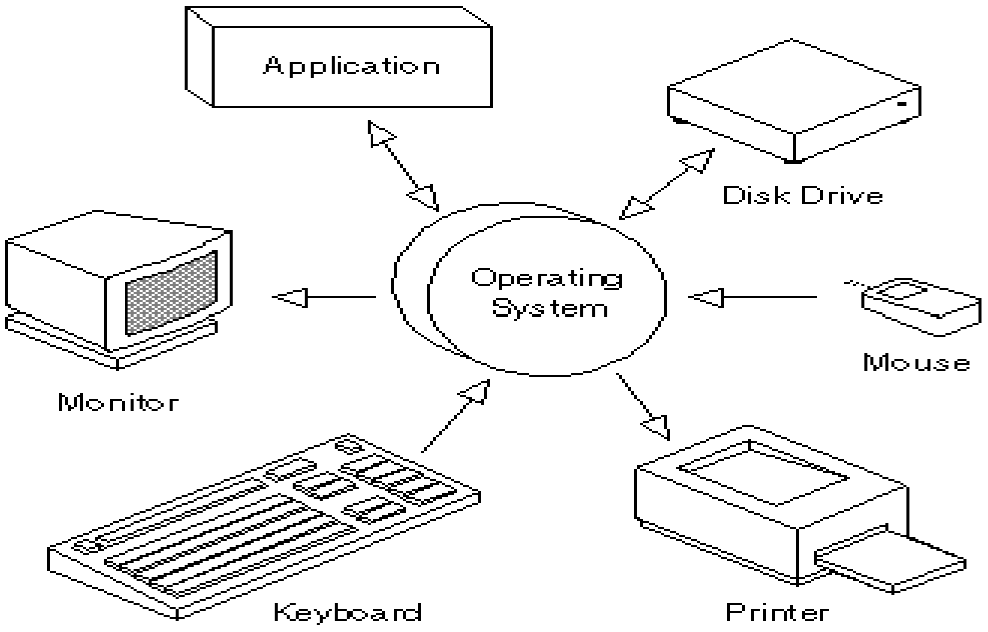
Definitions, Acronyms and Abbreviations:

GUI: Graphical User Interface
HTML: Hypertext Markup language. is the predominant mark-up language for web pages.
C#.NET : **C#** (pronounced "C-sharp") is an object-oriented programming language from Microsoft that aims to combine the computing power of C++ with the programming ease of Visual Basic. **C#** is based on C++ and contains features similar to those of Java.
SRS-Software Requirement Specification.

Overall Description:

Operational Environment:

This application is designed to run on Windows environment. Also it will be compatible with the Windows 2000 or higher version, IE 6.0. The only requirement to use this product would be the internet connection.



Languages:

The application is implemented using the following:

- HTML

- C#
- SQL server 2000 as a database

User Characteristics:

In common the customer should be familiar and must have the knowledge of windows operating system. Basic computer knowledge of using keyboard, mouse and common windows environment is required. The primary user of the application is the admin who maintains the software or product he should know about the data present and the data he want to view, so that he can efficiently use the function of the proposed system. The software is GUI based to reduce the user non-friendliness towards the software.
There are various kinds of users for the product. Usually web products are visited by various users for different reasons.

System Features:

In general the user should be familiar and must have the knowledge of windows operating system. The primary user is admin who maintains the software or product he should know about the data present and the data he want to view, so that he can effectively use the function of the proposed system. The software is GUI based to reduce the user non-affability towards the software. It is also provided with the proper information message when clicked on the button.

3.Specific Requirements:

External Interface Requirements:

- User Interface: Accomplishes via mouse and keyboard input to the GUI based forms.
- Software Interface: The product requires to runtime C# environment in server machine.

Functional Requirements:

Functional requirements are statements of the services that the system must provide or are descriptions of how some computations must be carried out. The plan for implementing functional requirements is detailed in the system design.

- Admin is provided with Username and Password to avoid unauthorized access.
- Basic and advance admin facilities like add/update/delete Product details are provided.
- Manages details about the citizen and vehicles.
- Manages registered vehicles details.
- Manages queries about road.
- Back up/report of data such as citizen, payment, advance payment details

Non-Functional Requirements:

A non-functional requirement is a requirement that specifies criteria that can be used to judge the operation of a system, rather than specific behaviours. This should be contrasted with functional requirements that define specific behaviour or functions. The plan for implementing non-functional requirements is detailed in the system architecture.

- The application is available from several locations and it is accessible 24*7 with the help of internet
- The code is easily understood and read because of the structure and coding style of the application.
- Various components are provided in order to improve the performance at peak time

User Interface:

The user interface allows the users to remotely access the system via several applications Users will be able to use the service through applications such as Mozilla Firefox, Microsoft Internet Explorer, etc. Allow the user to access the information fast and easily from remote locations.

1.Compatibility: The software is compatible with Windows XP.It also works well with Windows 2000 or higher .It requires Microsoft Internet Explorer 5.0 or above with Wamp Server.

2.Portability: The software is extremely portable in the sense that it can be run on any machine with a web-browser.

3.Acceptance Criteria: The system must work well and compile with all the requirements and constraints stated above. All conditions defined by the need user by the end user are to be satisfied. The system should satisfy all the requirement and constraints of the "PERFECT WHOLESELLER" and must work well according to necessity.

Design Constraints:

In case of any errors Error Handling mechanisms have been provided

- Without proper authorization the data in the database can't be modified.
- Complete validation has been done so that no mandatory fields are empty by chance if any field is left empty then appropriate alert message will be displayed.
- In case of any errors Error Handling mechanisms have been provided

Feasibility Study:

Preliminary investigations examine project feasibility; the likelihood of how the system is will be useful to the organization. Three tests of feasibility-all equally important are studied.

Three tests of feasibility-all equally important are studied.

1. Operational feasibility:

Operational feasibility is based on whether the user will accept the developed application and will satisfy the needs of the user.

2. Technical Feasibility:

Technical feasibility is based on whether the proposed system is capable of holding the data required for the application.

3. Financial Feasibility:

This concept is based on the financial status of the owner such that he must have sufficient budget to develop the proposed system which is cost effective.

4.SYSTEM REQUIREMENTS:

System specification forms the foundation on which the architecture, design, and implementation of a software is built. Documents containing system specifications are critical because major expenses come along with having to fix the implementation of incorrect requirements as a result of not having a specification document on hand. System specification documents can thus be defined as the requirements documentation that formally specifies the system-level requirements of a software application.

- performance levels
- reliability
- quality
- interfaces
- security and privacy
- constraints and limitations
- functional capabilities
- data structures and elements

CLIENT SIDE:

HARDWARE SPECIFICATION:

Sl.No.	Hardware	Description
1	PROCESSOR	Core 2 duo or higher
2	RAM	1gb Mb or more

SOFTWARE SPECIFICATION:

Operating System	Window XP & above	Operating System

Front	Visual studio 2010	Front End

SERVER SIDE:

HARDWARE SPECIFICATION:

Processor	Intel p-IV & above
RAM	512 MB & above
Hard disc	10GB & above

COMMUNICATION INTERFACE:

The admin or user must connect to the Internet to access the NO
Broadband Internet :NO

Object-Oriented Analysis Design:

Object-oriented analysis and design (OOAD) is a software engineering approach that models a system as a group of interacting objects. Each object represents some entity of interest in the system being modelled, and is characterized by its class, its state (data elements), and its behaviour. Various models can be created to show the static structure, dynamic behaviour, and run-time deployment of these collaborating objects. There are a number of different notations for representing these models, such as the Unified Modelling Language (UML).

Object-oriented analysis (OOA) applies object-modelling techniques to analyze the functional requirements for a system. Object-oriented design (OOD) elaborates the analysis models to produce implementation specifications. OOA focuses on *what* the system does, OOD on *how* the system does it.

Analysis:

Object-oriented analysis (OOA) is the process of analyzing a task (also known as a problem domain) to develop a conceptual model that can then be used to complete the task. A typical OOA model would describe computer software that could be used to satisfy a set of customer-defined requirements. During the analysis phase of problem-solving, the analyst might consider a written requirements statement, a formal vision document, or interviews with stakeholders or other interested parties. The task to be addressed might be divided into several subtasks (or domains), each representing a different business, technological, or other areas of interest. Each subtask would be analysed separately. Implementation constraints, (e.g., concurrency, distribution, persistence, or how the system is to be built) are not considered during the analysis phase; rather, they are addressed during object-oriented design (OOD).

The conceptual model that results from OOA will typically consist of a set of use cases, one or more UML class diagrams, and a number of interaction diagrams. It may also include some kind of user interface mock-up.

5.Design of the System:

Introduction:

The purpose of the design phase is to plain a solution of the problem specified by the requirements document. This phase is the first step moving the problem domain to the solution domain. It involves the process, in which conceiving, planning and the carrying out the plain by generating the necessary report and inputs, in other words, the design phase act as a bridge between SRS and implementation phase. The design of the system is perhaps the most critical factor affecting the quality of the software, and has a major impact on the later phase, particularly the testing and maintenance.

Design System:

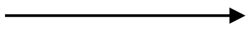
System design is concerned with how the system functionality is to be provided by the different components of the system Design is the key phase of any project. It is the first step in moving from problem domain to the solution domain. It may be defined as “The process of applying various techniques and principles for the purpose of defining a device, a process, or a system insufficient detail to permit its physical realization”. The input to design phase is the specifications of the system to be designed. The output of the top level design is the architectural design, or the system design for the software system to be built. A design should be very clear, verifiable, complete, traceable, efficient and simple.

Data Flow Diagram:

The data flow diagram (DFD) is one of the important modelling tools. It shows the user of the data pictorially. DFD represents the flow of the data between different transformations and processes in the system. The data flow diagram shows logical flow of the data. It represents the functional dependencies within a system. It shows output values in a computation are derived from input values. It is a simple pictorial representation or model for system behaviour. It specifies, “What is to be done but not how is to be done”. It describes the logical structure of the system. It relates data information to various processes of the system. It follows top-down approach.

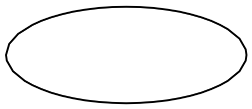
<u>Data Flow Diagram Notations:</u>

<u>Data Flow:</u>



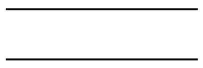
It may be from file-to-file or file-to-process or process- process. It is generally in terms of attributes. There may be either an input data flow or output data flow.

Functional Processing:



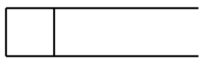
The process is nothing but the transformation of data it starts with the subject and it has the verb followed by the subject.

Data store:

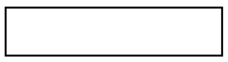


It includes file, data base and repository. To parallel lines represent it or a one end closed rectangle

Or



Actor/source/sink:



The files which are outside the system and used by the process or the processes of the system .Generally Source/Sink in the actor.

Objectives

• To graphically document boundaries of a system.
• To provide hierarchy breakdown of the system.
• To show movement of information between a system and its environment.
• To document information flows within the system.
• To aid communication between users and developers.

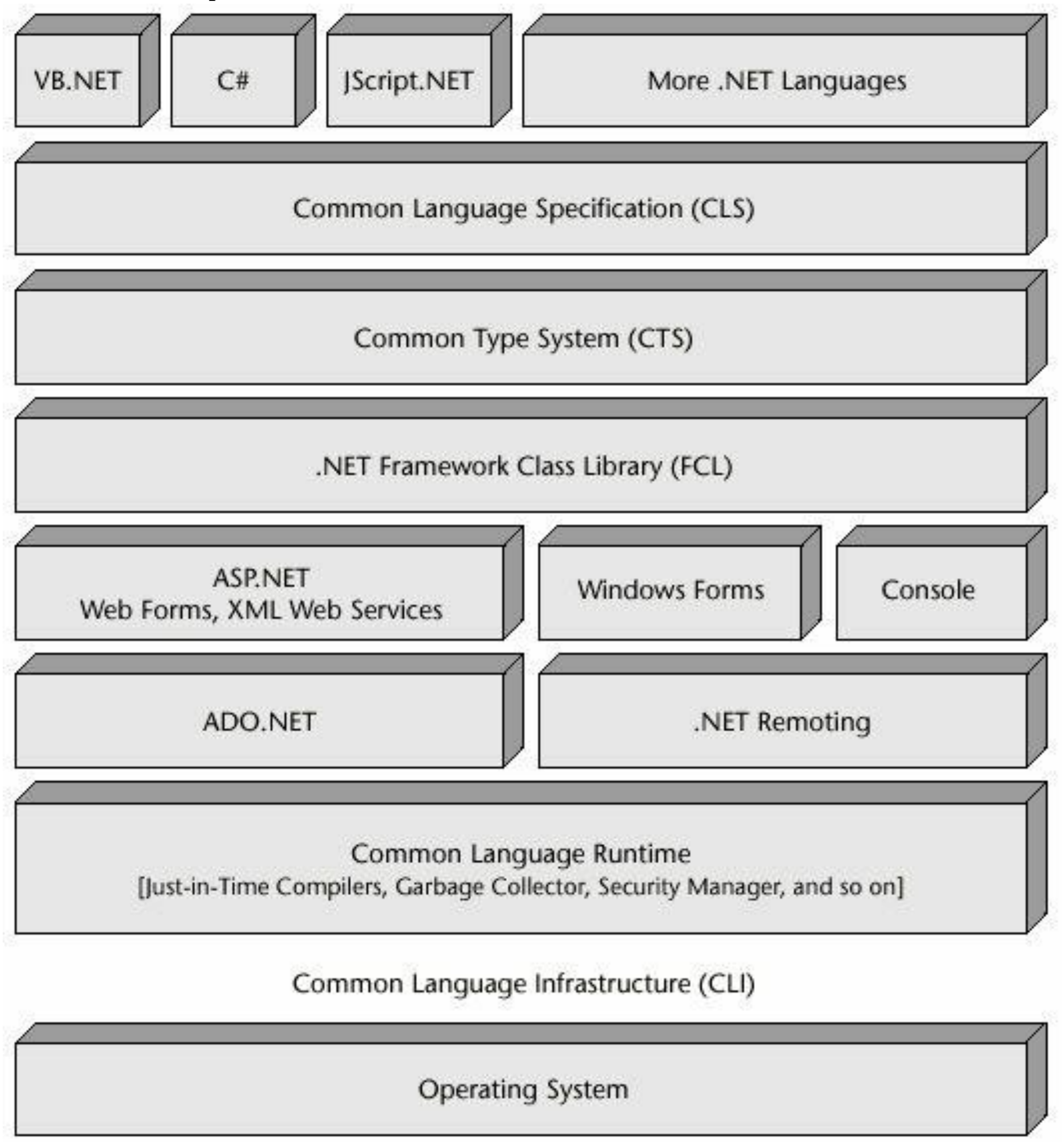
Architecture Diagram :

The initial design process of identifying the sub-system and establishing a frame work for sub-system control and communication is called architectural design and the output of this design process is a description of the software architecture. The user interface, functional process logic, product data storage and data access are developed and maintained. This diagram is very important to understand the overall concept of system interface,.

Logical Design:

The logical design describes the detailed specification for the system, describing its features, effective communication and the user interface requirements. The logical system of a proposed system should include the following.

1. External System Structure.
2. Relationship between all the activities.


VB.NET
C#
JScript.NET
More .NET Languages
Common Language Specification (CLS)
Common Type System (CTS)
.NET Framework Class Library (FCL)
ASP.NET Web Forms, XML Web Services
Windows Forms
Console
ADO.NET
.NET Remoting
Common Language Runtime [Just-in-Time Compilers, Garbage Collector, Security Manager, and so on]
Common Language Infrastructure (CLI)
Operating System

3. The physical construction and all the activities.
4. Global data.
5. Control flow.
6. Derived program structure.

Design Principles:

Basic design principles that enable the software engineer to navigate the design process are:

- The design process should not suffer from "Tunnel vision".
- The design should be traceable to analysis model.
- The design should not reinvent the wheel.
- The design should minimize the intellectual distance between the software and the problem, as it exists in the real world.
- The design should exhibit uniformity and integrity.
- The design should be structured to accommodate changes.
- The design not coding, the coding is not a design.
- The design should be assessed for the quality, as it is being created, not after the fast.
- The design should be reviewed to minimize the conceptual errors.

6.IMPLEMENTATION:

Introduction:

Implementation is the process of converting a new revised system design into operation. The objective is to put the new revised system, which has been tested into operation while holding costs, risks and personal irritation to the minimum. A critical aspect of the implementation process is to ensure that there will be no description in the function of the organization. The best methods for gaining control while implementation any new system would be to use well planned test files for testing all new programs. Another factor to be considered in the implementation phase in the acquisition of the hardware and software. Once the software is developed for the system and testing is carried out, it is the process of making the newly designed system fully operational and consistent in performance.

Introduction to technologies used in this project:

Implementation is the realization of an application, or execution of a plan, idea, model, design, specification, standard, algorithm, or policy and it is a process of having the systems personnel check out and put new equipment into use, train users, install new application a and construct any files of data needed to use it.

Why You Need .NET?

.NET security is not an island of technology, but a slice of a larger entity called the .NET Framework. Basic understanding of the .NET Framework is required before attempting .NET security programming. This chapter presents the basic concepts of the .NET Framework architecture and programming. This is an overview and is not intended to replace the independent study required for a mastery of this subject. (For a comprehensive discussion on the .NET Framework from a developer's perspective, I recommend *.NET Framework Essentials* by Thun L. Thai and Hoang Q. Lam, O'Reilly & Associates, February 2002.)
Microsoft .NET is not just a different spin on the Win32 operating model. Furthermore, despite reports to the contrary, it is not Java in wolf's clothing. You will never understand or adequately explain .NET simply by comparing it to existing products. .NET is new. As such, .NET introduces a fresh operating modality and perspective on computing software and devices.

Are there similarities to Java? Are there similarities to Win32? Yes, but there are many more differences. Successfully programming in .NET requires embracing this new technology as new and fully understanding the many things that make .NET unique. When object-oriented languages were introduced, developers faced a similar challenge and, unfortunately, mindset. Many programmers quickly learned the syntax and ported their C application to C++ or Small Talk.

However, without the requisite understanding of object-oriented programming, these new applications were procedural programs draped in the syntax of an object-oriented language. Some developers invested the time to learn object-oriented programming—not just the syntax, but the philosophy and intent. Their resulting applications were true object-oriented programs that provided all the benefits envisioned for the new programming modality. Similarly, understanding the philosophy and architecture of .NET is essential for creating applications that offer new solutions. Some industry analysts assert that Microsoft has gambled the company on .NET. I would not agree. .NET does represent a massive investment. However, Microsoft is a diversified and multibillion dollar company with many products and a sizable market share in many segments of the software industry. In addition, Microsoft is no longer simply a software company, having expanded into many markets outside their traditional stronghold.

But recognizing that Microsoft is not teetering on a precipice named .NET does not diminish the importance of .NET. .NET does represent a new philosophy in product development. From .NET will emerge an entirely new family of products that will drive Microsoft sales into the stratosphere over the next 5 to 10 years. If the .NET initiative fails, or more likely is adopted slowly, Microsoft will recover and continue, although maybe with a little less luster. Importantly, .NET allows Microsoft to escape the Windows conundrum. Although Windows has been enormously successful, it is still a box. .NET helps Microsoft emerge from that box and develop applications for a universal audience. This new opportunity will fuel growth not just for Microsoft, but for software developers everywhere.

I attended the formal launch of Microsoft .NET at the Professional Developers Conference in Orlando, Florida several years ago. William Gates III (aka Bill) was the keynote speaker. Part of his speech included an entertaining video. The video portrayed .NET as a new standard that will allow software to run anywhere, at anytime, on any platform, and on devices large and small.

- **Anywhere.** This has reported many times, but it is worth repeating: "Microsoft was late to realize the importance and then embrace the Internet." Recently, Microsoft has been making up for that late start. .NET marks the next major step in that journey. The Internet is not an adjunct of .NET, but is interwoven seamlessly into the product. The Internet was planned, integrated, and implemented into .NET—including the embracing of open standards such as XML and HTTP. Essentially, any platform that offers a browser that understands XML or HTML is a potential .NET client.
- **Anytime.** The Internet is open 7 days per week and 24 hours per day. The Internet never closes. Since .NET leverages the Internet, .NET applications such as a Web service are fully accessible at anytime.

- **Any platform.** .NET is a multi-language and multiplatform operating environment. Compare this to Java, which is single-language and multiplatform. .NET offers C#, Visual Basic .NET, and many more .NET-compliant languages. Programming in .NET does not require learning an entirely new language. To program to the Java Virtual Machine (JVM) requires learning the Java language. For many, this is a substantial drawback. The common language runtime is the common runtime of all .NET languages. In addition, Microsoft publishes the Common Language Infrastructure (CLI) document, which is a set of guidelines for creating a .NET common language runtime for any platform, such as Linux. To view one such initiative, visit www.go-mono.com. In the future, developers can create .NET applications in Windows and run them in Linux, Unix, Macintosh, or any platform that offers a common language runtime. server-side language, running C# scripts on your local machine requires installing a server on your local machine.

C# is free software released under the C# License; however it is incompatible with the GNU General Public License (GPL), due to restrictions on the usage of the term C#. It is a widely-used general-purpose scripting language that is especially suited for web development and can be embedded into HTML. It generally runs on a web server, taking C# code as its input and creating web pages as output. It can be deployed on most web servers and on almost every operating system and platform free of charge. C# is installed on more than 20 million websites and 1 million web servers.

6.1. SQL:

INTRODUCTION TO SQL SERVER

SQL Server is a Relational Database Management System (RDBMS) that runs exclusively under the Windows operating system. One benefit of using Windows exclusively is that you can send and receive E-mail messages based on SQL Server "events" and you can also let the operating system handle login security.The data base is an organized collection of data. A database management system (DBMS) such as Access, FileMaker Pro, Oracle or SQL Server provides you with the software tools you need to organize that data in a flexible manner. It includes facilities to add, modify or delete data from the database, ask questions (or queries) about the data stored in the database and produce reports summarizing selected contents. MySql is a multi-threaded, multi-user SQL database management system (DBMS). The basic program runs as a server providing multi-user access to a number of databases. Originally financed in a similar fashion to the JBoss model, MySql was owned and sponsored by a single for-profit firm, the Swedish company MySQLAB now a subsidiary of Sun Micro system , which holds the copyright to most of the codebase.

Database Evolution:

SQL was invented back in the 1960's by E.F. Cod of IBM. in order to increase data integrity and reduce repetitive data. RDBMS systems didn't appear until the late 70's when Sybase and Oracle introduced systems. These systems existed on mainframes at the time.

ANSI-SQL came to be in the 1980's. This was important because it meant that disparate systems could communicate through an agreed set of standards. There are different levels of ANSI-SQL compliance. Almost every major RDBMS today is entry level compliant, including SQL Server 2000. Every RDBMS has its own flavour of SQL that complements ANSI-SQL with proprietary elements. SQL Server's flavour of SQL is known as Transact SQL (T-SQL).

SQL Server was originally a Sybase product. Microsoft bought the product outright from Sybase and by version 7.0, the version prior to 2000, all the code had been rewritten by Microsoft's programming gurus.

FEATURES OF SQL:

- It is simple English like language and uses simple commands such as SELECT, CREATE, DROP etc.
- It is not having condition loops, variables and most of the commands are single line commands.
- To implement application logics, SQL has got extension language popularly called as PL/SQL (Procedural language of sql).
- One of the key features of sql server is the XML support. XML has Grown to be standard technology for organizations that share data on the web. Now with sql server 2000 XML documents can be retrieved directly from the database and it provides various ways to retrieve data in XML format.

The entire SQL has been divided into 4 major categories.

1. Data Manipulation Language.
2. Data Definition Language.
3. Transaction Control language.
4. Data Control Language.

Security: View are basically used as a part of security, means in many organizations ,the end user will never be given original tables & all data entry will be done with the help of views only. But the data base administrator will be able to see everything because all the operations done by the different users will come to the same table.

7.Codings:

```
<%@ Master Language="C#" AutoEventWireup="true" CodeFile="mainmaster.master.cs"
Inherits="mainmaster" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<!--

       Serious Face by nodethirtythree + Templated.org
       http://templated.org/ | @templatedorg
       Released under the Creative Commons Attribution 3.0 License.

       Note from the author: These templates take quite a bit of time to conceive,
       design, and finally code. So please, support our efforts by respecting our
       license: keep our footer credit links intact so people can find out about us
       and what we do. It's the right thing to do, and we'll love you for it :)

-->
<html xmlns="http://www.w3.org/1999/xhtml">
    <head>
        <meta name="keywords" content="" />
        <meta name="description" content="" />
        <meta http-equiv="content-type" content="text/html; charset=utf-8" />
              <title>ATM detector</title>
        <link href="http://fonts.googleapis.com/css?family=Bitter" rel="stylesheet"
type="text/css" />
              <link rel="stylesheet" type="text/css" href="style.css" />
    </head>
    <body>
              <div id="bg">
                     <div id="outer">
                            <div id="header">
                                   <div id="logo">
                                          <h1>
                                                 <a href="#">ATM Detector</a>
                                          </h1>
                                   </div>
                                   <!--<div id="search">
                                          <form action="" method="post">
                                                 <input class="text" name="search"
size="32" maxlength="64" /><input class="button" type="submit" value="Search" />
                                          </form>
                                   </div>-->
                                   <div id="nav">
                                          <ul>
                                                 <li class="first active">
                                                        <a href="mainhome.aspx">Home</a>
                                                 </li>
                                                 <li>
                                                        <a
href="userlogin.aspx">Login(user)</a>
                                                 </li>
                                                 <li>
                                                        <a
href="adminlogin.aspx">Login(admin)</a>
```

```
                                                </li>
                                                <!--<li>
                                                       <a href="#">Support</a>
                                                </li>
                                                <li>
                                                       <a href="#">Blog</a>
                                                </li>-->
                                                <li>
                                                       <a href="aboutpage.aspx">About</a>
                                                </li>
                                                <li class="last">
                                                       <a
href="contactpage.aspx">Contact</a>
                                                </li>
                                         </ul>
                                         <br class="clear" />
                                  </div>
                           </div>
                           <div id="banner">
                                  <img src="images/pic1.jpg" width="980" height="172"
alt="" />
                           </div>
                           <div id="main">
                                  <!--<div id="sidebar">
                                         <h3>
                                                Auctor lacus
                                         </h3>
                                         <ul class="linkedList">
                                                <li class="first">
                                                       <a href="#">Vitae et suspendisse
integer dolore</a>
                                                </li>
                                                <li>
                                                       <a href="#">Molestie magnis sed
amet nascetur</a>
                                                </li>
                                                <li>
                                                       <a href="#">Augue sagittis
dolor</a>
                                                </li>
                                                <li>
                                                       <a href="#">Orci lectus leo
urna</a>
                                                </li>
                                                <li>
                                                       <a href="#">Odio volutpat
vivamus</a>
                                                </li>
                                                <li class="last">
                                                       <a href="#">Porttitor fermentum
amet commodo</a>
                                                </li>
                                         </ul>
                                         <h3>
                                                Varius
                                         </h3>
                                         <div class="dateList">
```

```
                                                <ul class="linkedList">
                                                       <li class="first">
                                                              <a href="#">Sapien et
nascetur</a>
                                                       </li>
                                                       <li>
                                                              <a href="#">Nascetur sed
vivamus</a>
                                                       </li>
                                                       <li>
                                                              <a href="#">Nisi amet et
sagittis</a>
                                                       </li>
                                                       <li>
                                                              <a href="#">Duis cras lorem
odio</a>
                                                       </li>
                                                       <li class="last">
                                                              <a href="#">Sed consequat
purus</a>
                                                       </li>
                                                </ul>
                                         </div>
                                  </div>-->
                                  <div id="content">
                                         <div id="box1">
                       <br />
                                                <!--<h2>
                                                       Welcome to Serious Face
                                                </h2><!--<img class="left"
src="images/pic2.jpg" width="184" height="184" alt="" />-->
                                                <p>
        <asp:ContentPlaceHolder id="ContentPlaceHolder1" runat="server">

        </asp:ContentPlaceHolder>
</p>
<br />
<br />
<br />
<br />
<br />
<br />
<br />

                                         </div>
                                         <div id="box2">
                                                <h3>
                                                       About Project
                                                </h3>
                                                <ul class="imageList">
                                                       <li class="first">
                                                              <!--<img class="left"
src="images/pic2.jpg" width="72" height="72" alt="" />--> <span>Iaculis risus lectus
tortor libero nisl quisque nunc mauris parturient amet suspendisse in porta.</span>
                                                       </li>
                                                       <li>
```

```
                                                                <!--<img class="left" 
src="images/pic2.jpg" width="72" height="72" alt="" />--> <span>Lacinia tristique dis 
cras mattis ultricies massa leo metus rhoncus augue nascetur suscipit velit.</span>
                                                        </li>
                                                        <li class="last">
                                                                <!--<img class="left" 
src="images/pic2.jpg" width="72" height="72" alt="" />--> <span>Nunc integer sed arcu 
vitae nascetur placerat magnis donec proin fusce neque elementum nulla.</span>
                                                        </li>
                                                </ul>
                                        </div>
                                        <div id="box3">
                                                <!--<h3>
                                                        Aenean accumsan
                                                </h3><img class="top" 
src="images/pic1.jpg" width="360" height="155" alt="" />-->
                                                <p>
                                                        Parturient cras orci augue 
parturient. Cursus donec elementum consectetur consequat. Mattis ultricies 
                                                        amet consequat magna non vel 
placerat lorem ipsum dolore sit amet nullam consequat. Lacinia tristique 
                                                        dis cras mattis ultricies.
                                                </p>
                                        </div>
                                        <br class="clear" />
                                </div>
                                <br class="clear" />
                        </div>
                </div>
                <div id="copyright">
                        &copy; ATM Detector | All rights reserved <!--<a 
href="http://templated.org/free-css-templates/seriousface/"></a> by <a 
href="http://nodethirtythree.com">nodethirtythree</a> + <a 
href="http://templated.org/">Templated.org</a>-->
                </div>
        </div>
    </body>
</html>
```

About page.aspx

```
<%@ Page Title="" Language="C#" MasterPageFile="~/mainmaster.master" 
AutoEventWireup="true" CodeFile="aboutpage.aspx.cs" Inherits="aboutpage" %>

<asp:Content ID="Content1" ContentPlaceHolderID="ContentPlaceHolder1" Runat="Server">
</asp:Content>
```

Admin_home.aspx

```
<%@ Page Title="" Language="C#" MasterPageFile="~/adminmaster.master"
AutoEventWireup="true" CodeFile="admin_home.aspx.cs" Inherits="admin_home" %>

<asp:Content ID="Content1" ContentPlaceHolderID="ContentPlaceHolder1" Runat="Server">
Welcome Admin
</asp:Content>
```

Adminlogin.axps

```
<%@ Page Title="" Language="C#" MasterPageFile="~/mainmaster.master"
AutoEventWireup="true" CodeFile="adminlogin.aspx.cs" Inherits="adminlogin" %>

<asp:Content ID="Content1" ContentPlaceHolderID="ContentPlaceHolder1" Runat="Server">
<form id="form1" runat="server">
    <table style="width: 100%">
        <tr>
            <td colspan="3" style="font-size: medium">
                <strong>Admin Login:</strong></td>
        </tr>
        <tr>
            <td style="text-align: right; width: 127px">
                User Name:</td>
            <td style="width: 204px">
                <asp:TextBox ID="TextBoxuname" runat="server"
Width="180px"></asp:TextBox>
            </td>
            <td>
                <asp:RequiredFieldValidator ID="RequiredFieldValidator1" runat="server"
                    ControlToValidate="TextBoxuname" ErrorMessage="Please enter user
name."
                    ForeColor="Red"
ValidationGroup="group1"></asp:RequiredFieldValidator>
            </td>
        </tr>
        <tr>
            <td style="text-align: right; width: 127px">
                Password:</td>
            <td style="width: 204px">
                <asp:TextBox ID="TextBoxpwd" runat="server" TextMode="Password"
Width="180px"></asp:TextBox>
            </td>
            <td>
```

```
                <asp:RequiredFieldValidator ID="RequiredFieldValidator2" runat="server"
                    ControlToValidate="TextBoxpwd" ErrorMessage="Please enter the
password."
                    ForeColor="Red"
ValidationGroup="group1"></asp:RequiredFieldValidator>
            </td>
        </tr>
        <tr>
            <td style="width: 127px">
                &nbsp;</td>
```

Adminlogout.aspx

```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="adminlogout.aspx.cs"
Inherits="adminlogout" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title></title>
</head>
<body>
    <form id="form1" runat="server">
    <div>

    </div>
    </form>
</body>
</html>
```

Adminmaster.master

```
<%@ Master Language="C#" AutoEventWireup="true" CodeFile="adminmaster.master.cs"
Inherits="adminmaster" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<!--

       Serious Face by nodethirtythree + Templated.org
       http://templated.org/ | @templatedorg
       Released under the Creative Commons Attribution 3.0 License.

       Note from the author: These templates take quite a bit of time to conceive,
       design, and finally code. So please, support our efforts by respecting our
       license: keep our footer credit links intact so people can find out about us
       and what we do. It's the right thing to do, and we'll love you for it :)

-->
<html xmlns="http://www.w3.org/1999/xhtml">
    <head>
        <meta name="keywords" content="" />
        <meta name="description" content="" />
        <meta http-equiv="content-type" content="text/html; charset=utf-8" />
              <title>ATM detector</title>
        <link href="http://fonts.googleapis.com/css?family=Bitter" rel="stylesheet"
type="text/css" />
              <link rel="stylesheet" type="text/css" href="style.css" />
    </head>
    <body>
              <div id="bg">
                     <div id="outer">
                            <div id="header">
                                   <div id="logo">
                                          <h1>
                                                 <a href="#">ATM Detector</a>
                                          </h1>
                                   </div>
                                   <!--<div id="search">
                                          <form action="" method="post">
                                                 <input class="text" name="search"
size="32" maxlength="64" /><input class="button" type="submit" value="Search" />
                                          </form>
                                   </div>-->
                                   <div id="nav">
                                          <ul>
                                                 <li class="first active">
                                                        <a href="admin_home.aspx">Home</a>
                                                 </li>
                                                 <li>
                                                        <a
href="area_details_view.aspx">Area Details</a>
                                                 </li>
                                                 <li>
                                                        <a
href="bank_details_view.aspx">Bank Details</a>
                                                 </li>
                                                 <!--<li>
                                                        <a href="#">Support</a>
                                                 </li>
                                                 <li>
                                                        <a href="#">Blog</a>
                                                 </li>-->
                                                 <li>
                                                        <a
href="atm_details_view.aspx">ATM Details</a>
                                                 </li>
                                                 <li class="last">
                                                        <a
href="adminlogout.aspx">Logout</a>
                                                 </li>
                                          </ul>
```

```
                                                        <br class="clear" />
                                                </div>
                                        </div>
                                        <div id="banner">
                                                <img src="images/pic1.jpg" width="980" height="172"
alt="" />
                                        </div>
                                        <div id="main">
                                                <!--<div id="sidebar">
                                                        <h3>
                                                                Auctor lacus
                                                        </h3>
                                                        <ul class="linkedList">
                                                                <li class="first">
                                                                        <a href="#">Vitae et suspendisse
integer dolore</a>
                                                                </li>
                                                                <li>
                                                                        <a href="#">Molestie magnis sed
amet nascetur</a>
                                                                </li>
                                                                <li>
                                                                        <a href="#">Augue sagittis
dolor</a>
                                                                </li>
                                                                <li>
                                                                        <a href="#">Orci lectus leo
urna</a>
                                                                </li>
                                                                <li>
                                                                        <a href="#">Odio volutpat
vivamus</a>
                                                                </li>
                                                                <li class="last">
                                                                        <a href="#">Porttitor fermentum
amet commodo</a>
                                                                </li>
                                                        </ul>
                                                        <h3>
                                                                Varius
                                                        </h3>
                                                        <div class="dateList">
                                                                <ul class="linkedList">
                                                                        <li class="first">
                                                                                <a href="#">Sapien et
nascetur</a>
                                                                        </li>
                                                                        <li>
                                                                                <a href="#">Nascetur sed
vivamus</a>
                                                                        </li>
                                                                        <li>
                                                                                <a href="#">Nisi amet et
sagittis</a>
                                                                        </li>
                                                                        <li>
                                                                <a href="#">Duis cras lorem
odio</a>
                                                         </li>
                                                         <li class="last">
                                                                <a href="#">Sed consequat
purus</a>
                                                         </li>
                                                  </ul>
                                           </div>
                                    </div>-->
                                    <div id="content">
                                           <div id="box1">
                         <br />
                                                  <!--<h2>
                                                         Welcome to Serious Face
                                                  </h2><!--<img class="left"
src="images/pic2.jpg" width="184" height="184" alt="" />-->
                                                  <p>
        <asp:ContentPlaceHolder id="ContentPlaceHolder1" runat="server">

        </asp:ContentPlaceHolder>
    </p>
    <br />
<br />
<br />
<br />
<br />
<br />
<br />
                                           </div>
                                           <div id="box2">
                                                  <h3>
                                                         about project:</h3>
                             <p>
                                                         An atm detector helps in locating
the nearest atm to make it easier for users/
                                customers to withdraw the amount. This also tell the
address, name of the
                                building or bank in which the particular atm is
located.</p>
                                                  <ul class="imageList">
                                                         <li class="first">
                                                         </li>
                                                  </ul>
                                           </div>
                                           <br class="clear" />
                                    </div>
                                    <br class="clear" />
                             </div>
                      </div>
                      <div id="copyright">
                             &copy; ATM Detector | All rights reserved <!--<a
href="http://templated.org/free-css-templates/seriousface/"></a> by <a
href="http://nodethirtythree.com">nodethirtythree</a> + <a
href="http://templated.org/">Templated.org</a>-->
                      </div>
               </div>
    </body>
</html>
```

Areadetails.aspx

```
<%@ Page MasterPageFile="~/adminmaster.master" Language="C#" AutoEventWireup="true"
CodeFile="area_details.aspx.cs" Inherits="area_details" %>

<asp:Content ID="Content1" ContentPlaceHolderID="ContentPlaceHolder1" Runat="Server">
    <form id="form1" runat="server">
    <table class="style1">
        <tr>
            <td class="style5" colspan="2"
                style="text-align: left; text-decoration: underline;">
                AREA DETAILS</td>
        </tr>
        <tr>
            <td class="style6">
                AREA_NAME:</td>
            <td>
                <asp:TextBox ID="TextBoxareaname" runat="server" CssClass="style7"
                    Width="187px"></asp:TextBox>
                <asp:RequiredFieldValidator ID="RequiredFieldValidator1" runat="server"
                    ControlToValidate="TextBoxlandmark" CssClass="style8"
                    ErrorMessage="This field is empty"
ForeColor="Red"></asp:RequiredFieldValidator>
&nbsp;<asp:RegularExpressionValidator ID="RegularExpressionValidator1" runat="server"
                    ControlToValidate="TextBoxareaname" CssClass="style8"
                    ErrorMessage="Invalid Name" ForeColor="Red" ValidationExpression="[A-
Za-z ]*$"></asp:RegularExpressionValidator>
            </td>
        </tr>
        <tr>
            <td class="style6">
                City:</td>
            <td>
                <asp:TextBox ID="TextBoxlandmark" runat="server"
CssClass="style7"></asp:TextBox>
                <asp:RequiredFieldValidator ID="RequiredFieldValidator2" runat="server"
                    ControlToValidate="TextBoxlandmark" CssClass="style8"
                    ErrorMessage="This field is empty"
ForeColor="Red"></asp:RequiredFieldValidator>
                <asp:RegularExpressionValidator ID="RegularExpressionValidator2"
runat="server"
                    ControlToValidate="TextBoxlandmark" CssClass="style8"
                    ErrorMessage="Invalid Name" ForeColor="Red" ValidationExpression="[A-
Za-z ]*$"></asp:RegularExpressionValidator>
            </td>
        </tr>
        <tr>
            <td class="style2" colspan="2">
```

```
&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp
;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbs
p;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nb
sp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;
                <asp:Button ID="submit" runat="server" onclick="Button1_Click" 
Text="SUBMIT" 
                    CssClass="style4" />
            </td>
        </tr>
    </table>
    <div>
    
    </div>
    </form>
</asp:content>
```

Area_details_view.aspx

```
<%@ Page MasterPageFile="~/adminmaster.master" Language="C#" AutoEventWireup="true" 
CodeFile="area_details_view.aspx.cs" Inherits="area_details_view" %>

<asp:Content ID="Content1" ContentPlaceHolderID="ContentPlaceHolder1" Runat="Server">
    <form id="form1" runat="server">
    <div>
    
        <asp:GridView ID="GridView1" runat="server" AutoGenerateColumns="False" 
            CellPadding="4" DataKeyNames="area_id" DataSourceID="SqlDataSource1" 
            ForeColor="#333333" GridLines="None">
            <AlternatingRowStyle BackColor="White" />
            <Columns>
                <asp:CommandField ShowDeleteButton="True" ShowEditButton="True" />
                <asp:BoundField DataField="area_id" HeaderText="area_id" 
InsertVisible="False" 
                    ReadOnly="True" SortExpression="area_id" />
                <asp:BoundField DataField="area_name" HeaderText="area_name" 
                    SortExpression="area_name" />
                <asp:BoundField DataField="land_mark" HeaderText="land_mark" 
                    SortExpression="land_mark" />
            </Columns>
            <EditRowStyle BackColor="#7C6F57" />
            <FooterStyle BackColor="#1C5E55" ForeColor="White" Font-Bold="True" />
            <HeaderStyle BackColor="#1C5E55" Font-Bold="True" ForeColor="White" />
            <PagerStyle ForeColor="White" HorizontalAlign="Center" BackColor="#666666" />
            <RowStyle BackColor="#E3EAEB" />
            <SelectedRowStyle BackColor="#C5BBAF" Font-Bold="True" ForeColor="#333333" />
            <SortedAscendingCellStyle BackColor="#F8FAFA" />
            <SortedAscendingHeaderStyle BackColor="#246B61" />
            <SortedDescendingCellStyle BackColor="#D4DFE1" />
            <SortedDescendingHeaderStyle BackColor="#15524A" />
        </asp:GridView>
        <asp:SqlDataSource ID="SqlDataSource1" runat="server" 
            ConnectionString="<%$ ConnectionStrings:ConnectionString11 %>" 
            DeleteCommand="DELETE FROM [area_details] WHERE [area_id] = @area_id" 
            InsertCommand="INSERT INTO [area_details] ([area_name], [land_mark]) VALUES 
(@area_name, @land_mark)" 
            SelectCommand="SELECT * FROM [area_details]" 
            UpdateCommand="UPDATE [area_details] SET [area_name] = @area_name, 
[land_mark] = @land_mark WHERE [area_id] = @area_id">
            <DeleteParameters>
                <asp:Parameter Name="area_id" Type="Int32" />
            </DeleteParameters>
            <InsertParameters>
                <asp:Parameter Name="area_name" Type="String" />
                <asp:Parameter Name="land_mark" Type="String" />
            </InsertParameters>
            <UpdateParameters>
                <asp:Parameter Name="area_name" Type="String" />
                <asp:Parameter Name="land_mark" Type="String" />
                <asp:Parameter Name="area_id" Type="Int32" />
            </UpdateParameters>
        </asp:SqlDataSource>
    <br />
        <asp:HyperLink ID="HyperLink1" runat="server" 
NavigateUrl="~/area_details.aspx">+Add new</asp:HyperLink>
    </div>
    </form>
</asp:Content>
```

Atm_details.aspx

```
<%@ Page MasterPageFile="~/adminmaster.master" Language="C#" AutoEventWireup="true" 
CodeFile="atm_details.aspx.cs" Inherits="atm_details" %>

<asp:Content ID="Content1" ContentPlaceHolderID="ContentPlaceHolder1" Runat="Server">
    <form id="form1" runat="server">
    <table class="style1">
        <tr>
            <td class="style8" colspan="2" style="text-decoration: underline;">
                ATM DETAILS</td>
        </tr>
        <tr>
            <td class="style4">
                ATM_NAME:</td>
            <td>
                <asp:TextBox ID="atm_name" runat="server" 
CssClass="style6"></asp:TextBox>
                <asp:RequiredFieldValidator ID="RequiredFieldValidator1" runat="server" 
                    ControlToValidate="atm_name" CssClass="style7" 
                    ErrorMessage="This field is empty" 
ForeColor="Red"></asp:RequiredFieldValidator>
                <asp:RegularExpressionValidator ID="RegularExpressionValidator1" 
runat="server" 
                    ControlToValidate="atm_name" CssClass="style7" ErrorMessage="Invalid 
Name"
```

```
                    ForeColor="Red" ValidationExpression="[A-Za-z
]*$"></asp:RegularExpressionValidator>
            </td>
        </tr>
        <tr>
            <td class="style4">
                ATM_NO:</td>
            <td>
                <asp:TextBox ID="atm_no" runat="server" CssClass="style6"></asp:TextBox>
                <asp:RequiredFieldValidator ID="RequiredFieldValidator6" runat="server"
                    ControlToValidate="atm_no" ErrorMessage="This field is empty"
ForeColor="Red"></asp:RequiredFieldValidator>
            </td>
        </tr>
        <tr>
            <td class="style4">
                AREA_TYPE:</td>
            <td>
                <asp:TextBox ID="area_type" runat="server"
CssClass="style6"></asp:TextBox>
                <asp:RequiredFieldValidator ID="RequiredFieldValidator2" runat="server"
                    ControlToValidate="area_type" CssClass="style7"
                    ErrorMessage="This field is empty"
ForeColor="Red"></asp:RequiredFieldValidator>
            </td>
        </tr>
        <tr>
            <td class="style4">
                BANK_ID:</td>
            <td>
                <asp:ListBox ID="ListBox1" runat="server" DataSourceID="SqlDataSource1"
                    DataTextField="bank_name" DataValueField="bank_id" Rows="1"
Width="180px">
                </asp:ListBox>
                <asp:SqlDataSource ID="SqlDataSource1" runat="server"
                    ConnectionString="<%$ ConnectionStrings:ConnectionString11 %>"
                    SelectCommand="SELECT * FROM [bank_details]"></asp:SqlDataSource>
                <asp:RequiredFieldValidator ID="RequiredFieldValidator7" runat="server"
                    ControlToValidate="ListBox1" ErrorMessage="This field is empty"
                    ForeColor="Red"></asp:RequiredFieldValidator>
            </td>
        </tr>
        <tr>
            <td class="style4">
                AREA_ID:</td>
            <td>
                <asp:DropDownList ID="DropDownList1" runat="server"
                    DataSourceID="SqlDataSource2" DataTextField="area_name"
                    DataValueField="area_id" Width="180px">
                </asp:DropDownList>
                <asp:SqlDataSource ID="SqlDataSource2" runat="server"
                    ConnectionString="<%$ ConnectionStrings:ConnectionString11 %>"
                    SelectCommand="SELECT DISTINCT * FROM
[area_details]"></asp:SqlDataSource>
                <asp:RequiredFieldValidator ID="RequiredFieldValidator8" runat="server"
                    ControlToValidate="DropDownList1" ErrorMessage="This field is empty"
                    ForeColor="Red"></asp:RequiredFieldValidator>
            </td>
        </tr>
        <tr>
            <td class="style4">
                STATUS:</td>
            <td>
                <asp:RadioButtonList ID="RadioButtonList1" runat="server">
                    <asp:ListItem>cash</asp:ListItem>
                    <asp:ListItem>no cash</asp:ListItem>
                </asp:RadioButtonList>
                <asp:RequiredFieldValidator ID="RequiredFieldValidator3" runat="server"
                    ControlToValidate="RadioButtonList1" CssClass="style7"
ErrorMessage="This field is empty"
                    ForeColor="Red"></asp:RequiredFieldValidator>
            </td>
        </tr>
        <tr>
            <td class="style4">
                CC_CAMERA:</td>
            <td>
                <asp:RadioButtonList ID="RadioButtonList2" runat="server">
                    <asp:ListItem>Installed</asp:ListItem>
                    <asp:ListItem>not installed</asp:ListItem>
                </asp:RadioButtonList>
                <asp:RequiredFieldValidator ID="RequiredFieldValidator4" runat="server"
                    ControlToValidate="RadioButtonList2" CssClass="style7"
                    ErrorMessage="This field is empty"
ForeColor="Red"></asp:RequiredFieldValidator>
            </td>
        </tr>
        <tr>
            <td class="style4">
                SECURITY_GUARD:</td>
            <td>
                <asp:RadioButtonList ID="RadioButtonList3" runat="server">
                    <asp:ListItem>present</asp:ListItem>
                    <asp:ListItem>not present</asp:ListItem>
                </asp:RadioButtonList>
                <asp:RequiredFieldValidator ID="RequiredFieldValidator5" runat="server"
                    ControlToValidate="RadioButtonList3" CssClass="style7"
                    ErrorMessage="This field is empty"
ForeColor="Red"></asp:RequiredFieldValidator>
            </td>
        </tr>
        <tr>
            <td class="style2">
                <asp:Button ID="submit" runat="server" CssClass="style3"
onclick="submit_Click"
                    Text="SUBMIT" />
            </td>
            <td class="style3">
                &nbsp;</td>
        </tr>
    </table>
    <div>

    </div>
    </form>
</asp:content>
```

Atm_details_view.aspx

```
<%@ Page MasterPageFile="~/adminmaster.master" Language="C#" AutoEventWireup="true"
CodeFile="atm_details_view.aspx.cs" Inherits="atm_details_view" %>

<asp:Content ID="Content1" ContentPlaceHolderID="ContentPlaceHolder1" Runat="Server">
    <form id="form1" runat="server">
    <div>

        <asp:GridView ID="GridView1" runat="server" AutoGenerateColumns="False"
            DataKeyNames="atm_id" DataSourceID="SqlDataSource1" CellPadding="4"
            ForeColor="#333333" GridLines="None">
            <AlternatingRowStyle BackColor="White" />
            <Columns>
                <asp:CommandField ShowDeleteButton="True" ShowEditButton="True" />
                <asp:BoundField DataField="atm_name" HeaderText="atm name"
                    SortExpression="atm_name" />
                <asp:BoundField DataField="atm_no" HeaderText="atm no"
                    SortExpression="atm_no" />
                <asp:BoundField DataField="area_type" HeaderText="area type"
                    SortExpression="area_type" />
                <asp:BoundField DataField="bank_id" HeaderText="bank id"
                    SortExpression="bank_id" />
                <asp:BoundField DataField="area_id" HeaderText="area id"
                    SortExpression="area_id" />
                <asp:BoundField DataField="status" HeaderText="status"
                    SortExpression="status" />
                <asp:BoundField DataField="cc_camera" HeaderText="cc camera"
                    SortExpression="cc_camera" />
                <asp:BoundField DataField="security" HeaderText="security"
                    SortExpression="security" />
            </Columns>
            <EditRowStyle BackColor="#7C6F57" />
            <FooterStyle BackColor="#1C5E55" ForeColor="White" Font-Bold="True" />
            <HeaderStyle BackColor="#1C5E55" Font-Bold="True" ForeColor="White" />
            <PagerStyle ForeColor="White" HorizontalAlign="Center" BackColor="#666666" />
            <RowStyle BackColor="#E3EAEB" />
            <SelectedRowStyle BackColor="#C5BBAF" Font-Bold="True" ForeColor="#333333" />
            <SortedAscendingCellStyle BackColor="#F8FAFA" />
            <SortedAscendingHeaderStyle BackColor="#246B61" />
            <SortedDescendingCellStyle BackColor="#D4DFE1" />
            <SortedDescendingHeaderStyle BackColor="#15524A" />
        </asp:GridView>
        <asp:SqlDataSource ID="SqlDataSource1" runat="server"
            ConnectionString="<%$ ConnectionStrings:ConnectionString11 %>"
            DeleteCommand="DELETE FROM [atm_details] WHERE [atm_id] = @atm_id"
            InsertCommand="INSERT INTO [atm_details] ([atm_name], [atm_no], [area_type],
[bank_id], [area_id], [status], [cc_camera], [security]) VALUES (@atm_name, @atm_no,
@area_type, @bank_id, @area_id, @status, @cc_camera, @security)"
            SelectCommand="SELECT * FROM [atm_details]"
            UpdateCommand="UPDATE [atm_details] SET [atm_name] = @atm_name, [atm_no] =
@atm_no, [area_type] = @area_type, [bank_id] = @bank_id, [area_id] = @area_id, [status] =
@status, [cc_camera] = @cc_camera, [security] = @security WHERE [atm_id] = @atm_id">
            <DeleteParameters>
                <asp:Parameter Name="atm_id" Type="Int32" />
            </DeleteParameters>
            <InsertParameters>
                <asp:Parameter Name="atm_name" Type="String" />
                <asp:Parameter Name="atm_no" Type="Int32" />
                <asp:Parameter Name="area_type" Type="String" />
                <asp:Parameter Name="bank_id" Type="Int32" />
                <asp:Parameter Name="area_id" Type="Int32" />
                <asp:Parameter Name="status" Type="String" />
                <asp:Parameter Name="cc_camera" Type="String" />
                <asp:Parameter Name="security" Type="String" />
            </InsertParameters>
            <UpdateParameters>
                <asp:Parameter Name="atm_name" Type="String" />
                <asp:Parameter Name="atm_no" Type="Int32" />
                <asp:Parameter Name="area_type" Type="String" />
                <asp:Parameter Name="bank_id" Type="Int32" />
                <asp:Parameter Name="area_id" Type="Int32" />
                <asp:Parameter Name="status" Type="String" />
                <asp:Parameter Name="cc_camera" Type="String" />
                <asp:Parameter Name="security" Type="String" />
                <asp:Parameter Name="atm_id" Type="Int32" />
            </UpdateParameters>
        </asp:SqlDataSource>
    <br />
        <asp:HyperLink ID="HyperLink1" runat="server"
NavigateUrl="~/atm_details.aspx">+Add new</asp:HyperLink>
    </div>
    </form>
</asp:content>
```

Bank_details.aspx

```
<%@ Page MasterPageFile="~/adminmaster.master" Language="C#" AutoEventWireup="true"
CodeFile="bank_details.aspx.cs" Inherits="bank_details" %>

<asp:Content ID="Content1" ContentPlaceHolderID="ContentPlaceHolder1" Runat="Server">
    <form id="form1" runat="server">
    <table class="style1">
        <tr>
            <td class="style3" colspan="2"
                style="font-weight: 700; font-size: large; text-decoration: underline">
                BANK DETAILS</td>
        </tr>
        <tr>
            <td class="style5">
                BANK NAME:</td>
            <td class="style4">
```

```
                <asp:TextBox ID="bank_name" runat="server" 
CssClass="style6"></asp:TextBox>
                <span class="style7">
                <asp:RequiredFieldValidator ID="RequiredFieldValidator1" runat="server" 
                    ControlToValidate="bank_name" ErrorMessage="This field is empty" 
                    ForeColor="Red"></asp:RequiredFieldValidator>
                <asp:RegularExpressionValidator ID="RegularExpressionValidator1" 
runat="server" 
                    ControlToValidate="bank_name" ErrorMessage="Invalid Name" 
ForeColor="Red" 
                    ValidationExpression="[A-Za-z ]*$"></asp:RegularExpressionValidator>
                </span>
            </td>
        </tr>
        <tr>
            <td class="style5">
                BANK MAIN BRANCH:</td>
            <td class="style4">
                <asp:TextBox ID="bank_main_branch" runat="server" 
CssClass="style6"></asp:TextBox>
                <span class="style7">
                <asp:RequiredFieldValidator ID="RequiredFieldValidator2" runat="server" 
                    ControlToValidate="bank_main_branch" ErrorMessage="This field is 
empty" 
                    ForeColor="Red"></asp:RequiredFieldValidator>
                <asp:RegularExpressionValidator ID="RegularExpressionValidator2" 
runat="server" 
                    ControlToValidate="bank_main_branch" ErrorMessage="Invalid name" 
                    ForeColor="Red" ValidationExpression="[A-Za-z 
]*$"></asp:RegularExpressionValidator>
                </span>
            </td>
        </tr>
        <tr>
            <td class="style5">
                CONTACT NO:</td>
            <td class="style4">
                <asp:TextBox ID="contact_no" runat="server" 
CssClass="style6"></asp:TextBox>
                <span class="style7">
                <asp:RequiredFieldValidator ID="RequiredFieldValidator3" runat="server" 
                    ControlToValidate="contact_no" ErrorMessage="This field is empty" 
                    ForeColor="Red"></asp:RequiredFieldValidator>
                <asp:RegularExpressionValidator ID="RegularExpressionValidator4" 
runat="server" 
                    ControlToValidate="contact_no" ErrorMessage="Invalid No" 
ForeColor="Red" 
                    ValidationExpression="^([7-9]{1})([0-
9]{9})$"></asp:RegularExpressionValidator>
                </span>
            </td>
        </tr>
        <tr>
            <td class="style5">
                BRANCH MANAGER:</td>
            <td class="style4">
```

```
                <asp:TextBox ID="branch_manager" runat="server" 
CssClass="style6"></asp:TextBox>
                <span class="style7">
                <asp:RequiredFieldValidator ID="RequiredFieldValidator4" runat="server" 
                    ControlToValidate="branch_manager" ErrorMessage="This field is empty" 
                    ForeColor="Red"></asp:RequiredFieldValidator>
                <asp:RegularExpressionValidator ID="RegularExpressionValidator3" 
runat="server" 
                    ControlToValidate="branch_manager" ErrorMessage="Invalid Name" 
ForeColor="Red" 
                    ValidationExpression="[A-Za-z ]*$"></asp:RegularExpressionValidator>
                </span>
            </td>
        </tr>
        <tr>
            <td class="style2">
                <asp:Button ID="submit" runat="server" CssClass="style4" 
onclick="submit_Click" 
                    style="text-align: right" Text="SUBMIT" />
            </td>
            <td class="style4">
                &nbsp;</td>
        </tr>
    </table>
    <div>
    
    </div>
    </form>
</asp:content>
```

Bank_details_view.aspx

```
<%@ Page MasterPageFile="~/adminmaster.master" Language="C#" AutoEventWireup="true" 
CodeFile="bank_details_view.aspx.cs" Inherits="bank_details_view" %>

<asp:Content ID="Content1" ContentPlaceHolderID="ContentPlaceHolder1" Runat="Server">
    <form id="form1" runat="server">
    <div>
    
        <asp:GridView ID="GridView1" runat="server" AutoGenerateColumns="False" 
            CellPadding="4" DataKeyNames="bank_id" 
            DataSourceID="SqlDataSource1" ForeColor="#333333" GridLines="None">
            <AlternatingRowStyle BackColor="White" />
            <Columns>
                <asp:CommandField ShowDeleteButton="True" ShowEditButton="True" />
                <asp:BoundField DataField="bank_name" HeaderText=" bank name" 
                    SortExpression="bank_name" />
                <asp:BoundField DataField="bank_main_branchs" HeaderText=" _bank main 
branchs" 
                    SortExpression="bank_main_branchs" />
                <asp:BoundField DataField="contact_no" HeaderText=" _contact no"
                    SortExpression="contact_no" />
                <asp:BoundField DataField="branch_manager" HeaderText=" _  branch
manager"
                    SortExpression="branch_manager" />
            </Columns>
            <EditRowStyle BackColor="#7C6F57" />
            <FooterStyle BackColor="#1C5E55" ForeColor="White" Font-Bold="True" />
            <HeaderStyle BackColor="#1C5E55" Font-Bold="True" ForeColor="White" />
            <PagerStyle ForeColor="White" HorizontalAlign="Center" BackColor="#666666" />
            <RowStyle BackColor="#E3EAEB" />
            <SelectedRowStyle BackColor="#C5BBAF" Font-Bold="True" ForeColor="#333333" />
            <SortedAscendingCellStyle BackColor="#F8FAFA" />
            <SortedAscendingHeaderStyle BackColor="#246B61" />
            <SortedDescendingCellStyle BackColor="#D4DFE1" />
            <SortedDescendingHeaderStyle BackColor="#15524A" />
        </asp:GridView>
        <asp:SqlDataSource ID="SqlDataSource1" runat="server"
            ConnectionString="<%$ ConnectionStrings:ConnectionString11 %>"
            DeleteCommand="DELETE FROM [bank_details] WHERE [bank_id] = @bank_id"
            InsertCommand="INSERT INTO [bank_details] ([bank_name], [bank_main_branchs],
[contact_no], [branch_manager]) VALUES (@bank_name, @bank_main_branchs, @contact_no,
@branch_manager)"
            SelectCommand="SELECT * FROM [bank_details]"
            UpdateCommand="UPDATE [bank_details] SET [bank_name] = @bank_name,
[bank_main_branchs] = @bank_main_branchs, [contact_no] = @contact_no, [branch_manager] =
@branch_manager WHERE [bank_id] = @bank_id">
            <DeleteParameters>
                <asp:Parameter Name="bank_id" Type="Int32" />
            </DeleteParameters>
            <InsertParameters>
                <asp:Parameter Name="bank_name" Type="String" />
                <asp:Parameter Name="bank_main_branchs" Type="String" />
                <asp:Parameter Name="contact_no" Type="String" />
                <asp:Parameter Name="branch_manager" Type="String" />
            </InsertParameters>
            <UpdateParameters>
                <asp:Parameter Name="bank_name" Type="String" />
                <asp:Parameter Name="bank_main_branchs" Type="String" />
                <asp:Parameter Name="contact_no" Type="String" />
                <asp:Parameter Name="branch_manager" Type="String" />
                <asp:Parameter Name="bank_id" Type="Int32" />
            </UpdateParameters>
        </asp:SqlDataSource>
    <br />
        <asp:HyperLink ID="HyperLink1" runat="server"
NavigateUrl="~/bank_details.aspx">+Add new</asp:HyperLink>
    </div>
    </form>
</asp:content>
```

Connect.aspx

```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="connect.aspx.cs"
Inherits="connect" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title></title>
</head>
<body>
    <form id="form1" runat="server">
    <div>

        <asp:SqlDataSource ID="SqlDataSource1" runat="server"
            ConnectionString="<%$ ConnectionStrings:ConnectionString11 %>"
            SelectCommand="SELECT * FROM [area_details]"></asp:SqlDataSource>

    </div>
    </form>
</body>
</html>
```

Contactpage

```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="connect.aspx.cs"
Inherits="connect" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title></title>
</head>
<body>
    <form id="form1" runat="server">
    <div>

        <asp:SqlDataSource ID="SqlDataSource1" runat="server"
            ConnectionString="<%$ ConnectionStrings:ConnectionString11 %>"
            SelectCommand="SELECT * FROM [area_details]"></asp:SqlDataSource>

    </div>
    </form>
</body>
</html>
```

Index.html

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<!--

       Serious Face by nodethirtythree + Templated.org
       http://templated.org/ | @templatedorg
       Released under the Creative Commons Attribution 3.0 License.

       Note from the author: These templates take quite a bit of time to conceive,
       design, and finally code. So please, support our efforts by respecting our
       license: keep our footer credit links intact so people can find out about us
       and what we do. It's the right thing to do, and we'll love you for it :)

-->
<html xmlns="http://www.w3.org/1999/xhtml">
    <head>
        <meta name="keywords" content="" />
        <meta name="description" content="" />
        <meta http-equiv="content-type" content="text/html; charset=utf-8" />
              <title>Serious Face | Designed by nodethirtythree + Templated.org</title>
        <link href="http://fonts.googleapis.com/css?family=Bitter" rel="stylesheet"
type="text/css" />
              <link rel="stylesheet" type="text/css" href="style.css" />
    </head>
    <body>
              <div id="bg">
                     <div id="outer">
                            <div id="header">
                                   <div id="logo">
                                          <h1>
                                                 <a href="#">Serious Face</a>
                                          </h1>
                                   </div>
                                   <div id="search">
                                          <form action="" method="post">
                                                 <input class="text" name="search"
size="32" maxlength="64" /><input class="button" type="submit" value="Search" />
                                          </form>
                                   </div>
                                   <div id="nav">
                                          <ul>
                                                 <li class="first active">
                                                        <a href="#">Home</a>
```

8.Development Tools

Certainly! C# (pronounced "C sharp") is a modern, object-oriented programming language developed by Microsoft. It was first released in 2000 and has since become one of the most popular languages for building Windows applications, web applications, and more. Here are some key features and concepts related to C#:

Object-Oriented: C# is an object-oriented programming (OOP) language, which means it allows you to model real-world entities and their interactions using objects and classes.

Type-Safe: C# is a statically-typed language, which means that variable types are determined at compile-time. This helps catch type-related errors early in the development process.

Managed Code: C# programs are compiled into Intermediate Language (IL) code, which is executed by the Common Language Runtime (CLR). This makes C# a managed language, as the CLR handles memory management, garbage collection, and other low-level tasks.

Platform Independence: While C# was initially designed for Windows development, it has become more versatile over the years. With technologies like .NET Core (now .NET 5 and beyond), you can build cross-platform applications for Windows, Linux, and macOS.

Syntax: C# syntax is similar to other C-style languages like C, C++, and Java. It uses curly braces {} to define blocks of code and has a rich set of keywords for control structures, data types, and more.

.NET Framework and .NET Core: C# is commonly used with the .NET Framework (Windows-specific) and .NET Core (cross-platform) to build various types of applications, including Windows Forms, ASP.NET web applications, and console applications.

Asynchronous Programming: C# has robust support for asynchronous programming using the async and await keywords. This is particularly useful for building responsive and scalable applications.

LINQ (Language Integrated Query): C# includes LINQ, which allows developers to query collections, databases, XML, and other data sources using a uniform syntax. It greatly simplifies data manipulation and retrieval.

Strong Standard Library: C# comes with a rich standard library (in the form of .NET libraries) that provides a wide range of pre-built classes and functions for common tasks, making development faster and more efficient.

IDE Support: Visual Studio is the primary Integrated Development Environment (IDE) for C# development. It offers features like code completion, debugging, and project management, making it a powerful tool for C# programmers.

C# is commonly used for developing a variety of applications, including:

Desktop Applications: Windows Forms and WPF applications for creating desktop software.

Web Applications: ASP.NET is used for building web applications, and ASP.NET Core extends this to cross-platform web development.

Mobile Applications: Xamarin allows C# developers to create cross-platform mobile apps for iOS and Android.

Game Development: Unity3D, a popular game development engine, uses C# for scripting game logic.

Server Applications: C# is also used for building server-side applications, including web services and APIs.

C# continues to evolve, with Microsoft regularly releasing new versions and features to enhance its capabilities and maintain its relevance in the software development landscape.

Object-Oriented:

Certainly, let's delve deeper into the concept of object-oriented programming (OOP).

Object-Oriented Programming (OOP) is a programming paradigm that is centered around the concept of "objects." Objects are instances of classes, which are user-defined blueprints for creating objects. In OOP, you model real-world entities and their interactions using these objects. C# is an object-oriented programming language, and OOP principles are fundamental to its design.

Here are some key concepts and principles of OOP:

Classes and Objects: In OOP, you define classes to represent templates or blueprints for objects. Objects are instances of these classes. For example, you might have a Car class that describes the attributes and behaviors of cars, and then you can create individual car objects from this class.

Encapsulation: Encapsulation is the concept of bundling data (attributes or fields) and methods (functions or behaviors) that operate on that data into a single unit called a class. This unit provides control over the access to the data, allowing you to define public and private members. In C#, you can use access modifiers like public, private, protected, and internal to control access to class members.

Inheritance: Inheritance is a mechanism that allows a new class (called a derived or subclass) to inherit properties and behaviors from an existing class (called a base or superclass). This promotes code reuse and hierarchy. In C#, you can use the : baseClass syntax to create derived classes.

Polymorphism: Polymorphism allows objects of different classes to be treated as objects of a common base class. This is achieved through method overriding and interfaces. In C#, you can use the override keyword to implement polymorphism by providing specialized implementations of methods defined in base classes.

Abstraction: Abstraction is the process of simplifying complex reality by modeling classes based on essential properties and behaviors while hiding unnecessary details. Abstract classes and interfaces in C# help in achieving abstraction.

Objects Communicate: In OOP, objects interact with each other by sending messages. This communication often involves calling methods on objects or accessing their properties.

Association and Composition: Objects in OOP can have associations with each other, representing relationships between objects. Composition is a strong form of association, where one class contains another class as a part.

Polymorphism and Dynamic Binding: Polymorphism allows you to write code that can work with objects of various classes in a generalized way. Dynamic binding ensures that the correct method is called at runtime, based on the actual type of the object.

OOP promotes code organization, reusability, and maintainability. It is particularly useful when dealing with complex systems where modeling real-world entities and their relationships is essential.

In C#, you'll use classes and objects extensively to build software systems, and understanding these OOP principles will help you design and develop effective and modular code.

Type-Safe:

"Type-safe" is a term used in programming languages, including C#, to describe a language's ability to prevent type-related errors at compile-time or runtime. In a type-safe language like C#,:

Compile-Time Type Checking: The C# compiler checks the types of variables, expressions, and method arguments at compile-time. It ensures that you don't perform operations or assignments that violate type rules. For example, you can't assign a string to an integer variable without explicit type conversion.

Strong Typing: C# is strongly typed, which means that variables have specific data types, and these types are enforced strictly. Once a variable is declared with a particular type, you cannot change its type without explicit casting.

Type Inference: In modern versions of C#, type inference is used to automatically determine the data type of a variable based on its initialization value. This reduces the need for explicit type declarations while maintaining type safety.

Preventing Type Mismatch Errors: Type-safe languages prevent common type-related errors, such as adding a string to an integer or calling a method on an object that doesn't support it. This ensures that the program behaves predictably and avoids unexpected runtime errors.

Safe Memory Management: In managed languages like C# (where memory is managed by the runtime), type safety helps prevent memory-related errors such as buffer overflows or accessing memory locations with incorrect types.

Here's an example in C# to illustrate type safety:

Csharp Code

```
int num1 = 10;
string text = "Hello, World!";
// The following line would result in a compilation error because you can't add an integer and a string.
// int result = num1 + text;
```

In this example, trying to add an integer (num1) and a string (text) would lead to a compile-time error because the compiler detects the type mismatch. This is an example of type safety in action.

Type safety is a crucial feature in programming languages because it helps catch errors early in the development process, making code more reliable and easier to maintain. It also contributes to the overall security and stability of software systems by reducing the likelihood of unexpected runtime errors and vulnerabilities related to type manipulation.

Managed Code:

"Managed code" is a concept commonly associated with languages and platforms like C# and .NET. It refers to code that is executed and managed by a runtime environment, such as the Common Language Runtime (CLR) in the context of C# and the .NET Framework or .NET Core.

Here are key aspects of managed code:

Compilation to Intermediate Language (IL): When you write C# code and compile it, the C# compiler translates your source code into Intermediate Language (IL) code rather than directly generating machine code. IL is a low-level, platform-agnostic representation of your code.

Common Language Runtime (CLR): In the .NET ecosystem, the CLR is responsible for executing managed code. It loads the IL code, manages memory, handles exceptions, and provides various runtime services. It also performs Just-In-Time (JIT) compilation, translating IL code into machine code as needed.

Memory Management: One of the advantages of managed code is automatic memory management, specifically garbage collection. The CLR tracks and frees memory that is no longer in use, helping to prevent memory leaks and related bugs.

Security: Managed code is often more secure because the CLR enforces various security checks, such as type safety and code access security. This helps protect against buffer overflows and other common security vulnerabilities.

Platform Independence: Managed code can be run on any platform that has a compatible runtime environment. For example, .NET Core (now .NET 5 and beyond) allows you to build cross-platform applications, so code written in C# can run on Windows, Linux, and macOS.

Interoperability: Managed code can interact with unmanaged code (code not managed by the runtime) through mechanisms like Platform Invocation Services (P/Invoke) or COM interop. This allows you to use existing libraries and components written in languages like C++ within your C# applications.

Here's a simplified example of managed code in C#:

Csharp Code

```
using System;

class Program
{
    static void Main()
    {
        // This C# code is managed code.
        // It gets compiled to IL and executed by the CLR.

        Console.WriteLine("Hello, world!");
    }
}
```

In the context of C# and the .NET framework, here's how platform independence is achieved:

Common Intermediate Language (CIL): When you write C# code and compile it, the C# compiler generates Common Intermediate Language (CIL) code, also known as Intermediate Language (IL) code. This In this example, the C# code is compiled to IL code. When you run the program, the CLR manages the execution, including loading the IL code, executing it, and managing resources like memory and I/O operations.

The benefits of managed code include improved productivity, better security, and platform flexibility. However, there may be some performance overhead due to the runtime environment, especially when compared to languages that compile directly to native machine code.

Platform Independence:

"Platform independence," in the context of software development, refers to the ability of a program or application to run on different computer platforms or operating systems without requiring significant modifications or recompilation. Platform independence is a desirable trait for software because it allows developers to write code once and deploy it on multiple platforms, saving time and effort.

CIL code is a platform-agnostic representation of your program.

Common Language Runtime (CLR): The CLR is responsible for executing CIL code. It is a part of the .NET framework and is designed to be platform-independent. The CLR is available for multiple operating systems, including Windows, Linux, and macOS. When you run a C# program, the CLR loads the CIL code and performs Just-In-Time (JIT) compilation to generate native machine code for the specific platform it's running on.

Cross-Platform Frameworks: The availability of cross-platform implementations of the .NET framework, such as .NET Core (now .NET 5 and beyond), further enhances platform independence. With .NET Core, you can develop C# applications that can run on Windows, Linux, and macOS. The framework provides platform-specific libraries and APIs to ensure compatibility.

Platform-Specific Considerations: While C# and .NET Core provide a high degree of platform independence, there may still be some platform-specific considerations. For example, user interface (UI) elements in a C# application may need to adapt to the conventions and capabilities of the target platform. In such cases, you may use platform-specific libraries or frameworks for the UI while keeping the core application logic portable.

Here's a simplified example of platform independence in C# using .NET Core:

Csharp code

```
using System;

class Program
{
    static void Main()
    {
        // This C# code can run on Windows, Linux, or macOS
        Console.WriteLine("Hello, platform-independent world!");
    }
}
```

In this example, the C# code is written once and can be executed on multiple platforms without modification, thanks to the platform independence provided by the .NET Core runtime.

Platform independence is a significant advantage for developers because it allows them to target a broader range of devices and operating systems while maintaining code consistency and reusability. This makes C# a versatile choice for cross-platform application development.

Syntax:

Syntax in programming refers to the set of rules and conventions that dictate how programs written in a specific programming language should be structured. It defines how statements, expressions, and other language elements should be written to create valid and meaningful code. In C#, like in most programming languages, there are specific syntax rules that developers must follow. Here are some fundamental syntax elements in C#:

Statements: Statements are individual lines of code that perform specific actions. Statements in C# typically end with a semicolon (;). For example:

Csharp Code

```
int x = 5; // Declaration and assignment statement
Console.WriteLine("Hello, World!"); // Method call statement
```

Variables and Data Types: Variables are used to store data, and C# is a statically typed language, meaning you must declare the type of a variable before using it. For example:

Csharp Code

```
int age = 30; // Declare an integer variable 'age'
string name = "John"; // Declare a string variable 'name'
```

Comments: Comments are used to annotate code and provide explanations. In C#, you can use // for single-line comments and /* */ for multi-line comments. Comments are ignored by the compiler.

Csharp Code

```
// This is a single-line comment
/*
This is a
multi-line comment
*/
```

Methods and Functions: In C#, methods or functions are blocks of code that perform specific tasks. They have a signature that includes a name, optional parameters, and a return type. For example:

Csharp Code

```
// Method definition
int Add(int a, int b)
{
    return a + b;
}

// Method call
int result = Add(3, 4);
```

Control Structures: Control structures are used to manage the flow of a program. Common control structures in C# include if statements, for loops, while loops, and switch statements. For example:

Csharp Code

```
if (condition)
{
    // Code to execute if the condition is true
}

for (int i = 0; i < 5; i++)
{
   // Code to execute in each iteration of the loop
}
```

Class and Object Definitions: C# is an object-oriented language, so it involves defining classes and creating objects (instances of classes). Classes are defined using the class keyword, and objects are created using the new keyword. For example:

Csharp Code

```
// Class definition
class Person
{
   public string Name { get; set; }
   public int Age { get; set; }
}

// Object creation
Person person = new Person();
person.Name = "Alice";
person.Age = 25;
```

Access Modifiers: Access modifiers control the visibility and accessibility of classes, methods, and variables. Common access modifiers in C# include public, private, protected, and internal.

Csharp Code

```
public class MyClass
{
   // Public class accessible from anywhere
}
```

private int myPrivateVariable;

// Private variable only accessible within the same class

These are some of the fundamental syntax elements in C#. Understanding and following these rules is essential for writing correct and maintainable code in the language. Proper syntax adherence ensures that your code is not only valid but also readable and comprehensible to other developers.

.NET Framework and .NET Core:

The .NET Framework and .NET Core (now evolved into .NET 5 and later) are two related but distinct software development platforms created by Microsoft for building various types of applications. Both frameworks are based on the Common Language Runtime (CLR) and share many core concepts and libraries, but they have some key differences:

.NET Framework:

Windows-Centric: The .NET Framework was initially designed for Windows-based applications. It is tightly integrated with the Windows operating system and provides extensive libraries for building desktop applications, web applications using ASP.NET, and other Windows-specific software.

Mature: The .NET Framework has been around since the early 2000s and has a long history of development and use in enterprise applications.

Full Framework: It's often referred to as the "full" .NET Framework because it includes a comprehensive set of libraries, such as Windows Presentation Foundation (WPF) for desktop applications and Windows Forms.

Compatibility: Older applications built on the .NET Framework may not be immediately compatible with newer .NET Core versions without modifications.

Windows-Only: .NET Framework applications are primarily designed to run on Windows operating systems. While there were attempts to make it cross-platform (e.g., Mono project), it remained largely Windows-centric.

Lack of Modern Features: Over time, the .NET Framework became feature-heavy and was considered less suitable for modern, lightweight, and cross-platform development.

.NET Core (and Beyond, like .NET 5+):

Cross-Platform: .NET Core was designed with cross-platform development in mind from the start. It can be used to build applications that run on Windows, Linux, and macOS.

Open-Source: .NET Core is open-source and developed in the open on platforms like GitHub, allowing for community contributions and transparency.

Modular: .NET Core uses a more modular and lightweight approach, allowing developers to include only the necessary libraries and dependencies, reducing the size of applications.

Unified Platform: Starting from .NET 5, Microsoft unified the .NET ecosystem under a single platform called ".NET," which includes .NET Core, Xamarin, and .NET Framework components. This means there's no more ".NET Core" and ".NET Framework" differentiation.

Modern Features: .NET 5 and later versions introduce modern features and performance improvements, making it more suitable for modern development scenarios.

Side-by-Side Versioning: You can install multiple versions of .NET (such as .NET 5, .NET 6, etc.) side by side on a machine, which helps with backward compatibility and allows you to upgrade applications at your own pace.

Web Framework: ASP.NET Core (part of .NET Core) is the modern web framework that replaced ASP.NET in the .NET Framework. It is lightweight, cross-platform, and designed for modern web development.

Microsoft has been encouraging developers to migrate their existing .NET Framework applications to .NET Core and now to the unified .NET platform for improved performance, cross-platform compatibility, and access to modern development features. This unified .NET platform aims to offer the best of both worlds, combining the strengths of .NET Framework and .NET Core while providing a clear path forward for .NET developers.

Asynchronous Programming:

Asynchronous programming is a programming paradigm that allows you to write code that can perform non-blocking operations. This means that while one part of your program is waiting for something to complete, other parts of your program can continue to execute. Asynchronous programming is crucial for building responsive and scalable software, especially in situations where tasks might take a long time to complete, such as network operations or file I/O.

In C# and many other modern programming languages, asynchronous programming is typically implemented using the async and await keywords. Here's an overview of how asynchronous programming works in C#:

Async and Await Keywords: The async keyword is used to declare a method as asynchronous, and the await keyword is used to indicate where the program should pause and wait for an asynchronous operation to complete.

Task-Based Asynchronous Pattern (TAP): Asynchronous methods often return a Task or Task<T>. These objects represent a pending operation. You can use await to asynchronously wait for the completion of the task without blocking the thread.

Non-Blocking: When you use await, the program doesn't block the current thread while waiting for the operation to finish. Instead, the thread can be used to execute other tasks, which improves the overall responsiveness of your application.

Here's a simple example of asynchronous programming in C#:

Csharp Code

```
using System;
using System.Net.Http;
using System.Threading.Tasks;

class Program
{
    static async Task Main()
    {
        await DownloadAndPrintContentAsync();
        Console.WriteLine("This will execute while downloading is in progress.");
    }

    static async Task DownloadAndPrintContentAsync()
    {
        using (HttpClient client = new HttpClient())
        {
            string url = "https://example.com";
            string content = await client.GetStringAsync(url);
            Console.WriteLine(content);
        }
    }
}
```

In this example, the DownloadAndPrintContentAsync method downloads content from a website asynchronously using HttpClient. While waiting for the download to complete (during the await operation), the program can continue executing other tasks. This ensures that the application remains responsive and doesn't freeze while waiting for the web request to finish.

9.Testing methodology:

Software testing methods are traditionally divided into black box testing and white box testing. These two approaches are used to describe the point of view that a test engineer takes when designing test cases.

1) **Black box testing -**
   Black box testing treats the software as a "black box," without any knowledge of internal implementation. Black box testing methods include: equivalence partitioning, boundary value analysis, all-pairs testing, fuzz testing, model-based testing, traceability matrix, exploratory testing and specification-based testing.
2) **White box testing**–
   White box testing, by contrast to black box testing, is when the tester has access to the internal data structures and algorithms (and the code that implement these).White box testing methods can also be used to evaluate the completeness of a test suite that was created with black box testing methods. This allows the software team to examine parts of a system that are rarely tested and ensures that the most important function points have been tested.
3) **Grey Box Testing**–
   Grey box testing involves having access to internal data structures and algorithms for purposes of designing the test cases, but testing at the user, or black-box level. Manipulating input data and formatting output do not qualify as "grey box," because the input and output are clearly outside of the "black-box" that we are calling the system under test. This distinction is particularly important when conducting integration testing between two modules of code written by two different developers, where only the interfaces are exposed for test. Grey box testing may also include reverse engineering to determine, for instance, boundary values or error messages.
4) **Acceptance testing –**
   Acceptance testing can mean one of two things:

1. A smoke test is used as an acceptance test prior to introducing a build to the main testing process.
2. Acceptance testing performed by the customer is known as user acceptance testing

5) **Regression Testing–** Regression testing is any type of software testing that seeks to uncover software regressions. Such regression occurs whenever software functionality that was previously working correctly stops working as intended. Typically regressions occur

as an unintended consequence of program changes. Common methods of regression testing include re-running previously run tests and checking whether previously fixed faults have re-emerged.

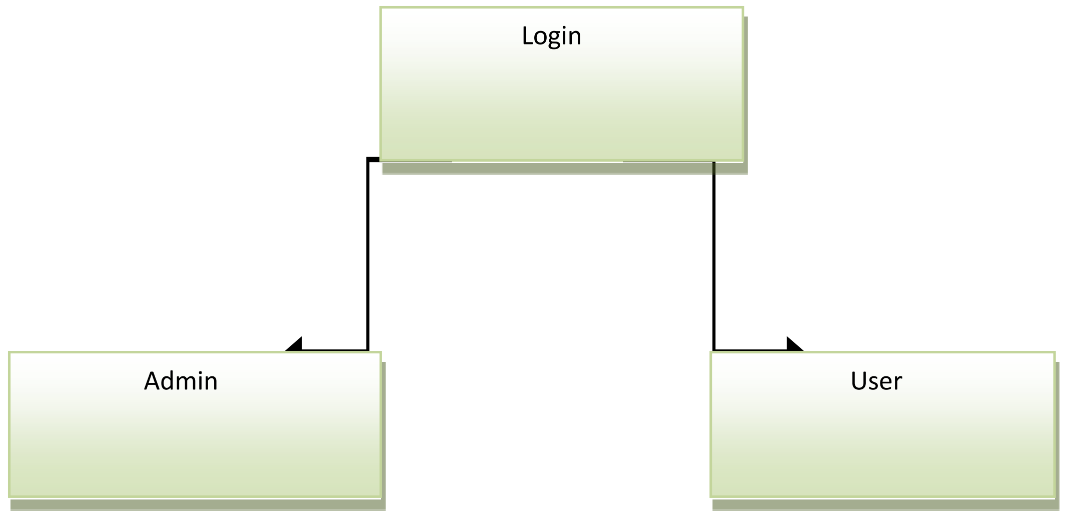
6) **Non -Functional Software Testing–**

Special methods exist to test non-functional aspects of software.

- Performance testing checks to see if the software can handle large quantities of data or users. This is generally referred to as software scalability. This activity of Non Functional Software Testing is often times referred to as Load Testing.
- Stability testing checks to see if the software can continuously function well in or above an acceptable period. This activity of Non Functional Software Testing is often times referred to as indurations test.
- Usability testing is needed to check if the user interface is easy to use and understand.
- Security testing is essential for software which processes confidential data and to prevent system intrusion by hackers.
- Internationalization and localization is needed to test these aspects of software, for which a pseudo localization method can be used.

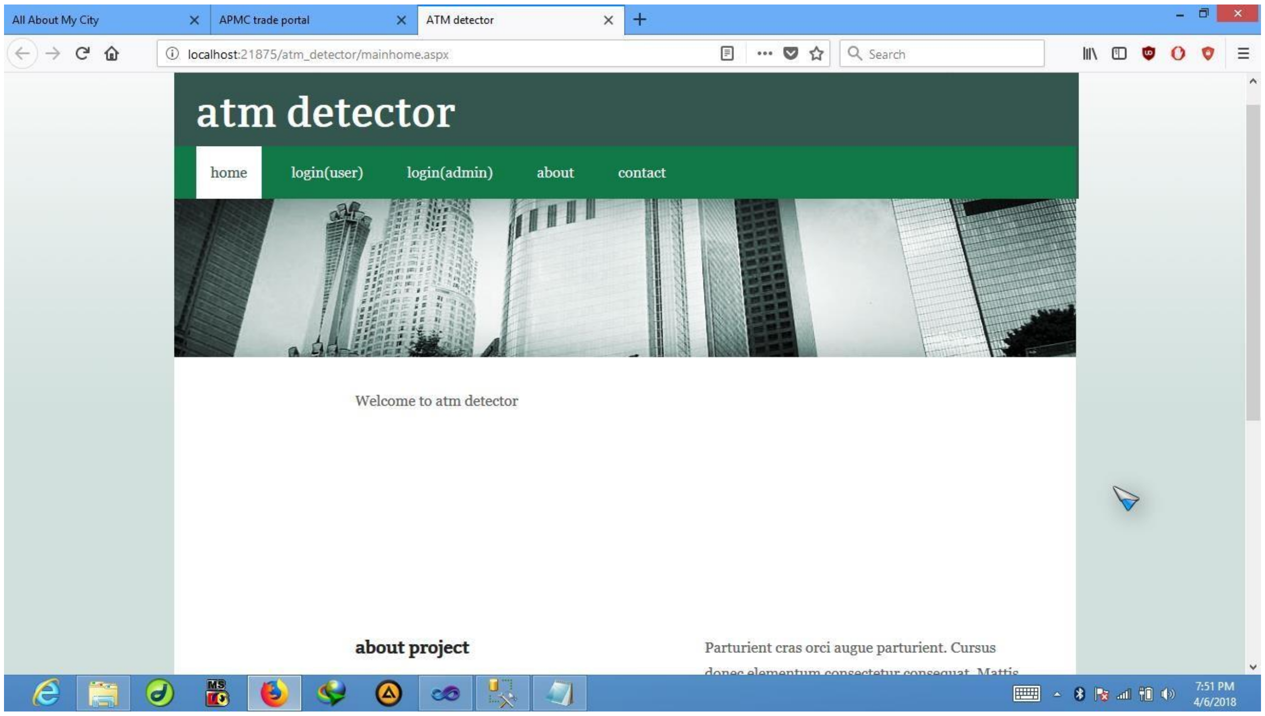
10.Data Flow Diagrams:

Login Page:



11.Screenshots:

Home Page:

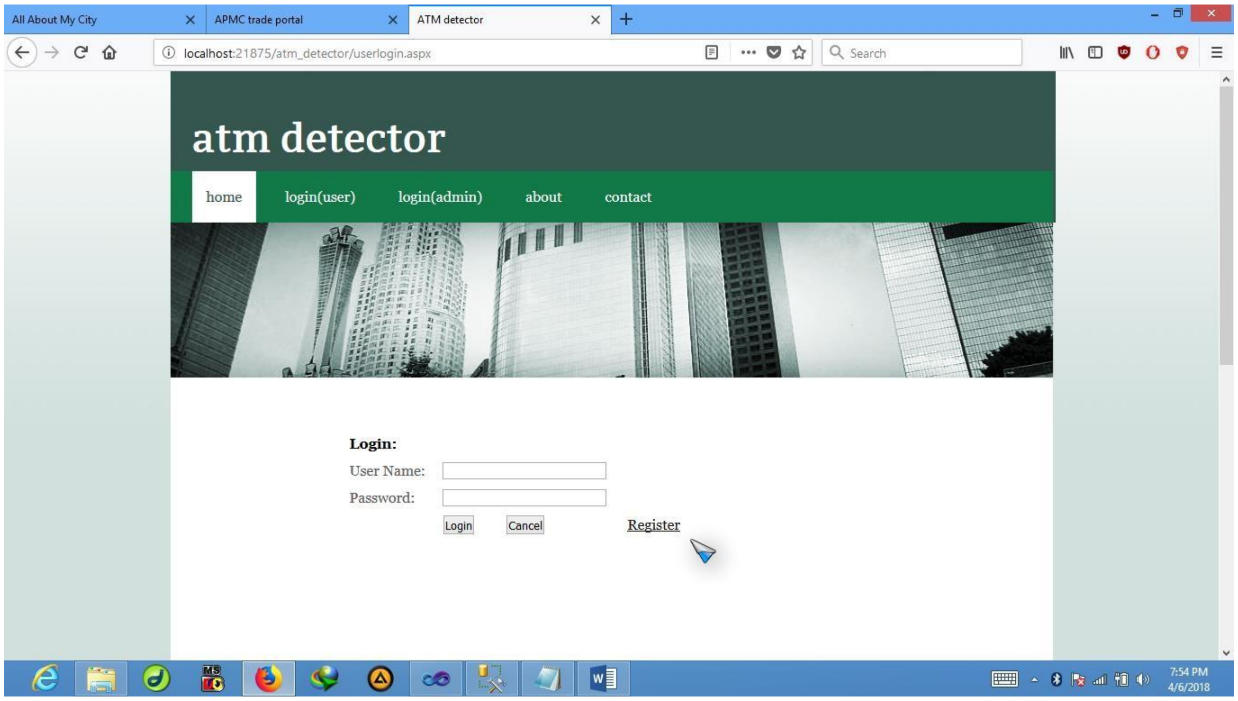


Welcome to atm detector

about project

Parturient cras orci augue parturient. Cursus donec elementum consectetur consequat. Mattis

User Login:

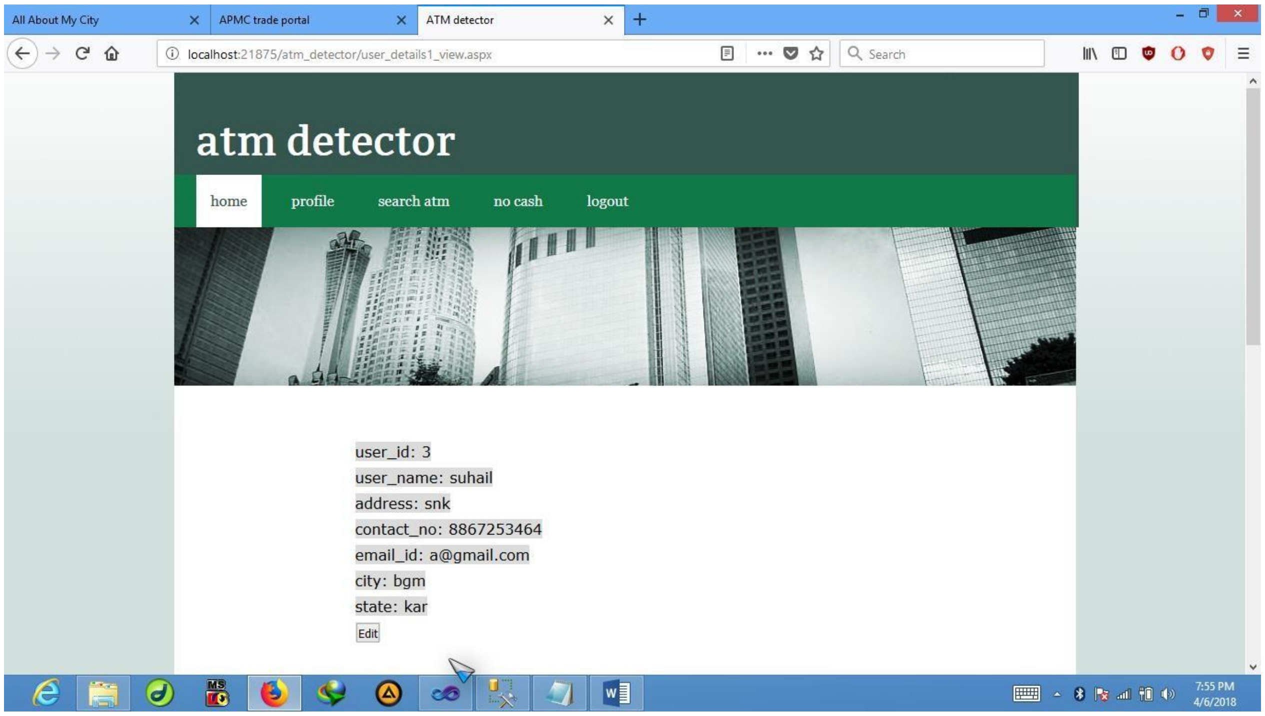


Login:

User Name:

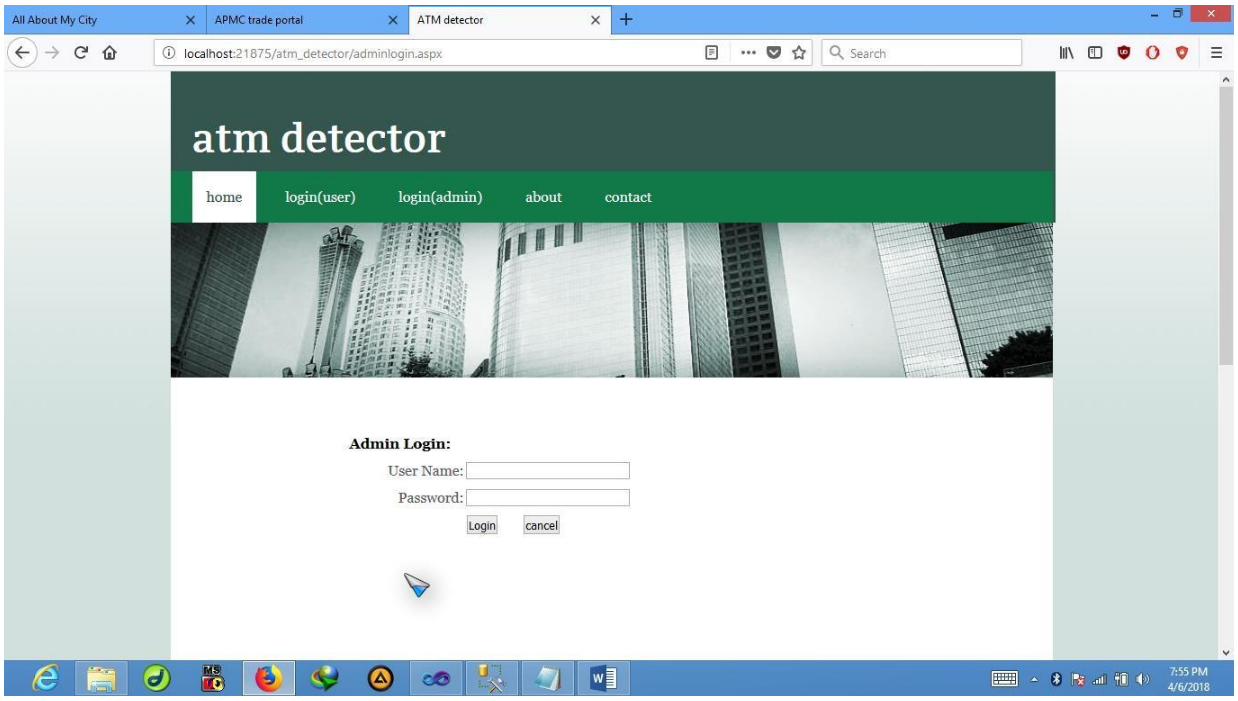
Password:

User Module:

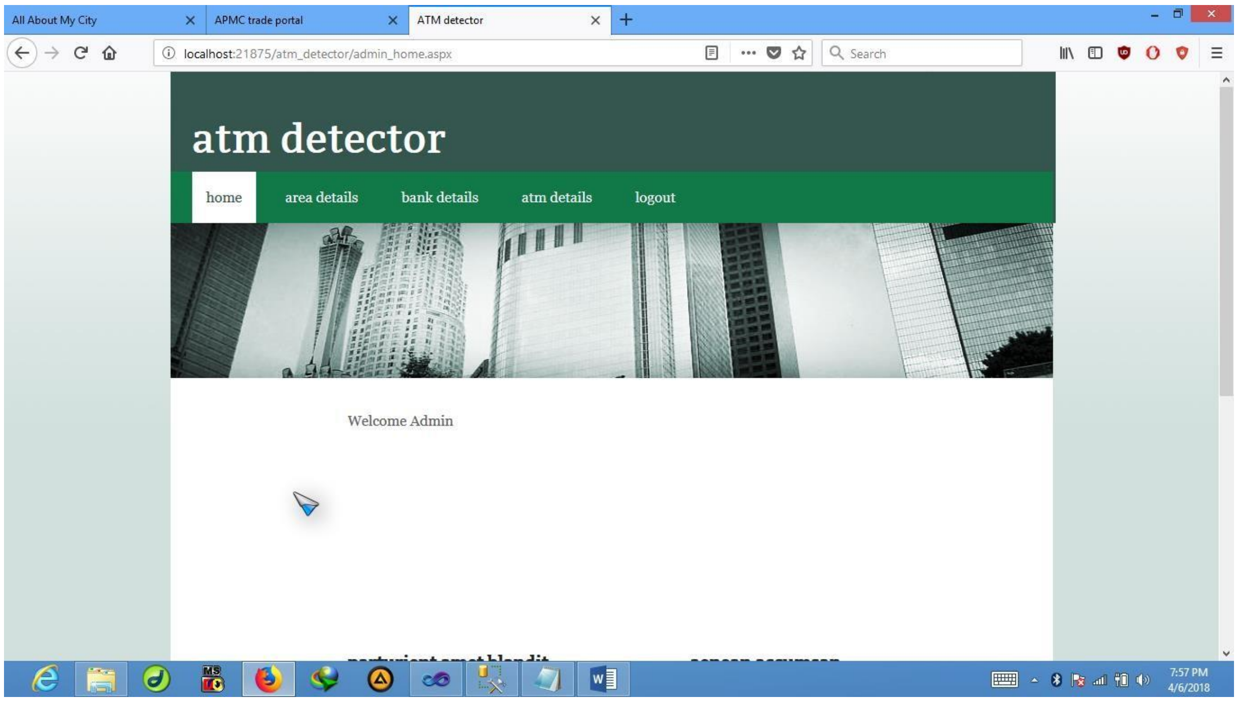


user_id: 3
user_name: suhail
address: snk
contact_no: 8867253464
email_id: a@gmail.com
city: bgm
state: kar

Admin Login:



Admin Module:



Welcome Admin

12.Conclusion :

Since the introduction of ATM we find people rushing to the ATM to withdraw money rather than approaching the bank. Most often we find people trying to locate an ATM and keep asking passersby about the nearest ATM. It is also noticed that they end up at the ATM to find that there are no funds. They then try and locate another ATM. The application proposes to automate an environment where the administrator publishes the list of ATM’s from different banks, location, map (image). The administrator also allows the details of the cash being loaded. This allows the public to identify the ATM's and funds available and its functioning. The system allows the administrator to register banker. The Banker registers the customer details and their ATM’s.

13.Future Enhancement:

- Updating ATMs to google maps with GPS location for more accurate access.
- Show the funds available status to the customers.