**BLOCKING SERVICE ATTACK**

A Mini Project Report  
submitted in fulfilment of the requirements  
for the award of degree of   
**BACHELOR OF TECHNOLOGY  
IN  
COMPUTER SCIENCE AND ENGINEERING**

By

**RITWIK KUMAR (13QH5A0544)  
A.PRAKASH SINHA (13QH1A0505)  
SHAHNAWAZ ALAM (13QH1A0547)  
RAJ SOREN (13QH1A0541)**

Guide

**Mr. S.k.Juber**

**Assistant Professor**



**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

**HOLY MARY INSTITUTE OF TECHNOLOGY**

(Approved by AICTE, New Delhi, Affiliated to JNTUH)

Bogram (Vill), Keesara (Mndl), RangaReddy (Dist)-501301

2013-2014

**BLOCKING SERVICE ATTACK**

A Mini project report

Submitted in fulfilment of the requirements

For the award of degree of

**BACHELOR OF TECHNOLOGY**

**IN**

**COMPUTER SCIENCE AND ENGINEERING**

**BY**

**RITWIK KUMAR (13QH5A0544)  
A.PRAKASH SINHA (13QH1A0505)  
SHAHNAWAZ ALAM (13QH1A0547)  
RAJ SOREN (13QH1A0541)**

**Under the Esteemed Guidance of**

**Mr. S.k. Juber**

**Assistant Professor**



**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

**HOLY MARY INSTITUTE OF TECHNOLOGY**

**(Approved by AICTE, New Delhi, Affiliated to JNTUH)**

**Hyderabad-500090**

**2013-2014**

**HOLY MARY INSTITUTE OF TECHNOLOGY**

**(Approved by AICTE, New Delhi, Affiliated to JNTUH)**

**Hyderabad-500090**

**2013-2014**

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**



**CERTIFICATE**

This is to certify that the Project report entitled **“Blocking Service Attack”** is being submitted by the following student in partial fulfilment of the requirements for the award of degree of **Bachelor of Technology** in **Computer Science and Engineering** from **Jawaharlal Nehru Technological University,** Hyderabad, Andhra Pradesh is record of bonafide work carried out during the academic year 2013-2014.

**RITWIK KUMAR (13QH5A0544)  
A.PRAKASH SINHA (13QH1A0505)  
SHAHNAWAZ ALAM (13QH1A0547)  
RAJ SOREN (13QH1A0541)**

**INTERNAL GUIDE HEAD OF THE DEPARTMENT**

**Mr. S.k.Juber Mr. SomaSekhar**

Assistant Professor Head of the Department

Department of CSE Department of CSE

**EXTERNAL EXAMINER**

**DECLARATION**

****

**HOLY MARY INSTITUTE OF TECHNOLOGY**

(Recognized by A.I.C.T.E Affiliated to JNTU, Hyderabad)

Bogaram(v), Keesara(m), R.R.Dist 501301.

We hereby declare that this submission is our own work and that, to the best of our knowledge and belief, it contains no material previously published or written by another person nor material which to a substantial extent has been accepted for the award of any other degree or diploma of the university or other institute of higher learning, except where due acknowledgment has been made in the text.

**RITWIK KUMAR (13QH5A0544)  
A.PRAKASH SINHA (13QH1A0505)  
SHAHNAWAZ ALAM (13QH1A0547)  
RAJ SOREN (13QH1A0541)**

**ACKNOWLEDGEMENT**

The successful completion of any task is not possible without proper suggestions guidance and environment. Combination of these three factors acts like backbone to my project titled “**Blocking Service Attack”.**

We express our sincere thanks to our guide, Department of Computer Science and Engineering, **Mr. S.k. Juber** for her valuable suggestions during our course period, timely help, guidance and providing us with most essential materials required for the completion of this work.

We express our since thanks to our HOD, Department of Computer Science and Engineering.

**Mr. G. SOMASEKHAR** for her valuable suggestions during our course period, timely help, guidance and providing us with the most essential materials required for the completion of this work.

We would like to thank our principal, **Dr. B.S. REDDAPPA** for providing support and simulating environment. We would like express my gratitude to the management of college name for providing me a pleasant environment and excellent and excellent laboratory facilities.

We special thanks to project coordinator **Mrs. V.B. RAMANA** and all the faculty of CSE Department for giving the required information in doing my project work. Not to forget, I thank all the non-teaching staff, and my friends who had directly or indirectly helped and supported me in completing my project in time.

We are thankful to all teaching and non-teaching staff of the Department of Computer Science & Engineering for the cooperation given for the successful completion of my project.

Finally we wish to convey my gratitude to my parents who fostered all the requirements and facilities that I need.

**RITWIK KUMAR (13QH5A0544)  
A.PRAKASH SINHA (13QH1A0505)  
SHAHNAWAZ ALAM (13QH1A0547)  
RAJ SOREN (13QH1A0541)**

**INDEX**

**CONTENTS PAGE NO**

**ABSTRACT i**

**LIST OF DIAGRAMS ii**

**1. INTRODUCTION 01**

**2. LITERATURE SURVEY 03**

2.1 Introduction

2.2 Existing System

2.3 Proposed System

**3. REQUIREMENT SPECIFICATION 04**

3.1 Introduction

3.2 Requirement specification

3.3 Functional requirement

**4. DESIGN 26**

4.1 UML Diagrams

**5. IMPLEMENTATION 37**

**6. CODING 45**

**7. TESTING 55**

7.1 Test plan

7.2 Test cases

**8. OUTPUTS 63**

**9. CONCLUSION 74**

**10. FUTURE SCOPE 75**

**11. BIBLIOGRAPHY 76**

**ABSTRACT**

In recent years, a number of puzzle-based defense mechanisms have been proposed against flooding denial-of-service (DOS) attacks in networks. Nonetheless, these mechanisms have not been designed through formal approaches and thereby some important design issues such as effectiveness and optimality have remained unresolved. This paper utilizes game theory to propose a series of optimal puzzle-based strategies for handling increasingly sophisticated flooding attack scenarios. In doing so, the solution concept of Nash equilibrium is used in a prescriptive way, where the defender takes his part in the solution as an optimum defense against rational attackers. This study culminates in a strategy for handling distributed attacks from an unknown number of sources.

i

**LIST OF DIAGRAMS**

**FIGURE NAME OF FIGURE**

Fig: 4.1 Admin Use Case Diagram

Fig: 4.2 User Use Case Diagram

Fig: 4.3 Class Diagram

Fig: 4.4 Activity Diagram

Fig: 4.5 Sequence Diagram

Fig: 4.6 Collaboration Diagram

Fig: 8.1 Home Page

Fig: 8.2 Registration Page

Fig: 8.3 Registration Successfully Page

Fig: 8.4 Login Page

Fig: 8.5 User Login Page

Fig: 8.6 User Fileupload Page

Fig: 8.7 Fileupload Successfully Page

Fig: 8.8 User View Files Page

Fig: 8.9 Admin Login Page

Fig: 8.10 Admin View Files Page

Fig: 8.11 Admin View Profile Page

ii

**CHAPTER-01**

**INTRODUCTION**

**User-**The user can login into the system, browse the file and solve the puzzle and get the information.

**Admin-**The full control over the users and their functions

## In recent years security concerned issues has received enormous attention in networked system because of availability of services. Networked systems are vulnerable to DoS (Denial of Services) attack. A DoS attack characterizes a malicious behaviour preventing the legitimate users of a network from using the services provided by that network. Flooding attacks and Logic attacks are the two principal classes of DoS attack. The database is perfectly designed sticking to the rules Data Consistency, Data Dependency, Non Redundancy and Data Integrity.

**1.1 SCOPE:**

The Theoretical background of the “GAME THEORY” mainly used to know how to get interacted with the USER and to know the updates and to obtain the solutions to our problems.

**1.1.1 Existing system with limitations:**

In existing system, there is a drawback i.e. server may get crashed by attackers flooding attacks which is nothing but number of requests coming from the clients and the reason behind server getting crashed is that more number of requests coming in from clients or attackers and also may slow sown the processing speed of the server.

**1.1.2 Proposed system with features:**

In our proposed system, we are blocking the requests of attackers by using puzzle based defense mechanism i.e. before reaching the server user need to firstly solve the puzzle being displayed and only after the success it is assumed that the user is a genuine user and his/her request will reach server and then the processing happens.

**Advantages of proposed system:**

We can stop the unnecessary attacks coming from the malicious users and thereby increasing the server functionality and also its processing time will be less as only the reliable or genuine requests reach server for their requests.

**CHAPTER-02**

**LITERATURE SURVEY**

**2.1 INTRODUCTION**

Plagiarism is a 'wrongful appropriation' and 'stealing and publication' of another authors language, thoughts, ideas or expression .It is the act of taking another person's writing, conversation and even ideas and passing it off as your own. To avoid all such cases we use Article Comparer. It shows the percentage of copied content which is not unique. By this it is easy to evaluate the documents which are plagiarized. Recently, the problem of plagiarism is becoming an important issue in many debates in the field of education and technology. By using this article comparer we can create our new articles which will not match with already existed articles.

**2.2 EXISTING SYSTEM**

There is a software called Plagiarism checker which is already existing in today’s world which is used to show the copied content or information. In that particular software we need to paste the content in the text box to check plagiarism. But there is no option to upload files and check.

**2.3 PROPOSED SYSTEM WITH FEATURE**

In the “Article Comparer” project we have a feature of uploading files. Here we need to upload the files which are in “.txt” and “.doc” format. No need of copying and pasting, directly we can upload files and check for plagiarism.

**CHAPTER-03**

**REQUIREMENT SPECIFICATION**

**3.1 Software Requirements:**

* **Operating system:** WINDOWS 7,8,10.
* **Front end: MICROSOFT VISUAL STUDIO 2012:** Microsoft Visual Studio is an Integrated Development Environment from Microsoft .It can be used to develop console and graphical user interface applications along with windows forms applications , websites, web applications and web services in both native code together with managed code for all platforms supported by Microsoft Windows,.NET framework,.NET compact Framework and Microsoft Silver light .It also supports the c# that we use to create web site.
* **Back end: SQL SERVER 2008 R2:** In this SQL Server Management Studio is a software application first launched with the Microsoft SQL Server 2005 that is used for configuring, managing, and administering all components with in Microsoft SQL Server.
* **Microsoft Office Word 2013:** We use Microsoft Office Word 2013 to do our documentation of this project. This is very important for us to do this project. We have use the feature of text box to draw the hierarchical chart to describe the various of subsystem, modules and sub-modules in the system. It also uses to check our spelling and grammar and justify all the words to make our document look nicer.

**3.2 Software Requirements:**

Language Specification C# with Asp.Net

Databases SQLSERVER 2008

Framework .Net 4.0 IDE: Visual Studio 2010

Operating Systems Windows 2000, windows NT 4.0, Windows & Above

**Hardware Requirements:**

Processor Pentium IV & LATERs

RAM 512MB RAM & above

Hard Disk 80GB & above

**Functional Requirements:**

” In software engineering, a **functional requirement** defines a function of a software system or its component. A function is described as a set of inputs, the behavior, and outputs (see also software).

Functional requirements may be calculations, technical details, data manipulation and processing and other specific functionality that define *what* a system is supposed to accomplish. Behavioral requirements describing all the cases where the system uses the functional requirements are captured in use cases. Functional requirements are supported by non-functional requirements (also known as *quality requirements*), which impose constraints on the design or implementation (such as performance requirements, security, or reliability). *How* a system implements functional requirements is detailed in the system *design*.

As defined in requirements engineering, functional requirements specify particular results of a system. This should be contrasted with non-functional requirements which specify overall characteristics such as cost and reliability. Functional requirements drive the *application architecture* of a system, while non-functional requirements drive the *technical architecture* of a system.

The Functional Requirements of our site

**Non-Functional Requirements:**

In systems engineering and requirements engineering, a **non-functional requirement** is a requirement that specifies criteria that can be used to judge the operation of a system, rather than specific behaviors. This should be contrasted with functional requirements that define specific behavior or functions

In general, functional requirements define what a system is supposed to *do* whereas non-functional requirements define how a system is supposed to *be*. Non-functional requirements are often called **qualities** of a system. Other terms for non-functional requirements are "constraints", "quality attributes", "quality goals" and "quality of service requirements". Qualities, that is, non-functional requirements, can be divided into two main categories:

1. Execution qualities, such as security and usability, which are observable at run time.
2. Evolution qualities, such as testability, maintainability, extensibility and scalability, which are embodied in the static structure of the software system.

.**NET Framework:**

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

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

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

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

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

Internet Explorer is an example of an unmanaged application that hosts the runtime (in the form of a MIME type extension). Using Internet Explorer to host the runtime enables you to embed managed components or Web Forms controls in HTML documents. Hosting the runtime in this way makes managed mobile code (similar to Microsoft® ActiveX® controls) possible, but with significant improvements that only managed code can offer, such as semi-trusted execution and secure isolated file storage.

The following illustration shows the relationship of the common language runtime and the class library to your applications and to the overall system. The illustration also shows how managed code operates within a larger architecture.

## Features of the Common Language Runtime:

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

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

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

The runtime also enforces code robustness by implementing a strict type- and code-verification infrastructure called the common type system (CTS). The CTS ensures that all managed code is self-describing. The various Microsoft and third-party language compilers generate managed code that conforms to the CTS. This means that managed code can consume other managed types and instances, while strictly enforcing type fidelity and type safety.

In addition, the managed environment of the runtime eliminates many common software issues. For example, the runtime automatically handles object layout and manages references to objects, releasing them when they are no longer being used. This automatic memory management resolves the two most common application errors, memory leaks and invalid memory references.

The runtime also accelerates developer productivity. For example, programmers can write applications in their development language of choice, yet take full advantage of the runtime, the class library, and components written in other languages by other developers. Any compiler vendor who chooses to target the runtime can do so. Language compilers that target the .NET Framework make the features of the .NET Framework available to existing code written in that language, greatly easing the migration process for existing applications.

While the runtime is designed for the software of the future, it also supports software of today and yesterday. Interoperability between managed and unmanaged code enables developers to continue to use necessary COM components and DLLs.

The runtime is designed to enhance performance. Although the common language runtime provides many standard runtime services, managed code is never interpreted. A feature called just-in-time (JIT) compiling enables all managed code to run in the native machine language of the system on which it is executing. Meanwhile, the memory manager removes the possibilities of fragmented memory and increases memory locality-of-reference to further increase performance.

Finally, the runtime can be hosted by high-performance, server-side applications, such as Microsoft® SQL Server™ and Internet Information Services (IIS). This infrastructure enables you to use managed code to write your business logic, while still enjoying the superior performance of the industry's best enterprise servers that support runtime hosting.

## .NET Framework Class Library:

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

For example, the .NET Framework collection classes implement a set of interfaces that you can use to develop your own collection classes. Your collection classes will blend seamlessly with the classes in the .NET Framework.

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

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

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

**C#.NET**

# Introduction to Web Forms (C#.NET):

Web Forms is the new platform for Microsoft Web application development, based on the .NET Framework. This framework provides a clear, object-oriented, extensible set of classes that enable you to develop rich Web applications. Additionally, Web Forms can act as the local user interface in a multi-tier distributed solution. Web Forms is a framework for building Windows client applications that utilize the common language runtime. Web Forms applications can be written in any language that the common language runtime supports.

**ACTIVE X DATA OBJECTS.NET:**

**ADO.NET Overview:**

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

ADO.NET uses some ADO objects, such as the Connection and Command objects, and also introduces new objects. Key new ADO.NET objects include the Dataset, Data Reader, and Data Adapter.

The important distinction between this evolved stage of ADO.NET and previous data architectures is that there exists an object -- the Dataset -- that is separate and distinct from any data stores. Because of that, the Dataset functions as a standalone entity. You can think of the Dataset as an always disconnected record set that knows nothing about the source or destination of the data it contains. Inside a Dataset, much like in a database, there are tables, columns, relationships, constraints, views, and so forth.

A Data Adapter is the object that connects to the database to fill the Dataset. Then, it connects back to the database to update the data there, based on operations performed while the Dataset held the data. In the past, data processing has been primarily connection-based. Now, in an effort to make multi-tiered apps more efficient, data processing is turning to a message-based approach that revolves around chunks of information. At the center of this approach is the Data Adapter, which provides a bridge to retrieve and save data between a Dataset and its source data store. It accomplishes this by means of requests to the appropriate SQL commands made against the data store.

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

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

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

* **Connections**. For connection to and managing transactions against a database.
* **Commands**. For issuing SQL commands against a database.
* **Data Readers**. For reading a forward-only stream of data records from a MSACCESS data source.
* **Datasets** For storing, remoting and programming against flat data, XML data and relational data.
* **Data Adapters** For pushing data into a Dataset, and reconciling data against a database.

When dealing with connections to a database, there are two different options: MSACCESS .NET Data Provider (System.Data.SqlClient) and OLE DB .NET Data Provider (System.Data.OleDb). In these samples we will use the MSACCESS .NET Data Provider. These are written to talk directly to Microsoft MSACCESS. The OLE DB .NET Data Provider is used to talk to any OLE DB provider (as it uses OLE DB underneath).

**Connections:**

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

**Commands:**

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

**Data Readers:**

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

**Datasets and Data Adapters:**

**Datasets:**  
 The Dataset object is similar to the ADO Record set object, but more powerful, and with one other important distinction: the Dataset is always disconnected. The Dataset object represents a cache of data, with database-like structures such as tables, columns, relationships, and constraints. However, though a Dataset can and does behave much like a database, it is important to remember that Dataset objects do not interact directly with databases, or other source data. This allows the developer to work with a programming model that is always consistent, regardless of where the source data resides. Data coming from a database, an XML file, from code, or user input can all be placed into Dataset objects. Then, as changes are made to the Dataset they can be tracked and verified before updating the source data. The Get Changes method of the Dataset object actually creates a second Dataset that contains only the changes to the data. This Dataset is then used by a Data Adapter (or other objects) to update the original data source.

The Dataset has many XML characteristics, including the ability to produce and consume XML data and XML schemas. XML schemas can be used to describe schemas interchanged via Web Services. In fact, a Dataset with a schema can actually be compiled for type safety and statement completion.

**Data Adapters (OLEDB/SQL):**

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

The Data Adapter object uses commands to update the data source after changes have been made to the Dataset. Using the Fill method of the Data Adapter calls the SELECT command; using the Update method calls the INSERT, UPDATE or DELETES command for each changed row. You can explicitly set these commands in order to control the statements used at runtime to resolve changes, including the use of stored procedures. For ad-hoc scenarios, a Command Builder object can generate these at run-time based upon a select statement. However, this run-time generation requires an extra round-trip to the server in order to gather required metadata, so explicitly providing the INSERT, UPDATE, and DELETE commands at design time will result in better run-time performance.

Data, and navigate data relationships ADO.NET is the next evolution of ADO for the .Net Framework.

1. ADO.NET was created with n-Tier, statelessness and XML in the forefront. Two new objects, the Dataset and Data Adapter, are provided for these scenarios.
2. ADO.NET can be used to get data from a stream, or to store data in a cache for updates.
3. There is a lot more information about ADO.NET in the documentation.
4. Remember, you can execute a command directly against the database in order to do inserts, updates, and deletes. You don't need to first put data into a Dataset in order to insert, update, or delete it. Also, you can use a Dataset to bind to the data.

**ASP.Net**

## Server Application Development:

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

## The following illustration shows a basic network schema with managed code running in different server environments. Servers such as IIS and MSACCESS can perform standard operations while your application logic executes through the managed code.

**Server-side managed code:**

ASP.NET is the hosting environment that enables developers to use the .NET Framework to target Web-based applications. However, ASP.NET is more than just a runtime host; it is a complete architecture for developing Web sites and Internet-distributed objects using managed code. Both Web Forms and XML Web services use IIS and ASP.NET as the publishing mechanism for applications, and both have a collection of supporting classes in the .NE XML Web services, an important evolution in Web-based technology, are distributed, server-side application components similar to common Web sites. However, unlike Web-based applications, XML Web services components have no UI and are not targeted for browsers such as Internet Explorer and Netscape Navigator. Instead, XML Web services consist of reusable software components designed to be consumed by other applications, such as traditional client applications, Web-based applications, or even other XML Web services. As a result, XML Web services technology is rapidly moving application development and deployment into the highly distributed environment of the Internet. If you have used earlier versions of ASP technology, you will immediately notice the improvements that ASP.NET and Web Forms offers. For example, you can develop Web Forms pages in any language that supports the .NET Framework. In addition, your code no longer needs to share the same file with your HTTP text (although it can continue to do so if you prefer). Web Forms pages execute in native machine language because, like any other managed application, they take full advantage of the runtime. In contrast, unmanaged ASP pages are always scripted and interpreted. ASP.NET pages are faster, more functional, and easier to develop than unmanaged ASP pages because they interact with the runtime like any managed application.

The .NET Framework also provides a collection of classes and tools to aid in development and consumption of XML Web services applications. XML Web services are built on standards such as SOAP (a remote procedure-call protocol), XML (an extensible data format), and WSDL (the Web Services Description Language). The .NET Framework is built on these standards to promote interoperability with non-Microsoft solutions.

For example, the Web Services Description Language tool included with the .NET Framework SDK can query an XML Web service published on the Web, parse its WSDL description, and produce C# or Visual Basic source code that your application can use to become a client of the XML Web service. The source code can create classes derived from classes in the class library that handle all the underlying communication using SOAP and XML parsing. Although you can use the class library to consume XML Web services directly, the Web Services Description Language tool and the other tools contained in the SDK facilitate your development efforts with the .NET Framework. If you develop and publish your own XML Web service, the .NET Framework provides a set of classes that conform to all the underlying communication standards, such as SOAP, WSDL, and XML. Using those classes enables you to focus on the logic of your service, without concerning yourself with the communications infrastructure required by distributed software development.

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

**SQL SERVER**

**DATABASE:**

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

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

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

**SQL Server Tables:**

SQL Server stores records relating to each other in a table. Different tables are created for the various groups of information. Related tables are grouped together to form a database.

**Primary Key:**

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

**Relational Database:**

Sometimes all the information of interest to a business operation can be stored in one table. SQL Server makes it very easy to link the data in multiple tables. Matching an employee to the department in which they work is one example. This is what makes SQL Server a relational database management system, or RDBMS. It stores data in two or more tables and enables you to define relationships between the tables and enables you to define relationships between the tables.

**Foreign Key:**

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

**Referential Integrity:**

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

**Data Abstraction:**

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

**Physical Level:** This is the lowest level of abstraction at which one describes how the data are actually stored.

**Conceptual Level:** At this level of database abstraction all the attributed and what data are actually stored is described and entries and relationship among them.

**View Level:** This is the highest level of abstraction at which one describes only part of the database.

**Advantages of RDBMS:**

1. Redundancy can be avoided
2. Inconsistency can be eliminated
3. Data can be Shared
4. Standards can be enforced
5. Security restrictions can be applied
6. Integrity can be maintained
7. Conflicting requirements can be balanced
8. Data independence can be achieved.

**Disadvantages of DBMS:**

A significant disadvantage of the DBMS system is cost. In addition to the cost of purchasing of developing the software, the hardware has to be upgraded to allow for the extensive programs and the workspace required for their execution and storage. While centralization reduces duplication, the lack of duplication requires that the database be adequately backed up so that in case of failure the data can be recovered.

**FEATURES OF SQL SERVER (RDBMS):**

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

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

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

SQL SERVER with transactions processing option offers two features which contribute to very high level of transaction processing throughput, which are

1. The row level lock manager

**Enterprise wide Data Sharing:**

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

**Portability:**

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

**Open Systems:**

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

**Distributed Data Sharing:**

SQL Server’s networking and distributed database capabilities to access data stored on remote server with the same ease as if the information was stored on a single local computer. A single SQL statement can access data at multiple sites. You can store data where system requirements such as performance, security or availability dictate.

**Unmatched Performance:**

The most advanced architecture in the industry allows the SQL SERVER DBMS to deliver unmatched performance.

**Sophisticated Concurrency Control:**

Real World applications demand access to critical data. With most database Systems application becomes “contention bound” – which performance is limited not by the CPU power or by disk I/O, but user waiting on one another for data access. SQL Server employs full, unrestricted row-level locking and contention free queries to minimize and in many cases entirely eliminates contention wait times.

**DATA TABLES:**

**DATAFLOW DIAGRAMS**

**3.4 Data flow diagrams:**

**DATAFLOW DIAGRAMS:**

Data flow diagrams (“bubble charts”) are directed graphs in which the nodes specify processing activities and the arcs specify data items transmitted between processing nodes. A data flow diagram might represent data flow between individual statements or blocks of statements in a routine, data flow between sequential routines, data flow between concurrent process, or data flow in a distributed computing system, where each node represents a geographically remote processing unit.

Unlike flow charts, data flow diagrams do not indicate decision logic or condition under which various nodes in the diagram might be activated. Data flow diagrams can be expressed using special symbols. These symbols can be used to denote processing nodes, data nodes, data sources and data stores.

A Data Flow Diagram (DFD) is a graphical technique that depicts information flow and transforms that are applied as data move to represent a system or software at any level of abstraction. Data Flow Diagram may be partitioned into levels that represent increasing information flow and functional detail. The notation used to create a DFD is illustrated in the given as:

1. Process: A Process transforms incoming data flow(s) into outgoing data flow(s).
2. External Entity: An External Entity is a system element (hardware a person) that produces information for transformation by the software or received information produced by the software.
3. Data Store: A repository of data for the use of one or more process.

4. Arrow ( 🡪 ): A data item or collection of data items, the arrowhead indicates the direction of data flow.

Data Flow

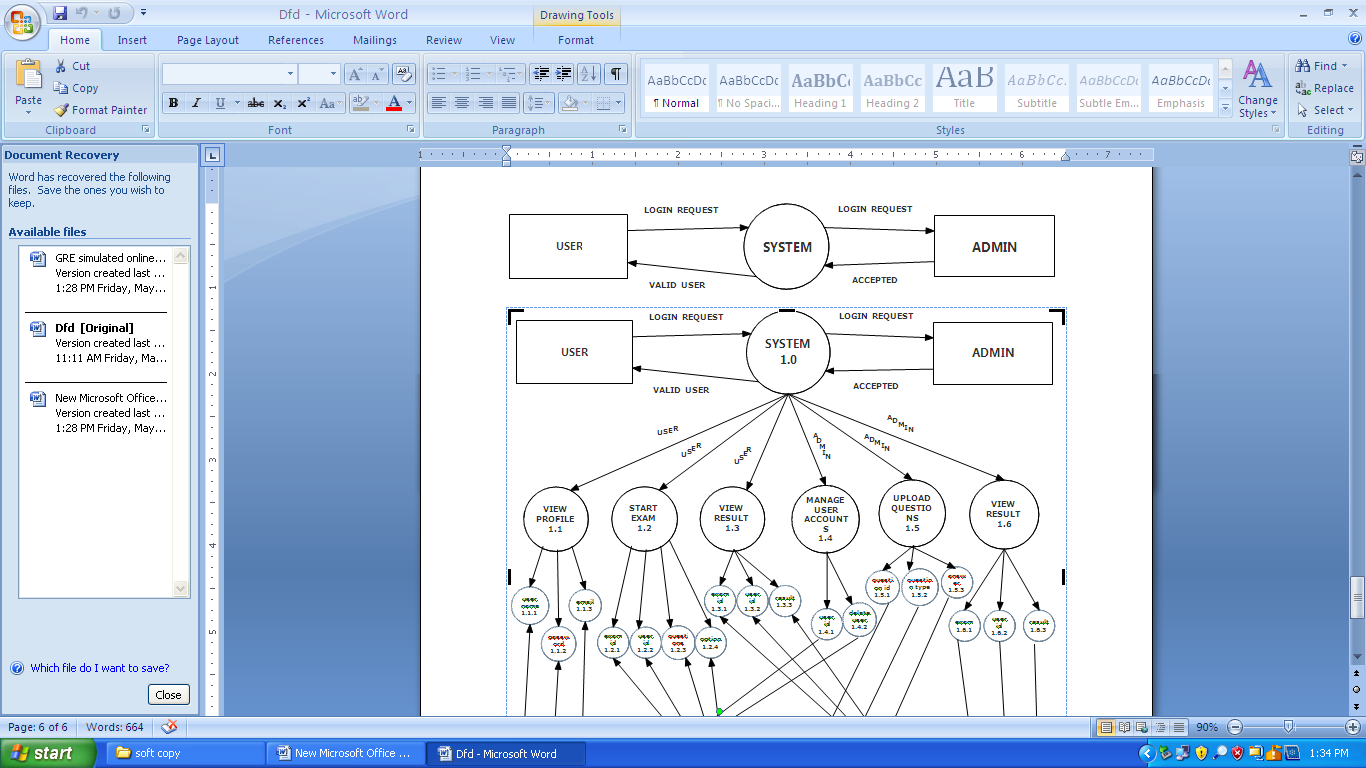
Processing Node

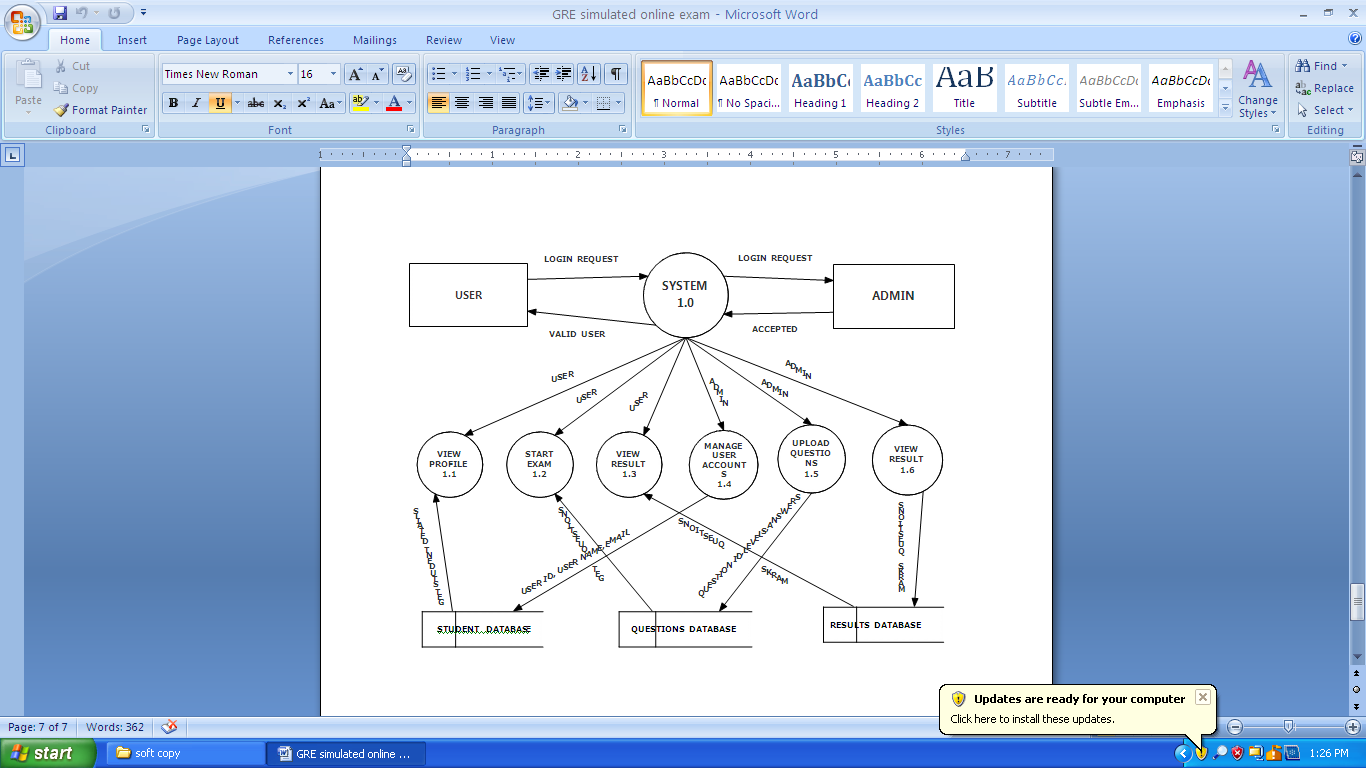
Source or Destination

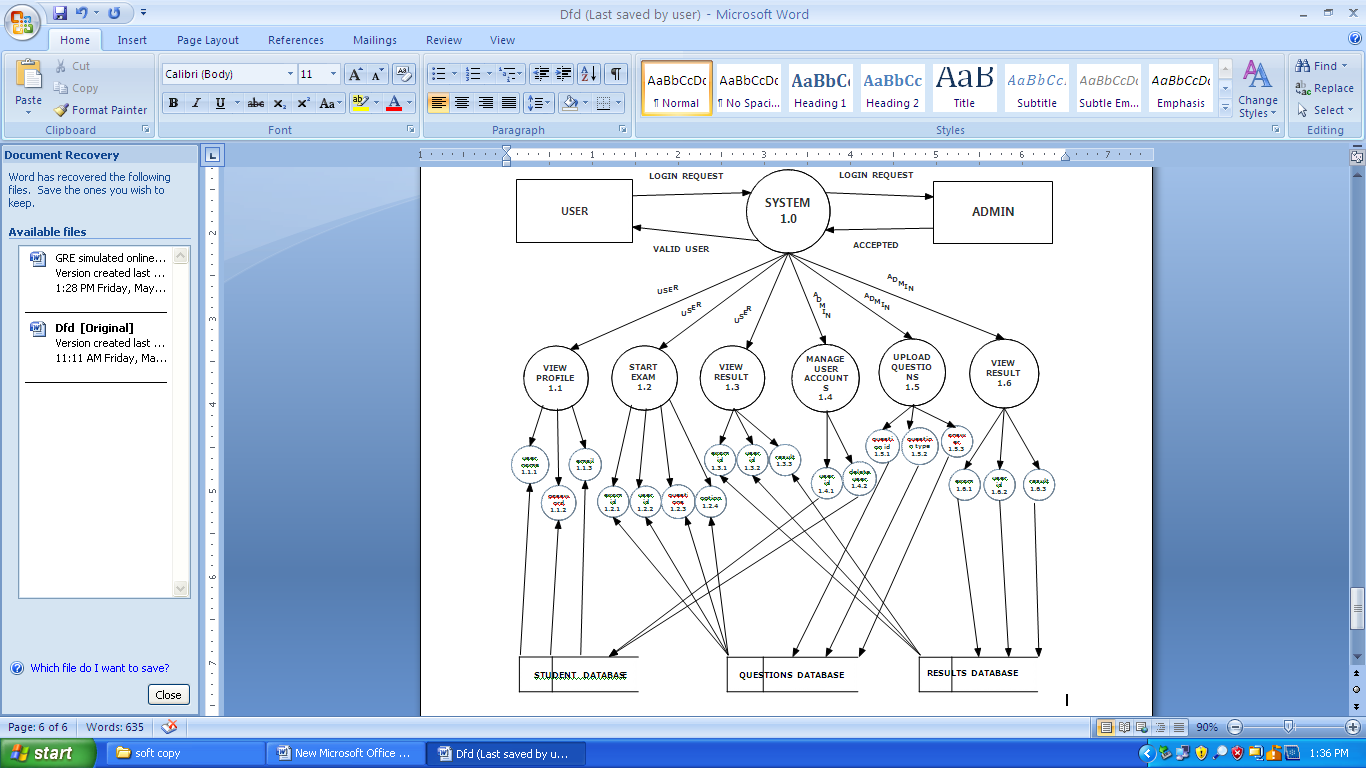
Data Store

**DATA FLOW DIAGRAMS**

**Context Level DFD/LEVEL 0:**

****

**Top Level:**

**Detailed level:****

**CHAPTER-04**

**SYSTEM DESIGN**

**UML DATA**

The UML’s four structural diagram exists to visualize, specify, construct and document the static aspects of system.Just as the static aspects of a house encompass the existence and placement of such things as walls, doors, windows, pipes, wires and vents. So to do the static aspects of a software system encompass the existence and placement of such things classes, interfaces, collaborations, components and nodes.

The UML’s structural diagrams are roughly organized around the major groups of things you will find when modeling a system.

1. **CLASS DIAGRAM:** classes, interfaces and collaborations
2. **OBJECT DIAGRAM:** objects
3. **COMPONENT DIAGRAM:** components
4. **DEPLOYMENT DIAGRAM:** nodes
5. **CLASS DIAGRAM**:

A class diagram shows a set of classes, interfaces, collaborations and their relationships. Class diagrams are the most common diagrams found in modeling object – oriented system. Class diagram that includes active classes are used to address the static process view of a system.

1. **COMPONENT DIAGRAM**:

A component diagram shows a set of components and their relationships. Component diagrams are related to class diagrams in that component typically maps to one or more classes, interfaces or collaborations. It is used to illustrate the static implementation view of a system.

1. **DEPLOYMENT DIAGRAM**:

A deployment diagrams shows a set of nodes and their relationships deployment diagrams are related to component diagrams in that node typically encloses one or more components.

**USECASE DIAGRAM**:

A use case diagram shows as set of use cases and actors and their relationships. Use case diagrams are especially important in organizing and modeling behavior of a system.

**SEQUENCE DIAGRAM**:

A sequence diagram is an interaction diagram that emphasizes the time ordering of messages. A sequence diagram shows a set of objects and messages sent and receive by those objects. The objects are typically named or anonymous instances of other things, such as collaborations, components and nodes. We can use sequence diagrams to illustrate the dynamic view of a system.

**COLLABORATION DIAGRAM**:

A collaboration diagram is an inter action diagram that emphasizes the structural organization of the objects that send and receive messages. It shows a set of objects, links among those objects and messages sent and receive by the objects. The objects are typically named or anonymous instances of classes but may also represent instance of other things such as collaborations, components and nodes.

**RELATIONSHIPS**

**1. Dependency**:

Dependency is a semantic relationship between two things in which change to one thing may affect the semantics of other things.

Graphically a dependency rendered as a dashed-line, possibly directed occasionally including a label.



Fig: Dependency

**2. Association**:

Association is a structural relationship that describes set of links.

a link being a connection among objects. Aggregation is a special kind of association a

structural relationship between a whole and its parts.

It is graphically rendered as a solid-line.Such as multiplicity and role names.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Employer employee

Fig: Association

**3. Generalization**:

Generalization is a specialization/generalization relationship in which objects of specialized element (child) are substitutable for object of the generalized element (parent).in this way the child shares the structure and behavior of the parent.

Graphically rendered as a solid-line with a hollow over-head pointing to the parent

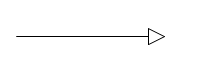


Fig: Generalization

**4. Realization**:

A realization is a semantic relationship between classifiers. Where in one classifier specifies a contract that another classifier guarantees to carry-out you will encounter relationships in two places between interfaces and classes or components between use case collaborations.

Graphically rendered as cross between generalization and dependency relationships

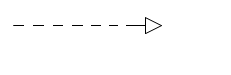


Fig: Realization

**5. Class**:

A class is a description of set of objects and the same attributes, operations relationships and semantics. A class implements one or more interfaces.

Graphically rendered as a rectangle, usually including a name, attributes and operations

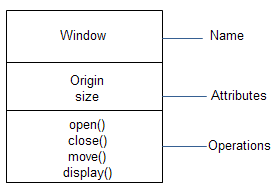


Fig: Classes

**6. Use case**:

Use case is a description of set of sequence of actions that a system performs that yields

and observable result of value to a particular actor.

Graphically rendered as an ellipse with solid-lines, usually including only its names

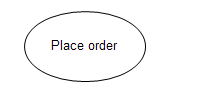


Fig: Use case

1. **Collaborations**:

Collaboration defines an interaction is a society of roles and other elements that

work together to provide some co-operative behavior that’s bigger than the some of elements.

Therefore collaborations have structural as well as behavioral dimensions.

Graphically rendered as an ellipse with dashed-lines usually including only its names

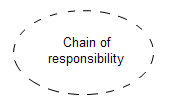


Fig: collaboration

1. **Component**:

A component is a physical and replicable part of a system that confirms to provide the relation of set of interfaces. In a system we can find and encounter different kinds of deployment components such as com+ components and java-beans.

A component typically represents the physical package logical elements such as classes, interfaces and collaborations.

It is graphically rendered as rectangle with tabs usually including only it’s names.

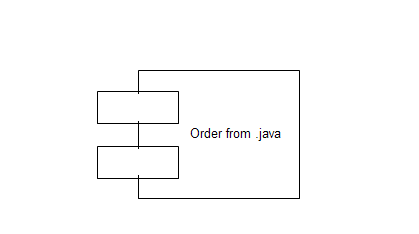


Fig: Component

1. **Node**:

A node is a physical element that exists at run-time and represents a computational resources generally having at least some memory and often processing capability.

Graphically rendered as a cube, usually including only its names

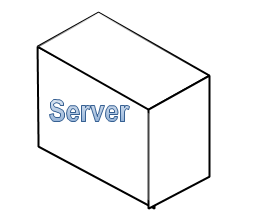


Fig: Node

**UML DIAGRAMS**

**Use case diagram for online fraud Auctions:**

**Use case analysis:**

The use case diagram is used to identify the primary element and processes. That form the system the primary element are termed as “actors” and the processes are called “use cases”.

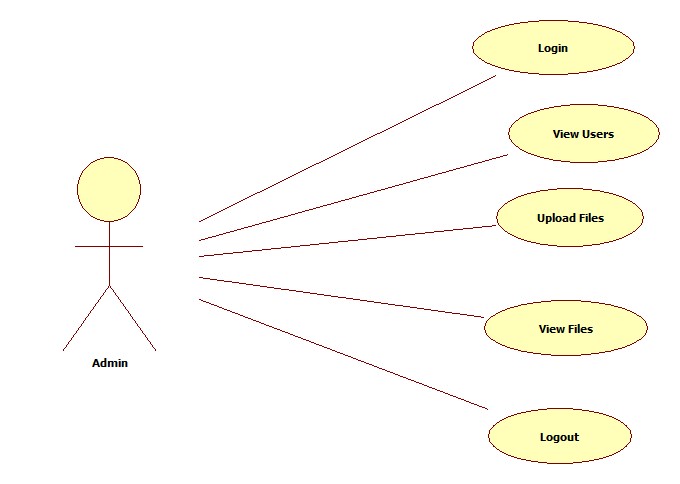
The use case diagram show which actors interacts with each use case.

Use case diagram of UML are used to view system from this perspective as a set of discrete Activities or transactions.

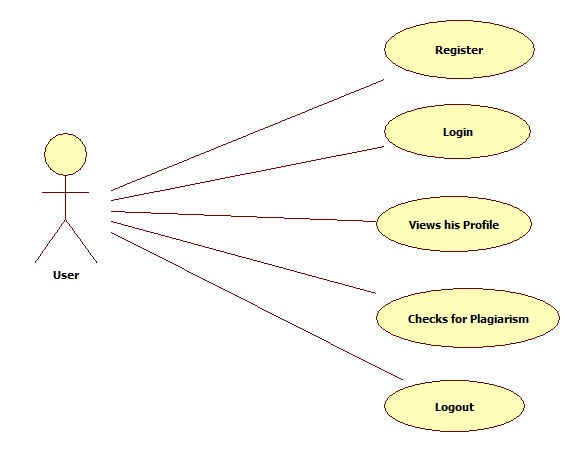
A ellipse represents a use case and a “Stick figure” represents an actor operating with in the use case.

* An actor can be user, system or other entity.
* The initiating actor appears on the left of the use case and the receiving actor appears on the right.
* The actor is identified by a name below the “stick figure”
* The name of the use case appears either in the side or below the ellipse.
* An association line is a that connects an actor and to the use case and represents communication between the actor and use cases
* The association line is solid similar to the line that connect associated classes.

**Admin Use Case Diagram:**

****

**User Use Case Diagram:**

****

**Class diagram:**



**Activity diagram:**

The process flows in the system are captured in the activity diagram similar to a state diagram an activity diagram also consists of activities, action, transitions

* Initial and final states and guard conditions

Example: from ordering to delivery and payment activity diagram describe the work slow behavior of a system.

# 

# Sequence diagram:

A sequence diagram is an interaction diagram that details how operations are carried out. What message are sent and when sequence diagram are organized according to time it represents the interaction between different objects in the system.

The important aspects of a sequence diagram are that it is time ordered.

This means that the exact sequence of the interaction between the object is represented step by step different objects in the sequence diagram interact with each other by passing message.



**COLLABORATION DIAGRAM**:

A collaboration diagram is an inter action diagram that emphasizes the structural organization of the objects that send and receive messages. It shows a set of objects, links among those objects and messages sent and receive by the objects. The objects are typically named or anonymous instances of classes but may also represent instance of other things such as collaborations, components and nodes.



**CHAPTER-05**

**System Testing and Implementation:**

**Testing Concept**

* **Testing**
* **Testing Methodologies**
* Black box Testing:
* White box Testing.
* Gray Box Testing.
* **Levels of Testing**
  + Unit Testing.
  + Module Testing.
  + Integration Testing.
  + System Testing.
  + User Acceptance Testing.
* **Types of Testing**
  + Smoke Testing.
  + Sanitary Testing.
  + Regression Testing.
  + Re-Testing.
  + Static Testing.
  + Dynamic Testing.
  + Alpha-Testing.
  + Beta-Testing.
  + Monkey Testing.
  + Compatibility Testing.
  + Installation Testing.
  + Adhoc Testing.
  + Ext….

**TCD (Test Case Documentation)**

* **STLC**
  + Test Planning.
  + Test Development.
  + Test Execution.
  + Result Analysis.
  + Bug-Tracing.
  + Reporting.
* **Microsoft Windows – Standards**
* **Manual Testing**
* **Automation Testing (Tools)**
  + Win Runner.
  + Test Director.

**Testing:**

* The process of executing a system with the intent of finding an error.
* Testing is defined as the process in which defects are identified, isolated, subjected for rectification and ensured that product is defect free in order to produce the quality product and hence customer satisfaction.
* Quality is defined as justification of the requirements
* Defect is nothing but deviation from the requirements
* Defect is nothing but bug.
* Testing --- The presence of bugs
* Testing can demonstrate the presence of bugs, but not their absence
* Debugging and Testing are not the same thing!
* Testing is a systematic attempt to break a program or the AUT
* Debugging is the art or method of uncovering why the script /program did not execute properly.

**Testing Methodologies:**

* Black box Testing: is the testing process in which tester can perform testing on an application without having any internal structural knowledge of application.

Usually Test Engineers are involved in the black box testing.

* White box Testing: is the testing process in which tester can perform testing on an application with having internal structural knowledge.

Usually the Developers are involved in white box testing.

* Gray Box Testing: is the process in which the combination of black box and white box techniques are used.

**Levels of Testing:**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | *Module1Module2Module3*   |  |  |  |  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | *Units*   |  |  |  | | --- | --- | --- | |  |  |  | |  | *Units*   |  |  |  | | --- | --- | --- | |  |  |  | |  | *Units*   |  |  |  | | --- | --- | --- | |  |  |  | |   I/p *Integration*  o/p I/p *Integration o/p* |   *System Testing: Presentation + business +Databases*  *🚹UAT: user acceptance testing* |

##### **(SOFTWARE TESTING LIFE CYCLE):**

**Test Planning**:

1. How to perform various testing on the total application in the most efficient Test Plan is defined as a strategic document which describes the procedure way.

2. This document involves the scope of testing,

3. Objective of testing,

4. Areas that need to be tested,

5. Areas that should not be tested,

6. Scheduling Resource Planning,

7. Areas to be automated, various testing tools Used….

**Test Development**: 1. Test case Development (check list)

2. Test Procedure preparation. (Description of the test cases)

**Test Execution**:     1. Implementation of test cases, observing the result.

**Result Analysis**: 1. Expected value: is nothing but expected behavior of application.

2. Actual value: is nothing but actual behavior of the Application

**Bug Tracing**: Collect all the failed cases, prepare documents

**Reporting:** Prepare document (status of the application) SAS

**Types of Testing**:

🚺>Smoke Testing: is the process of initial testing in which tester looks for the availability of all the functionality of the application in order to perform detailed testing on them. (Main check is for available forms)

🚺>Sanity Testing: is a type of testing that is conducted on an application initially to check for the proper behavior of an application that is to check all the functionality are available before the detailed testing is conducted by on them.

🚺>Regression Testing: is one of the best and important testing. Regression testing is the process in which the functionality, which is already tested before, is once again tested whenever some new change is added in order to check whether the existing functionality remains same.

🚺>Re-Testing: is the process in which testing is performed on some functionality which is already tested before to make sure that the defects are reproducible and to rule out the environments issues if at all any defects are there.

🚺Static Testing: is the testing, which is performed on an application when it is not been executed.

Ex: GUI, Document Testing

🚺Dynamic Testing: is the testing which is performed on an application when it is being executed.ex: Functional testing.

🚺Alpha Testing: it is a type of user acceptance testing, which is conducted on an application when it is just before released to the customer.

🚺Beta-Testing: it is a type of UAT that is conducted on an application when it is released to the customer, when deployed in to the real time environment and being accessed by the real time users.

🚺Monkey Testing: is the process in which abnormal operations, beyond capacity operations are done on the application to check the stability of it in spite of the user’s abnormal behavior.

🚺Compatibility testing: it is the testing process in which usually the products are tested on the environments with different combinations of databases (application servers, browsers…etc) In order to check how far the product is compatible with all these environments platform combination.

🚺Installation Testing: it is the process of testing in which the tester try to install or try to deploy the module into the corresponding environment by following the guidelines produced in the deployment document and check whether the installation is successful or not.

🚺Adhoc Testing: Adhoc Testing is the process of testing in which unlike the formal testing where in test case document is used, with out that test case document testing can be done of an application, to cover that testing of the future which are not covered in that test case document. Also it is intended to perform GUI testing which may involve the cosmetic issues.

**TCD (Test Case Document):**

Test Case Document Contains

* Test Scope (or) Test objective
* Test Scenario
* Test Procedure
* Test case

This is the sample test case document for the Academic details of student project:

**Test scope**:

* Test coverage is provided for the screen “Academic status entry” form of a student module of university management system application.
* Areas of the application to be tested.

**Test Scenario:**

* When the office personals use this screen for the marks entry, calculate the status details, saving the information on student’s basis and quit the form.

**Test Procedure:**

* The procedure for testing this screen is planned in such a way that the data entry, status calculation functionality, saving and quitting operations are tested in terms of GUI testing, Positive testing, Negative testing using the corresponding GUI test cases, Positive test cases, Negative test cases respectively.

**Guidelines for Test Cases:**

1. GUI Test Cases:

* Total no of features that need to be check
* Look & Feel
* Look for Default values if at all any (date & Time, if at all any require)
* Look for spell check

**Example for GUI Test cases:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| T.no | Description | Expected value | Actual value | Result |
| 1 | Check for all the features in the screen | The screen must contain all the features |  |  |
| 2 | Check for the alignment of the objects as per the validations | The alignment should be in proper way |  |  |

1. **Positive Test Cases:**

* The positive flow of the functionality must be considered
* Valid inputs must be used for testing
* Must have the positive perception to verify whether the requirements are justified.
* **Example for Positive Test cases:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| T.C.No | Description | Expected value | Actual value | Result |
| 1 | Valid Login | It should accept | It should accept | It should accept |
| 2 | Upload a file if they want to download it generates a puzzle.If the user enters right answer then he can download. | It should accept | It should accept | It should accept |

1. **Negative Test Cases:**

* Must have negative perception.
* Invalid inputs must be used for test

.

**Example for Negative Test cases:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| T.C.No | Description | Expected value | Actual value | Result |
| 1 | Invalid Username or Password | Modification should not be allow | Modification should not be allow | It should show invalid crediationals |
| 2 | Upload a file if they want to download it generates a puzzle.If the user enters right answer then he can download. | It should not accept . | It should not accept . | Show invalid answer. |

**CHAPTER-06**

**SAMPLE CODE**

**Registration Code**

using System;

usingSystem.Collections.Generic;

usingSystem.Linq;

usingSystem.Web;

usingSystem.Web.UI;

usingSystem.Web.UI.WebControls;

usingSystem.Data;

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

{

RadioButtonrb=new RadioButton();

balobj = new bal();

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

{

}

protected void TextBox7\_TextChanged(object sender, EventArgs e)

{

}

protected void Button1\_Click(object sender, EventArgs e)

{

if(Radmale.Checked)

{

rb = Radmale;

}

else

{

rb = Radfemale;

}

string qry = "insert into Regvalues('" + txtUserName.Text + "','" + txtemailid.Text + "','" + txtpassword.Text + "','" + rb.Text + "'," + Txtmobileno.Text + ",'" + Txtaddress.Text + "','" + ddlsecurityquestion.SelectedItem.Text + "','" + txtsecurityanswer.Text + "')";

inti = obj.InsUpDel(qry);

if (i> 0)

{

Label9.Text = "Registered Succesfully";

}

else

{

Label9.Text = "Not Yet Registered";

}

}

}

**Login code**

using System;

usingSystem.Collections.Generic;

usingSystem.Linq;

usingSystem.Web;

usingSystem.Web.UI;

usingSystem.Web.UI.WebControls;

usingSystem.Data;

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

{

balobj = new bal();

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

{

}

protected void TextBox2\_TextChanged(object sender, EventArgs e)

{

}

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

{

if (DropDownList1.SelectedItem.Text == "USER")

{

string query = "select \* from Reg where emaiid='" + Txtusername.Text.Trim() + "' and password='" + Txtpassword.Text.Trim() + "'";

DataSet ds = obj.GetDetalis(query);

if (ds.Tables[0].Rows.Count> 0)

{

Session["Username"] = ds.Tables[0].Rows[0][1];

Response.Redirect("~/User/Default.aspx");

}

else

{

Response.Write("<script>alert('Invalid Username & Password')</script>");

}

}

else if(DropDownList1.SelectedItem.Text=="ADMIN")

{

if (Txtusername.Text == "admin" &&Txtpassword.Text == "admin")

{

Response.Redirect("Admin/Home.aspx");

}

}

}

}

**Home code**

using System;

usingSystem.Collections.Generic;

usingSystem.Linq;

usingSystem.Web;

usingSystem.Web.UI;

usingSystem.Web.UI.WebControls;

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

{

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

{

}

}

**Contact Us code**

using System;

usingSystem.Collections.Generic;

usingSystem.Linq;

usingSystem.Web;

usingSystem.Web.UI;

usingSystem.Web.UI.WebControls;

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

{

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

{

}

}

**User Game code**

usingSystem.Collections.Generic;

usingSystem.Linq;

usingSystem.Web;

usingSystem.Web.UI;

usingSystem.Web.UI.WebControls;

using System.IO;

public partial using System;

classUser\_GAme : System.Web.UI.Page

{

Random rd;

int q;

publicint bind6()

{

Random rd = new Random();

q = rd.Next(15);

return q;

}

publicint bind()

{

Random r = new Random();

inti = r.Next(30);

returni;

}

publicint bind1()

{

Random r1 = new Random();

int j = r1.Next(10);

return j;

}

publicint bind2()

{

Random r2 = new Random();

int m = r2.Next(30);

return m;

}

publicint bind3()

{

Random r3 = new Random();

int s = r3.Next(25);

return s;

}

public void main()

{

int k, l, z, w;

k = bind();

l = bind1();

z = bind2();

w = bind3();

int add = k + l - w \* z;

Session["add"] = add;

Label1.Text = k + "+" + l + "-" + w + "\*" + z + "=" + "?";

}

public void main1()

{

int k, l, z, w;

k = bind();

l = bind1();

z = bind2();

w = bind3();

int add = k \* l + w - z;

Session["add"] = add;

Label1.Text = k + "\*" + l + "+" + w + "-" + z + "=" + "?";

}

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

{

if (!IsPostBack)

{

int u=bind6();

if (u < 7)

{

main();

}

else

{

main1();

}

}

}

protected void Button1\_Click(object sender, EventArgs e)

{

inti = Convert.ToInt32(TextBox1.Text);

if (i == Convert.ToInt32(Session["add"]))

{

Response.ContentType = ContentType;

Response.AppendHeader("Content-Disposition", "attachment; filename=" + Path.GetFileName(Session["path"].ToString()));

Response.WriteFile(Session["path"].ToString());

Response.End();

}

else

{

Response.Write("<script>alert('Invalid Answer')</script>");

}

}

}

**User code**

using System;

usingSystem.Collections.Generic;

usingSystem.Linq;

usingSystem.Web;

usingSystem.Web.UI;

usingSystem.Web.UI.WebControls;

public partial class User\_User : System.Web.UI.MasterPage

{

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

{

}

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

{

Session.Abandon();

Response.Redirect("~/home.aspx");

}

}

**Download code**

using System;

usingSystem.Collections.Generic;

usingSystem.Linq;

usingSystem.Web;

usingSystem.Web.UI;

usingSystem.Web.UI.WebControls;

usingSystem.Data;

public partial class User\_Download : System.Web.UI.Page

{

balobj = new bal();

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

{

if (!IsPostBack)

{

string query = "select \* from tbl\_file";

DataSet ds = obj.GetDetalis(query);

if (ds.Tables[0].Rows.Count> 0)

{

gvFiles.DataSource = ds.Tables[0];

gvFiles.DataBind();

}

}

}

protected void btnclick(object sender, EventArgs e)

{

Button btn = (Button)sender;

GridViewRow gr = (GridViewRow)btn.NamingContainer;

Label lblFileId = (Label)gr.FindControl("lblFileId");

Label lblFileName = (Label)gr.FindControl("lblFileName");

Label lblDesc = (Label)gr.FindControl("lblDesc");

Label lblFilepath = (Label)gr.FindControl("lblFilepath");

Session["id"] = lblFileId.Text;

Session["filename"] = lblFileName.Text;

Session["filedes"] = lblDesc.Text;

Session["path"] = lblFilepath.Text;

Response.Redirect("GAme.aspx");

}

}

**Admin view user**

using System;

usingSystem.Collections.Generic;

usingSystem.Linq;

usingSystem.Web;

usingSystem.Web.UI;

usingSystem.Web.UI.WebControls;

usingSystem.Data;

public partial class Admin\_View\_Users : System.Web.UI.Page

{

balobj = new bal();

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

{

if (!IsPostBack)

{

bind();

}

}

public void bind()

{

stringqry = "select \* from Reg";

DataSet ds = new DataSet();

ds = obj.GetDetalis(qry);

if (ds.Tables[0].Rows.Count> 0)

{

GridView1.DataSource = ds;

GridView1.DataBind();

}

}

}

**Admin view upload**

using System;

usingSystem.Collections.Generic;

usingSystem.Linq;

usingSystem.Web;

usingSystem.Web.UI;

usingSystem.Web.UI.WebControls;

usingSystem.Data;

public partial class Admin\_View\_Upload : System.Web.UI.Page

{

balobj = new bal();

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

{

bind();

}

public void bind()

{

stringqry = "select \* from tbl\_file";

DataSet ds = new DataSet();

ds = obj.GetDetalis(qry);

GridView1.DataSource = ds;

GridView1.DataBind();

}

}

**CHAPTER-07**

**TESTING**

Testing Concepts

* **Testing**
* **Testing Methodologies**

* Black box Testing:
* White box Testing.
* Gray Box Testing.

* **Levels of Testing**
  + Unit Testing.
  + Module Testing.
  + Integration Testing.
  + System Testing.
  + User Acceptance Testing.
* **Types Of Testing**
  + Smoke Testing.
  + Sanitary Testing.
  + Regression Testing.
  + Re-Testing.
  + Static Testing.
  + Dynamic Testing.
  + Alpha-Testing.
  + Beta-Testing.
  + Monkey Testing.
  + Compatibility Testing.
  + Installation Testing.
  + Adhoc Testing.

**TCD (Test Case Documentation)**

* **STLC**
  + Test Planning.
  + Test Development.
  + Test Execution.
  + Result Analysis.
  + Bug-Tracing.
  + Reporting.
* **Microsoft Windows – Standards**
* **Manual Testing**
* **Automation Testing (Tools)**
  + Win Runner.
  + Test Director.

**Testing:**

* The process of executing a system with the intent of finding an error.
* Testing is defined as the process in which defects are identified, isolated, subjected for rectification and ensured that product is defect free in order to produce the quality product and hence customer satisfaction.
* Quality is defined as justification of the requirements
* Defect is nothing but deviation from the requirements
* Defect is nothing but bug.
* Testing --- The presence of bugs
* Testing can demonstrate the presence of bugs, but not their absence
* Debugging and Testing are not the same thing!
* Testing is a systematic attempt to break a program or the AUT
* Debugging is the art or method of uncovering why the script /program did not execute properly.

**Testing Methodologies:**

* **Black box Testing**: is the testing process in which tester can perform testing on an application without having any internal structural knowledge of application.

Usually Test Engineers are involved in the black box testing.

* **White box Testing**: is the testing process in which tester can perform testing on an application with having internal structural knowledge.

Usually The Developers are involved in white box testing.

* **Gray Box Testing**: is the process in which the combination of black box and white box tonics are used.

**Levels of Testing:**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | ***Module1Module2Module3***   |  |  |  |  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | ***Units***   |  |  |  | | --- | --- | --- | |  |  |  | |  | ***Units***   |  |  |  | | --- | --- | --- | |  |  |  | |  | ***Units***   |  |  |  | | --- | --- | --- | |  |  |  | |   i/p ***Integration***  o/p i/p ***Integration o/p*** |   ***System Testing: Presentation + business +Databases***  ***🚹UAT: user acceptance testing*** |

##### **STLC (SOFTWARE TESTING LIFE CYCLE**)

**Test Planning**

**1.** Test Plan is defined as a strategic document which describes the procedure how to perform various testing on the total application in the most efficient way.

**2.** This document involves the scope of testing,

**3.** Objective of testing,

**4.** Areas that need to be tested,

**5.** Areas that should not be tested,

**6.** Scheduling Resource Planning,

**7.** Areas to be automated, various testing tools

Used….

**Test Development**:

**1.** Test case Development (check list)

**2.** Test Procedure preparation. (Description of the Test cases)

**Test Execution**: **1.** Implementation of test cases. Observing the result

**Result Analysis**: **1.** Expected value: is nothing but expected behavior of application.

1. Actual value: is nothing but actual behavior of Application

**Bug Tracing:** Collect all the failed cases, prepare documents.

**Reporting:** Prepare document (status of the application)

**Types Of Testing:**

**Smoke Testing**: is the process of initial testing in which tester looks for the availability of all the functionality of the application in order to perform detailed testing on them. (Main check is for available forms)

**Sanity Testing:** is a type of testing that is conducted on an application initially to check for the proper behavior of an application that is to check all the functionality are available before the detailed testing is conducted by on them.

**Regression Testing:** is one of the best and important testing. Regression testing is the process in which the functionality, which is already tested before, is once again tested whenever some new change is added in order to check whether the existing functionality remains same.

**Re-Testing:** is the process in which testing is performed on some functionality which is already tested before to make sure that the defects are reproducible and to rule out the environments issues if at all any defects are there.

**Static Testing:** is the testing, which is performed on an application when it is not been executed.ex: GUI, Document Testing

**Dynamic Testing:** is the testing which is performed on an application when it is being executed.ex: Functional testing.

**Alpha Testing:** it is a type of user acceptance testing, which is conducted on an application when it is just before released to the customer.

**Beta-Testing:** it is a type of UAT that is conducted on an application when it is released to the customer, when deployed in to the real time environment and being accessed by the real time users.

**Monkey Testing:** is the process in which abnormal operations, beyond capacity operations are done on the application to check the stability of it in spite of the user’s abnormal behavior.

**Compatibility testing:**it is the testing process in which usually the products are tested on the environments with different combinations of databases (application servers, browsers…etc) In order to check how far the product is compatible with all these environments platform combination.

**Installation Testing:** it is the process of testing in which the tester try to install or try to deploy the module into the corresponding environment by following the guidelines produced in the deployment document and check whether the installation is successful or not.

**Adhoc Testing:** Adhoc Testing is the process of testing in which unlike the formal testing where in test case document is used, with out that test case document testing can be done of an application, to cover that testing of the future which are not covered in that test case document. Also it is intended to perform GUI testing which may involve the cosmetic issues.

**TCD (Test Case Document:**

**Test Case Document Contains**

* **Test Scope (or) Test objective**
* **Test Scenario**
* **Test Procedure**
* **Test case**

This is the sample test case document for the Academic details of student project:

**Test scope:**

* Test coverage is provided for the screen “ Academic status entry” form of a student module of university management system application
* Areas of the application to be tested.

**Test Scenario:**

* When the office personals use this screen for the marks entry, calculate the status details, saving the information on student’s basis and quit the form.

**Test Procedure:**

* The procedure for testing this screen is planned in such a way that the data entry, status calculation functionality, saving and quitting operations are tested in terms of Gui testing, Positive testing, Negative testing using the corresponding Gui test cases, Positive test cases, Negative test cases respectively

**Test Cases:**

* Template for Test Case:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| T.C.No | Description | Exp | Act | Result |
|  |  |  |  |  |

**Guidelines for Test Cases**:

**1. GUI Test Cases:**

* Total no of features that need to be check
* Look & Feel
* Look for Default values if at all any (date & Time, if at all any require)
* Look for spell check

**Example for GUI Test cases**:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| T.C.No | Description | Expected value | Actual value | Result |
| 1 | Check for all the features in the screen | The screen must contain all the features |  |  |
| 2 | Check for the alignment of the objects as per the validations | The alignment should be in proper way |  |  |

1. **Positive Test Cases:**

* The positive flow of the functionality must be considered
* Valid inputs must be used for testing
* Must have the positive perception to verify whether the requirements are justified.

**Example for Positive Test cases:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| T.C.No | Description | Expected value | Actual value | Result |
| 1 | Valid Login | It should accept | It should accept | It should accept |
| 2 | Upload a file if they want to download it generates a puzzle.If the user enters right answer then he can download. | It should accept | It should accept | It should accept |

1. **Negative Test Cases:**

* Must have negative perception.
* Invalid inputs must be used for test.

**Example for Negative Test cases**:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| T.C.No | Description | Expected value | Actual value | Result |
| 1 | Invalid Username or Password | Modification should not be allow | Modification should not be allow | It should show invalid crediationals |
| 2 | Upload a file if they want to download it generates a puzzle.If the user enters right answer then he can download. | It should not accept . | It should not accept . | Show invalid answer. |

**CHAPTER-08**

**OUTPUTS**

**Output Screen 8.1:**

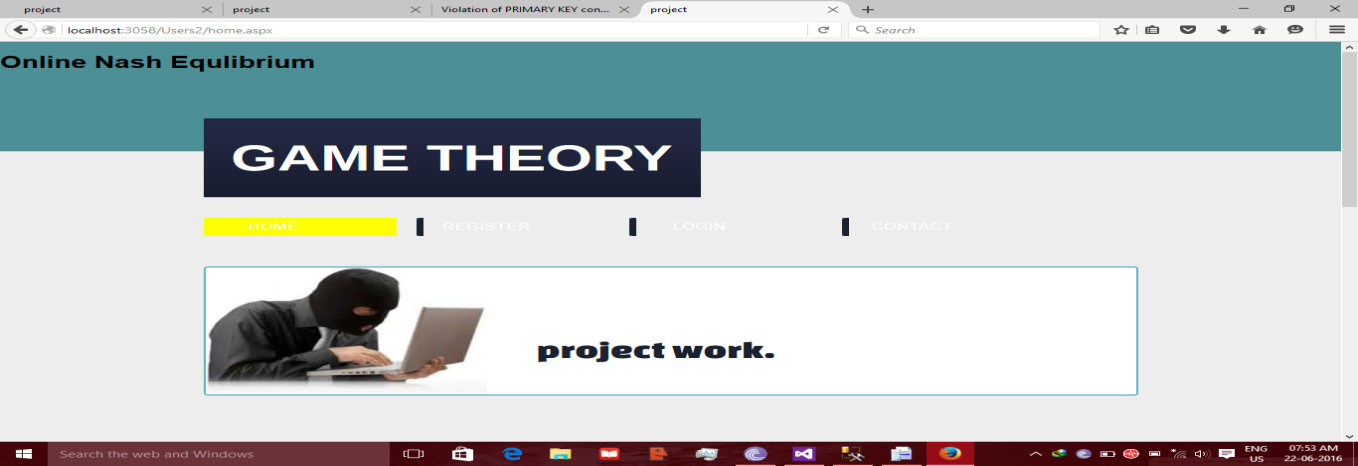
****

Fig 8.1: Home page

**Output Screen 8.2:**

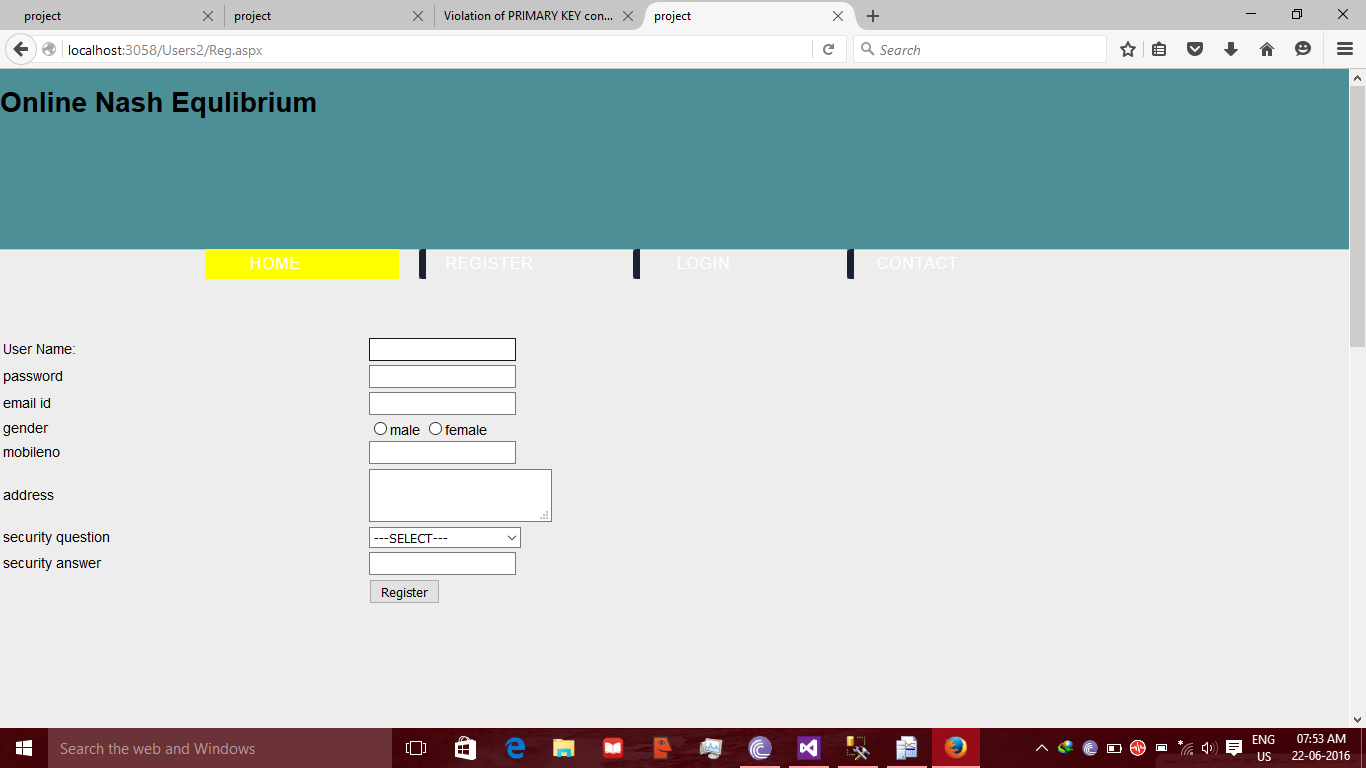


Fig 8.2: Registration Page

**Output Screen 8.3:**

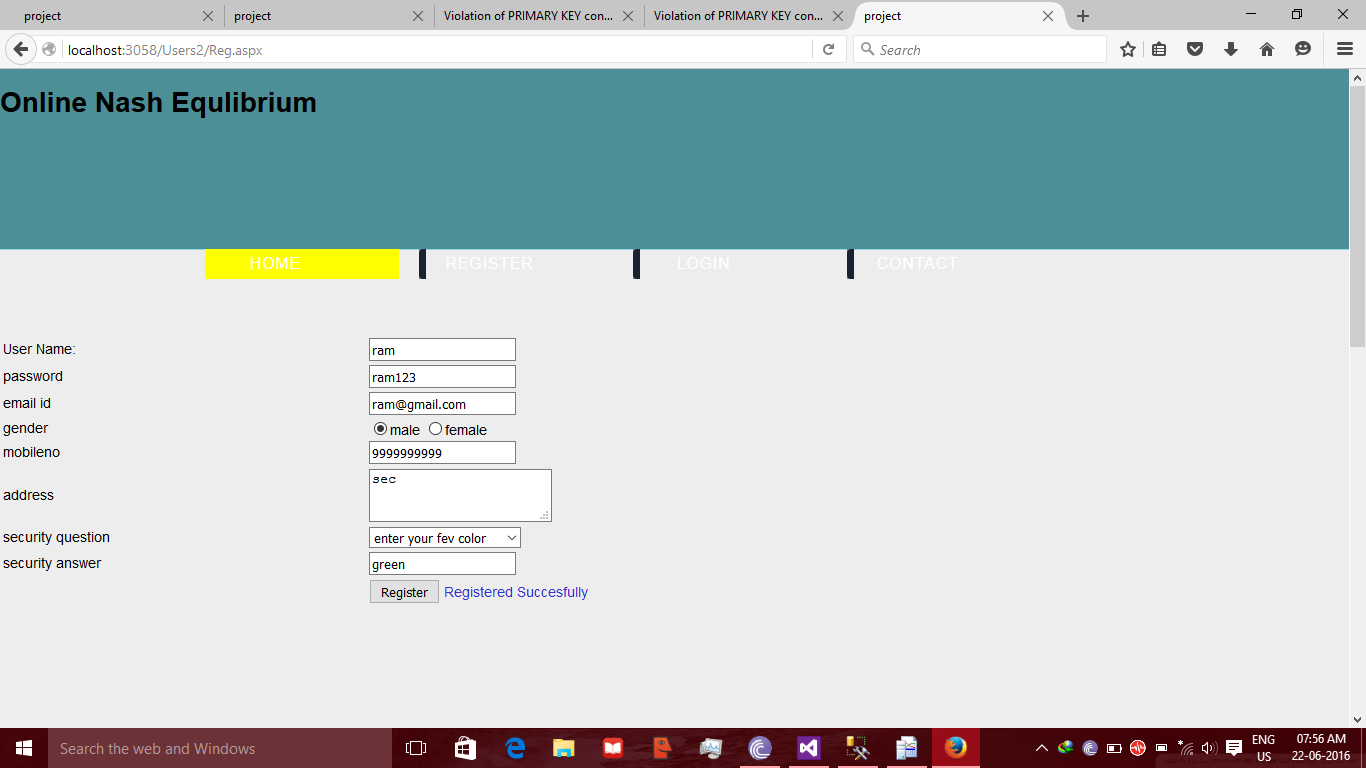
****

Fig 8.3: Registered Successfully Page

**Output Screen 8.4:**

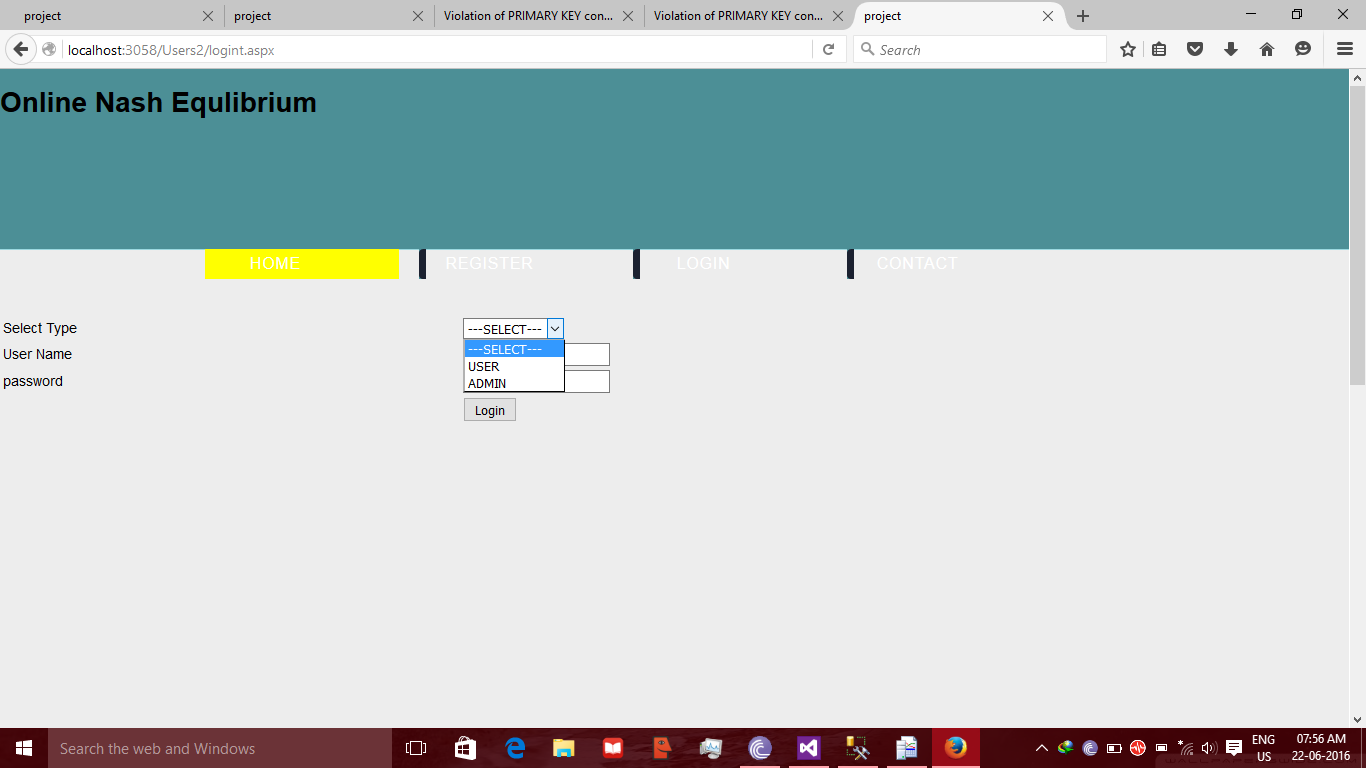


Fig 8.4: Login Page

**Output Screen 8.5:**

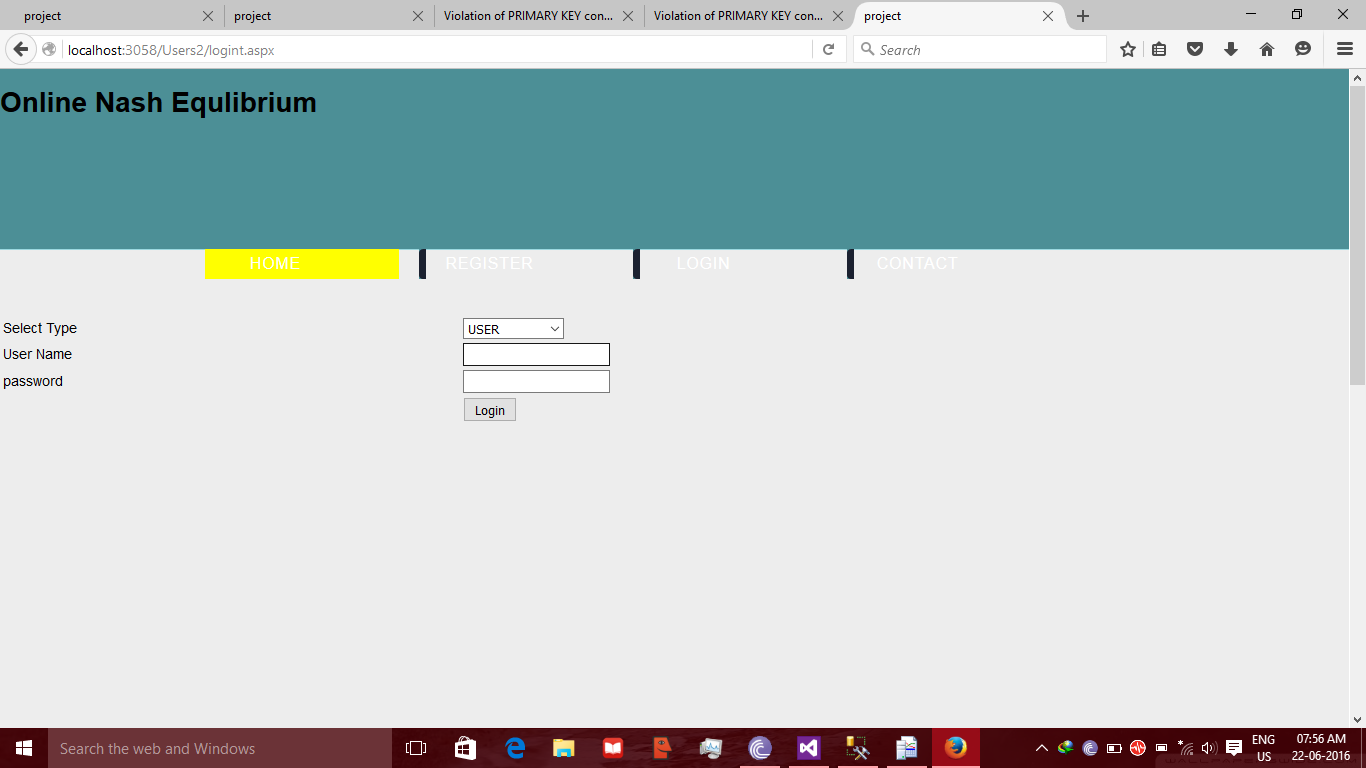


Fig 8.5: User Login Page

**Output Screen 8.6:**

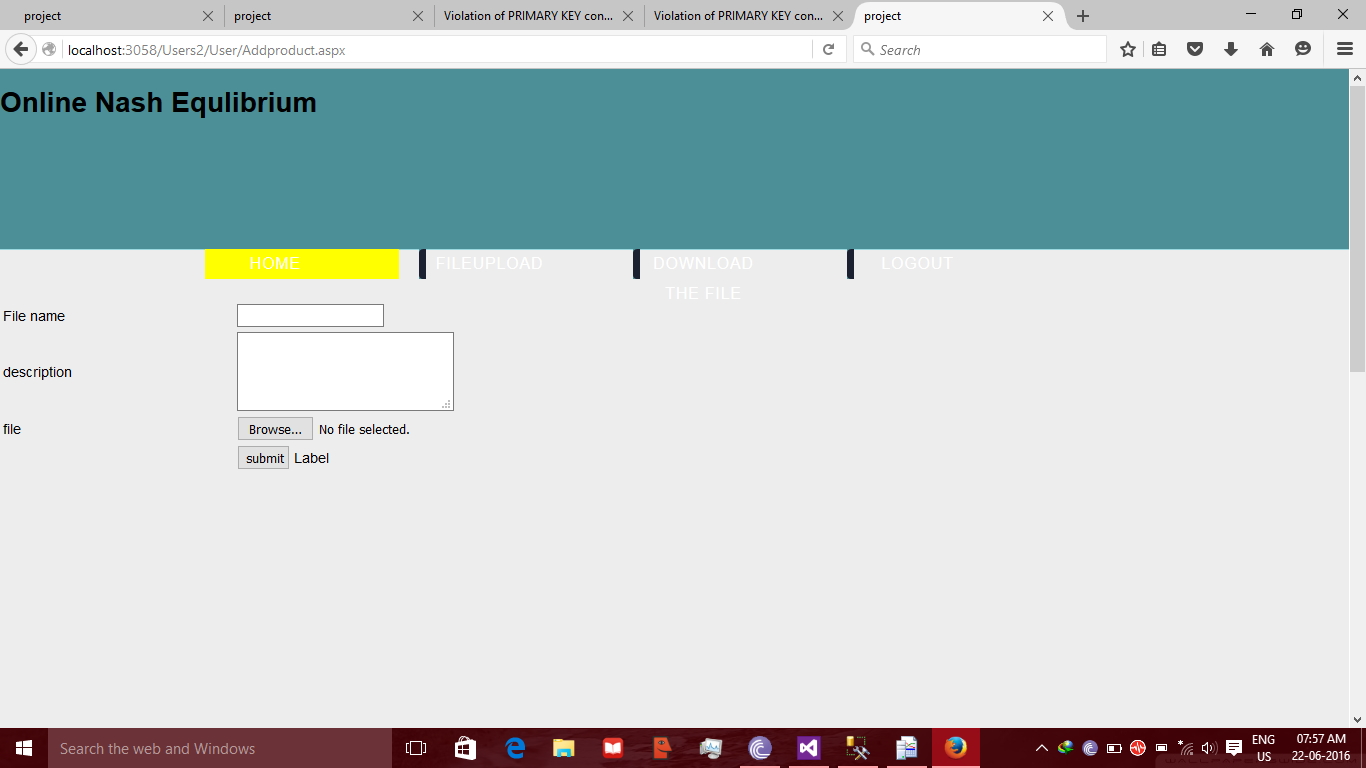


Fig 8.6: User Fileupload Page

**Output Screen 8.7:**

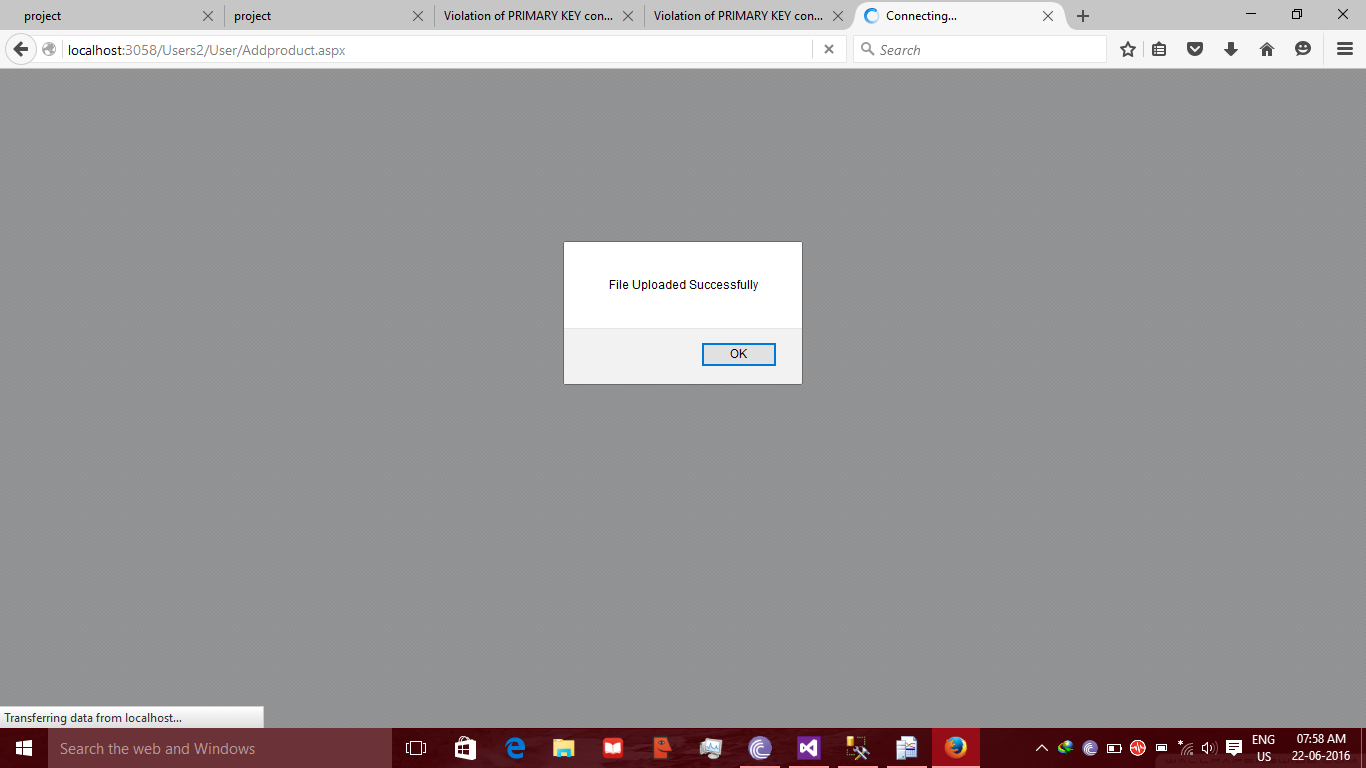


Fig 8.7: Fileupload Successfully Page

**Output Screen 8.8:**

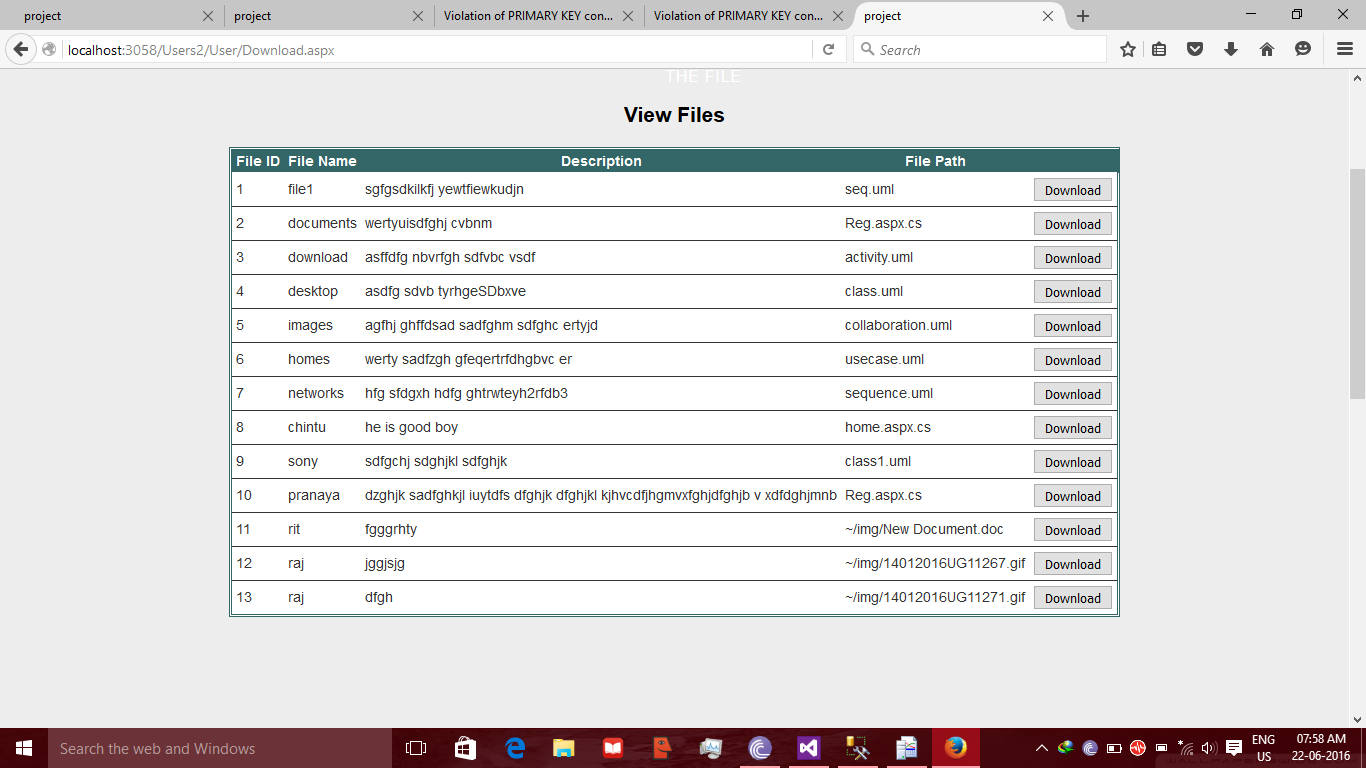


Fig 8.8: User View Files Page

**Output Screen 8.9:**

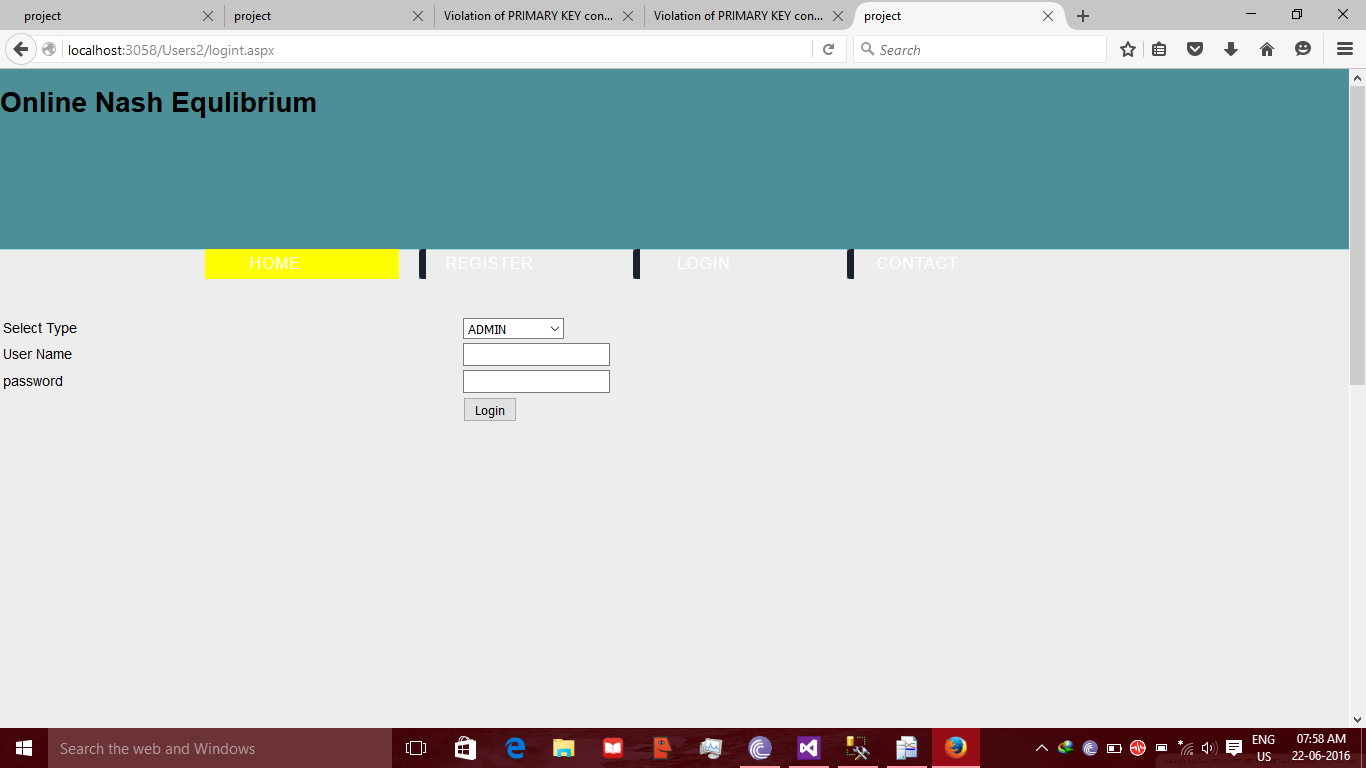


Fig 8.9: Admin Login Page

**Output Screen 8.10:**

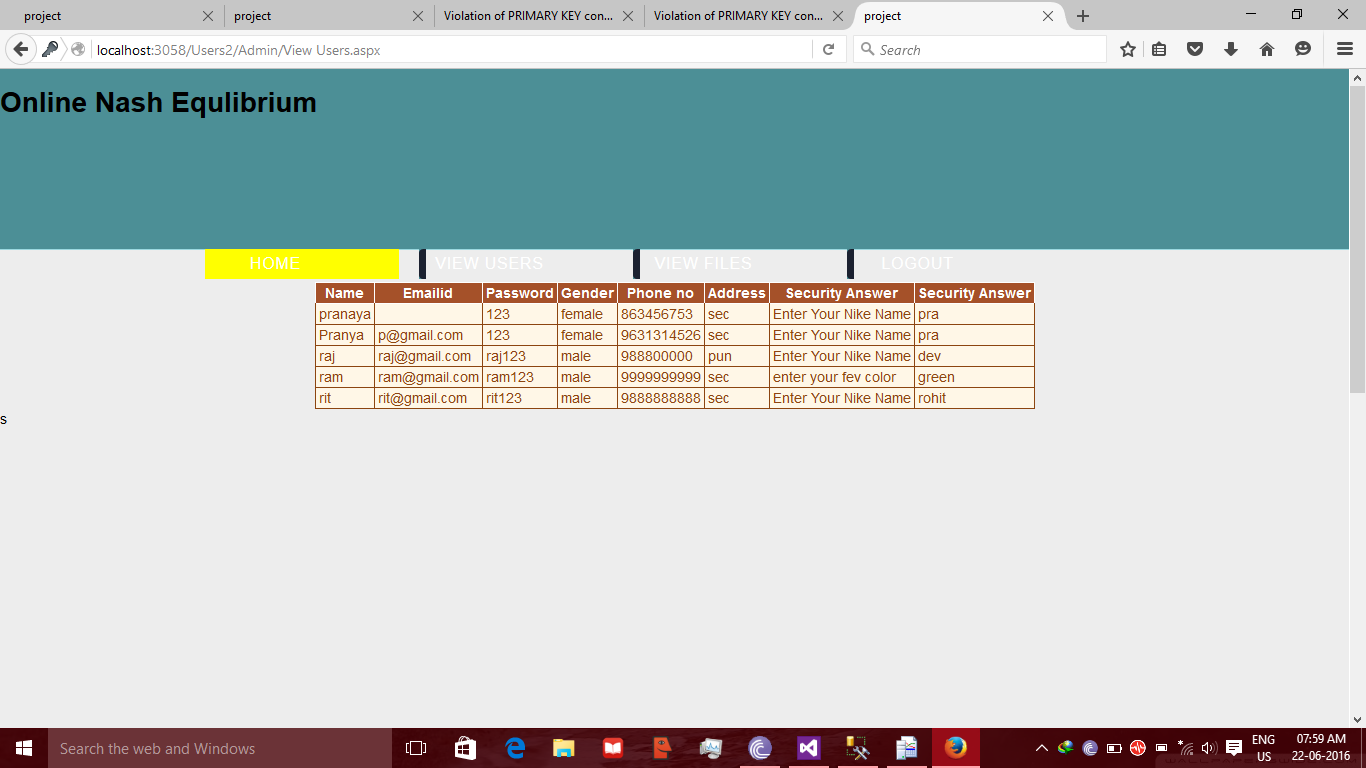


Fig 8.9: Admin View Files Page

**Output Screen 8.11:**

****

Fig 8.11: Admin View Profile Page

**CHAPTER-09**

**CONCLUSION**

Game theory has been used in this paper to provide defence mechanisms for flooding attacks using puzzles. The interaction between the defender and attacker is considered as an infinitely repeated game of discounted payoffs. The mechanism has been divided into different levels. The present problems of optimality and effectiveness have been solved by this mechanism. It also provides scalability and can be deployed in various environments with requirement of different security levels. Hence by use of game theory we can provide ultimate defence mechanism for flooding attacks.

**CHAPTER-10**

**FUTURE SCOPE**

In this project we have been done for the college level. In future we will be going to do for the university level and in the checking module also put more constraints because there is number of technical keywords which might be matched and also we are plan for connect our article comparer with www web server, if it happens then for admin no need to feed data in the databases it will redirect with the www web server so, it would be check the user request with globally**.**

**CHAPTER-11**

**BIBILIOGRAPHY**

**Books for Reference:**

[1]C. Anley. Advanced sql injection in sql server applications.Technicalreport, Next Generation Security Software, Ltd, 2002.

[2]K. Bai, H. Wang, and P. Liu. Towards database ﬁrewalls.In DBSec2005.

[3]B. I. A. Barry and H. A. Chan. Syntax, and semantics-based signaturedatabase for hybrid intrusion detection systems.Security and CommunicationNetworks, 2(6), 2009.

[4]D. Bates, A. Barth, and C. Jackson. Regular expressions considered harmful in client-side xss ﬁlters. In Proceedings of the 19th international conference on World wide web, 2010.

[5]M. Christodorescu and S. Jha. Static analysis of executables to detect malicious patterns.[6]M. Cova, D. Balzarotti, V. Felmetsger, and G. Vigna. Swaddler: An Approach for the Anomaly-based Detection of State Violations in Web Applications. I