**** ******** ****

**INDEX**

1. INTRODUCTION
2. LITRUTURE SURVEY

2.1 Introduction

2.2 Existing System

2.3 proposed system

**3. REQUIREMENT SPECIFICATION**

3.1 Introduction

3.2 Requirement Specification

3.3 Functional requirement

**4. DESIGN**

4.1 UML Diagrams

**5. IMPLEMENTATION**

**6. TESTING**

6.1 Test plan

6.2 Test cases

**7. OUTPUTS**

**8. CONCLUSION**

**9. SCOPE AND FUTURE ENHANCEMENT**

**10. BIBILIOGRAPHY**

**LIST OF FIGURES**

**Figure Name of the figure**

1. Admin Activity Diagram
2. User Activity Diagram
3. Admin Use case Diagram
4. User Use case Diagram

**5 Class Diagram**

**6 Collaboration Diagram**

**7 Sequence Diagram**

**LIST OF OUT PUT SCREENS**

**1. SCREEN SHOT 1**

**2. SCREEN SHOT 2**

**3. SCREEN SHOT 3**

**4. SCREEN SHOT 4**

**5. SCREEN SHOT 5**

**6. SCREEN SHOT 6**

**7. SCREEN SHOT 7**

**8. SCREEN SHOT 8**

**9. SCREEN SHOT 9**

**10. SCREEN SHOT 10**

**11. SCREEN SHOT 11**

**12. SCREEN SHOT 12**

**13.SCREEN SHOT 13**

**ABSTRACT**

The web data has exploded to a considerable amount with the invention of Internet Technology. The web being very vast covering billions of web sites has been monitored by a tool or program called ‘crawler’. Web crawler is a computer program that browsers the World Wide Web in a Methodical, automated manner or in an orderly fashion. Web crawling is an important method for collecting data on and keeping on with, the rapidly expanding internet. A vast number of Web pages are continuously being added everyday, and information is constantly changing .The Behavioral pattern of the web crawler based on these policies is also taken for the study. We will take one seed URL as input and search with a keyword, the searching result is based on keyword, and it will fetch the web pages where it will find the keyword. This focused based crawler approach retrieve documents that contain particular keyword from the users query, we are implementing this using breadth-first search. Now, when we retrieved the web pages we will apply pattern recognition over text. The evaluation of these Web crawler from basic general purpose Web crawler to the latest adaptive web crawler is studied. Crawlers can be used to gather specific types of information from Web pages, such as e-mail addresses.

**CHAPTER-1**

**INTRODUCTION**

We present FOCUS (Forum Crawler Under Supervision), a supervised web-scale forum crawler. The Web contains the huge amount of data and it contains the different websites to search. It is difficult to search the required content so we are implement the web crawler. The goal of FOCUS is to only crawl relevant forum content from the web with minimal overhead. Forum threads contain information content that is the target of forum crawlers. Although forums have different layouts or styles and are powered by different forum software packages, they always have similar implicit navigation paths connected by specific URL types to lead users from entry pages to thread pages. The paper also gives the overview of web crawling and web forums.

In the Existing system, Vidal et al proposed a method for learning regular expression patterns of URLs that lead a crawler from an entry page to target pages. Target pages were found through comparing DOM trees of pages with a preselected sample target page. It is very effective but it only works for the specific site from which the sample page is drawn. The same process has to be repeated every time for a new site. Therefore, it is not suitable to large- scale crawling.

We propose a new system for web crawl as FOCUS: Learning to Crawl Web Forums.It is a system overcome by existing crawl systems. In this method for learning regular expression patterns of URLs that lead a crawler from an entry page to target pages. Target pages were found through comparing DOM trees of pages with a pre-selected sample target page. It is very effective but it only works for the specific site from which the sample page is drawn. The same process has to be repeated every time for a new site. Therefore, it is not suitable to large- scale crawling. In contrast, FOCUS learns URL patterns across multiple sites and automatically finds forum entry page given a page from a forum. Experimental results show that FOCUS is effective in large scale forum crawling by leveraging crawling knowledge learned from a few annotated forum sites. A recent and more comprehensive work on forum crawling is iRobot. iRobot aims to automatically learn a forum crawler with minimum human intervention by sampling forum pages, clustering them, selecting informative clusters via an in formativeness measure, and finding a traversal path by a spanning tree algorithm. However, the traversal path selection procedure requires human inspection. The technology used in this title is web crawl technology for dealing with software and hardware requirements to accelerate iterative reconstruction

**About Project:**

INTERNET forums (also called web forums) are important services where users can request and exchange information with others. Proposed methods to extract and rank product features for opinion mining from forum posts. Glance et al. tried to mine business intelligence from forum data. Zhang et al. proposed algorithms to extract expertise network in forums. To harvest knowledge from forums, their content must be downloaded first. However, forum crawling is not a trivial problem. Generic crawlers, which adopt a breadth-first traversal strategy, are usually ineffective and inefficient for forum crawling. This is mainly due to two non crawler friendly characteristics of forums: 1) duplicate links and uninformative pages and 2) page-flipping links. A forum typically has many duplicate links that point to a common page but with different URLs, e.g., shortcut links pointing to the latest posts or URLs for user experience functions such as ―view by date‖ or ―view by title.‖ A generic crawler that blindly follows these links will crawl many duplicate pages, making it inefficient. A forum also has many uninformative pages such as login control to protect user privacy or forum software specific FAQs.

* 1. Existing system with drawbacks:

In the Existing system, Vidal et al proposed a method for learning regular expression patterns of URLs that lead a crawler from an entry page to target pages. Target pages were found through comparing DOM trees of pages with a preselected sample target page. It is very effective but it only works for the specific site from which the sample page is drawn. The same process has to be repeated every time for a new site. Therefore, it is not suitable to large- scale crawling.

**Disadvantages:**

1. Consuming large amount of data’s.

2. Time wasting while crawl in the web.

**1.3 Proposed system with features** **:**

We propose a new system for web crawl as FOCUS: Learning to Crawl Web Forums.It is a system overcome by existing crawl systems. In this method for learning regular expression patterns of URLs that lead a crawler from an entry page to target pages. Target pages were found through comparing DOM trees of pages with a pre-selected sample target page. It is very effective but it only works for the specific site from which the sample page is drawn. The same process has to be repeated every time for a new site. Therefore, it is not suitable to large- scale crawling. In contrast, FOCUS learns URL patterns across multiple sites and automatically finds forum entry page given a page from a forum. Experimental results show that FOCUS is effective in large scale forum crawling by leveraging crawling knowledge learned from a few annotated forum sites. A recent and more comprehensive work on forum crawling is iRobot. iRobot aims to automatically learn a forum crawler with minimum human intervention by sampling forum pages, clustering them, selecting informative clusters via an informativeness measure, and finding a traversal path by a spanning tree algorithm. However, the traversal path selection procedure requires human inspection.

**1.4 Features:**

FOCUS learns URL patterns across multiple sites and automatically finds forum entry page given a page from a forum. Experimental results show that FOCUS is effective in large scale forum crawling by leveraging crawling knowledge learned from a few annotated forum site.

**CHAPTER 2**

**LITERATURE SURVEY**

De-Duping URLs via Rewrite Rules: In the paper ―De-duping URLs via Rewrite Rules‖ is stated that a large fraction of the URLs on the web contain duplicate (or near-duplicate) content. De-duping URLs is an extremely important problem for search engines, since all the principal functions of a search engine, including crawling, indexing, ranking, and presentation, are adversely impacted by the presence of duplicate URLs.

Traditionally, the de-duping problem has been addressed by fetching and examining the content of the URL; their approach here is different. Given a set of URLs partitioned into equivalence classes based on the content (URLs in the same equivalence class have similar content), we address the problem of mining this set and learning URL rewrite rules that transform all URLs of an equivalence class to the same canonical form. These rewrite rules can then be applied to eliminate duplicates among URLs that are encountered for the first time during crawling, even without fetching their content. In order to express such transformation rules, they proposed a simple framework that is general enough to capture the most common URL rewrite patterns occurring on the web; in particular, it encapsulates the DUST (Different URLs with similar text) framework .They provide an efficient algorithm for mining and learning URL rewrite rules and show that under mild assumptions, it is complete, i.e., their algorithm learns every URL rewrite rule that is correct, for an appropriate notion of correctness. They demonstrate the expressiveness of their framework and the effectiveness of their algorithm by performing a variety of extensive large-scale experiments. Several previous studies have established that a large fraction of the web consists of duplicate URLs — syntactically distinct URLs having similar content. These duplicate URLs adversely affect the performance of commercial search engines in various ways. In crawling, they waste valuable bandwidth, affect refresh times, and impact politeness constraints; in indexing, they consume unnecessary disk space; in link-based ranking, they impart disproportionate authority to undeserving URLs; in presentation, they pollute displayed search results and lead to a poor user experience. Deducing URLs is thus an extremely important problem in end-to-end web search, and enormous resources are invested by search engines for this task. The traditional approach to de-duping has been to fetch the content of the URL and then apply standard fingerprinting methods on the content to eliminate duplicates. However, it is desirable to identify duplicate URLs as early in the workflow as possible, ideally even prior to crawling. Duplicate URLs occur on the web due to a multitude of reasons beyond blatant plagiarism. These include hosting the same set of URLs on different mirrors that are typically done for load balancing and fault tolerance, e.g., http://www-1.ibm.com and http://www-2.ibm.com. Often these are simple web-server based canonicalizations of URLs, e.g., dropping index.html from the website name, or other simple syntactic modifications such as removing the trailing slash, interchanging upper and lower cases etc. Dynamic scripts frequently encode session-specific identifying information in the URL that is used to track the user and the session but has no impact on the content of the page. The presence of such content-neutral parts in a URL is an important reason for the proliferation of duplicates. Furthermore, even the order of dynamic parameters is mostly inconsequential with respect to the content of a URL, e.g., the URLs http://domain/show.php?a=10&b=20 and http://domain/show.php?b=20&a=10 are the same. Unlike plagiarized content, these are structured transformations on the URL string that mostly happen due to server software. With a proper understanding of these transformations, it is possible to detect whether two URLs have similar content even without explicitly examining their content. Suppose we have a large collection of URLs along with their duplicate information, i.e., for every URL, which other URLs are duplicates (or near-duplicates) of this URL. Is it then possible to mine this collection and learn whether two URLs are duplicates of one another by examining only the URL strings? Equivalently, can we learn a set of rewrite rules that, given any two duplicate URLs, canonicalizes them to the same URL? If these rules are indeed learnable from an offline computation, they can be deployed in conjunction with the crawler to de-dup URLs prior to crawling them, thus ensuring that duplicate URLs are not even crawled! This is a significant departure from traditional de-duping approaches that require the content of URLs in order to identify duplicates. Considering that duplicate URLs constitute a large portion of the web, learning URL rewrite rules and deploying them in the crawler can tremendously improve the efficiency of not only the crawler but also subsequent steps in the processing pipeline. Given that URLs appear and disappear at a rapid rate, the value of mining and learning URL rewrite rules, especially in an offline manner, is not immediately apparent. Note however that the rewrite rules are typically specific to a particular web server, and more specifically, to the software used by the web servers. As a result, these rules are likely to be more stable and have a longer life than the actual URLs themselves Given a URL, they represent it as a function from a set of keys to a set of values. The set of key and values for the URL is defined as follows: first they split the URL into the static part comprising of the protocol, hostname, and the static path components, and the dynamic parts composed of the parameters and their values. They assumed that the set of separator tokens is known and fixed apriori; for the static part, they use the separator ―/‖ and for the dynamic part, they use the tokens ―?‖, ―&,‖ and ―;‖ for identifying key-value pairs, and the token ―=‖ for separating a key from a value. They represent the static portion with the static keys {k1, . . . , k`} corresponding to the ` components in the static part. For the dynamic part, each parameter in the URL is defined as a key. In this paper, they present a different approach to the URL de-duping problem based on automatically generating URL rewrite rules by mining a given collection of URLs with content-similarity information. These rewrite rules can then be applied to eliminate duplicates among URLs that are en countered for the first time during crawling, even without fetching their content. This has the huge advantage of trapping duplicates much earlier in a search-engine workflow, improving the efficiency of entire processing. Our framework is simple and has provable guarantees, and is shown to be effective in a large-scale experiment. In their formulation, they used a fixed set of delimiters; it will be useful to study the effect of a more flexible tokenization on their algorithm. For instance, can our method be used to detect site mirrors, by segmenting the hostname using ‗.‘ as the delimiter? Extending their formalization to capture a wider set of rules, while still being efficiently learnable is also an interesting research direction.

**3. REQUIREMENT SPECIFICATION**

**3.1 SOFTWARE AND HARDWARE REQUIRMENTS**

Software specifications of the system that is used in the project is

* **Operating system:** WINDOWS 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 2008 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.

**Hardware requirements:**

The selection of hardware is very important in the existence and proper working of any software. In the selection of hardware , the size and capacity requirements are also important.

* Processor Intel i3
* RAM 512MB RAM & above
* Hard Disk 80GB & above

**4. DESIGN**

**4.1 UML DIAGRAMS:**

The Unified Modeling Language (UML) is a graphical language for visualizing, specifying, constructing and documenting of a software intensive system. The UML gives a standard way to standard way to write a system blueprints, covering conceptual things, such as classes written in a specified programmed language, database schemas and reusable software components.

There are types of diagrams

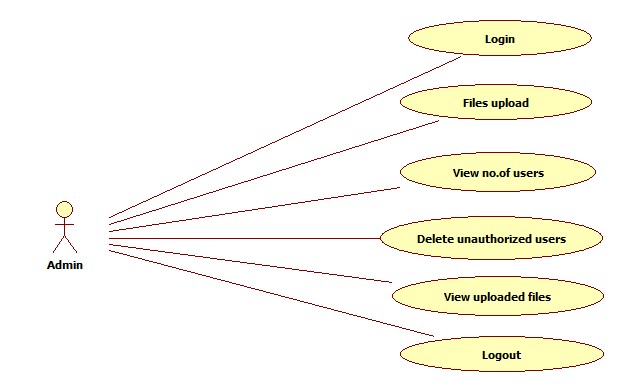
* Class Diagram
* Use case Diagram
* Sequence Diagram
* Collaboration Diagram
* Activity Diagram

**4.1.1 Use Case Diagram**

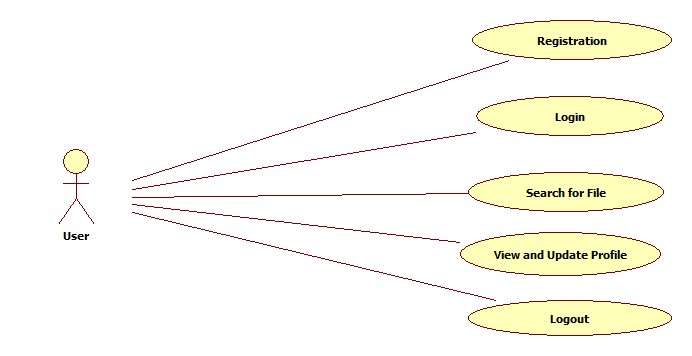
* Use case is a description of set of sequence of actions that a system performs that yields an observable results. These are important in organizing the behaviors of a system.
* Use case is used to structure the behavioral things in a model.
* A Use case is realized by collaboration.
* Graphically , a use case is rendered as a ellipse with solid lines, usually including only its name

**ACTOR**- also referred to as roles. Name and stereotype of an actor can be changed in its properties lab

**Use Cases-** these can have extension points.



**Admin Use case Diagram**

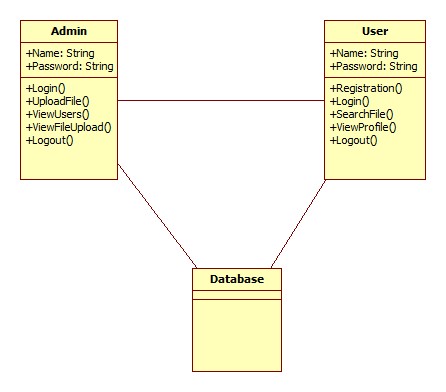


**User Use case Diagram**

**4.1.2 CLASS DIAGRAM**

The class diagram is a static diagram. It represents the static view of an application. Class diagram is not only used for visualizing , describing documenting different aspects of a system but also for constructing executable code of the software application. The class diagram describes the attributes and operations of a class and also the constraints imposed on the system. The class diagrams are widely used in the modeling of object oriented system because they are the only UML diagrams which can be mapped directly with object oriented languages.

The class diagram shows a collection of classes, interfaces, associations, collaboration and constraints. It is also known as ***structural diagram*.**

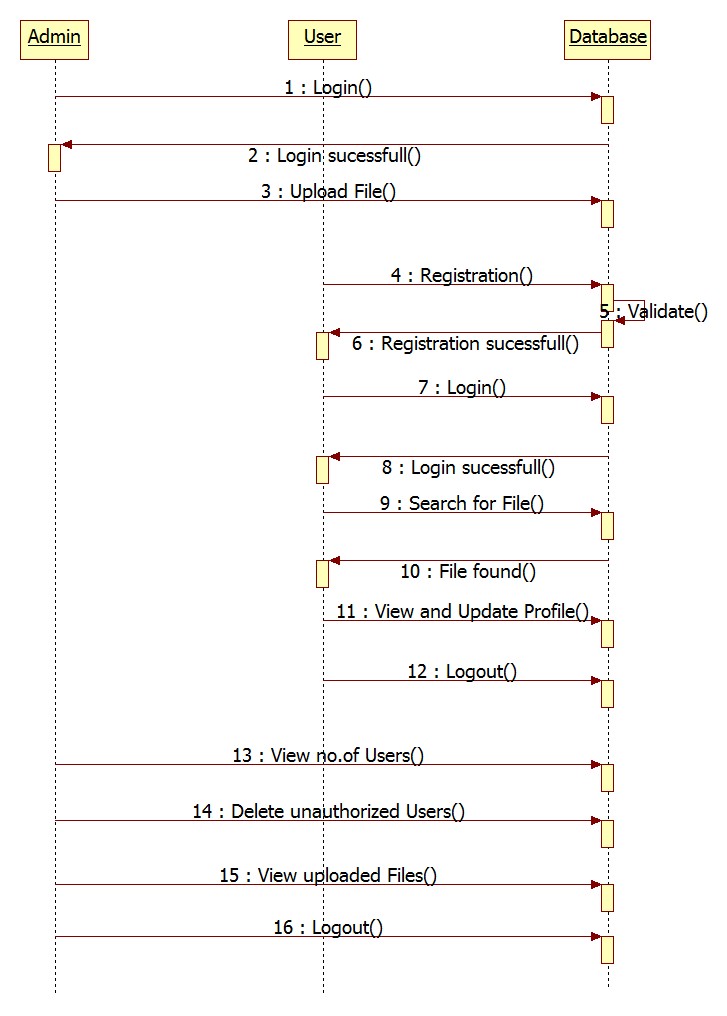
s

**Class Diagram**

**4.1.3 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 received by those objects. The objects are typically named or anonymous instances of classes, but may also represent instances of other thing such as collaboration, components and nodes. These are used to illustrate the dynamic view of a system. It is isomorphic, which means taking any one of them in to consideration, and can be transformed to another.

A Sequence diagram shows, as parallel vertical lines (*Life lines*), a different processes or objects that live simultaneously, and as horizontal arrows, the messages exchanged between them, in the in which they occur. This allows the specification of simple runtime scenarios in a graphical manner. In order to show the sequence of time ordering of messages involved in the working of PHC a sequence diagram was used.

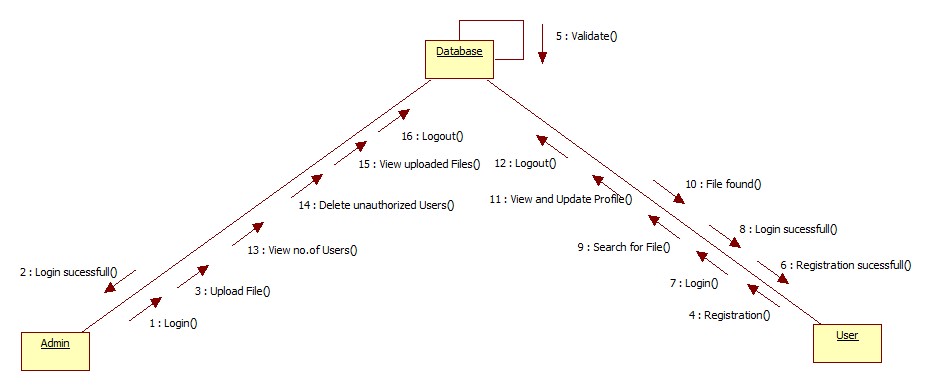


**Sequence Diagram**

**4.1.4 COLLABORATION DIAGRAM:**

A Collaboration diagram is an interaction diagram that emphasizes the structural organization of the objects that send and receive messages. Collaboration diagrams are isomorphic, meaning that you can take one and transform it into another.

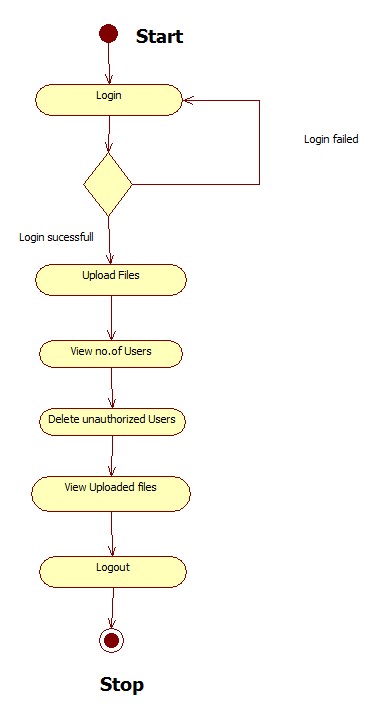
Graphically, collaboration is rendered as ellipse with dashed lines, usually including only its name.



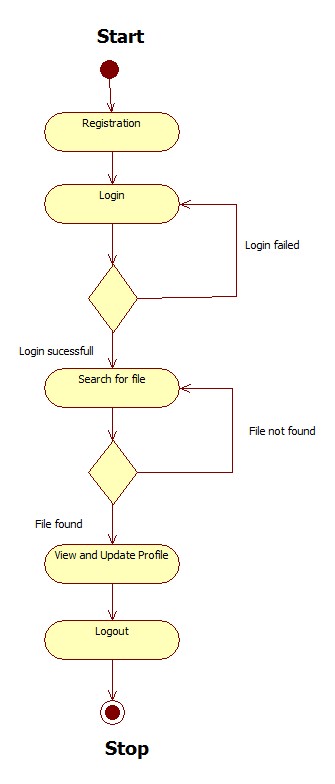
**Collaboration Diagram**

**4.1.5 ACTIVITY DIAGRAMS:**

Activity diagrams are graphical representations of work flows of stepwise activities and actions with support for choice, iteration and concurrency. In the Unified Modeling Language, activity diagrams are intended to model both computational and organizational processes. Activity diagrams show the overall flow of control.



**Admin Activity Diagram**



**User Activity Diagram**

**CHAPTER 5**

**IMPLEMENTATION**

Implementation is the stage of the project when the theoretical design is turned out into a working system. Thus it can be considered to be the most critical stage in achieving a successful new system and in giving the user, confidence that the new system will work and be effective.

The implementation stage involves careful planning, investigation of the existing system and it’s constraints on implementation, designing of methods to achieve changeover and evaluation of changeover methods.

Implementation is the process of converting a new system design into operation. It is the phase that focuses on user training, site preparation and file conversion for installing a candidate system. The important factor that should be considered here is that the conversion should not disrupt the functioning of the organization.

The application is completely written .net .This enables the debit card issues to use this application across a variety of devices independent of the vendor of the devices. We use sql as a back end for storing database.

**5.1: TECHNOLOGY USED:**

**5.1.1: Dot Net:**

The .NET Framework (pronounced dot net) is a proprietary, partially open source freeware software framework developed by Microsoft that runs primarily on Microsoft Windows.

**5.1.2: What is Dot Net?**

The **.NET** is the technology from Microsoft, on which all other Microsoft technologies will be depending on in future. It is a major technology change, introduced by Microsoft, to catch the market from the SUN's Java. Few years back, Microsoft had only VC++ and VB to compete with Java, but Java was catching the market very fast. With the world depending more and more on the Internet/ Web and java related tools becoming the best choice for the web applications, Microsoft seemed to be losing the battle. Thousands of programmers moved to java from VC++ and VB. To recover the .market, .Microsoft announced .NET.

But Microsoft has a wonderful history of starting late but catching up quickly. This is true in case of .NET too. Microsoft put their best men at work for a secret project called Next Generation Windows Services (NGWS)., under the direct supervision of Mr. Bill Gates.

The outcome of the project is what we now know as .NET. Even though .NET has borrowed most of its ideas from Sun's J2EE, it has really outperformed their competitors.

Microsoft's VC++ was a powerful tool. But it was too complex. It has too many data types, and developers had to learn many libraries including Windows SDK, MFC, ATL, COM etc. There were many data type compatibility issues while exchanging data between different layers. Visual Basic was too easy, and many serious programmers hated it just for that reason. Even though Visual basic was very easy to use, it was not very flexible to develop serious applications. SUN's Java became a very good choice for these reasons. It had the flexibility and power of C++ and at the same time easy enough to catch the attention of VB programmers.

**5.1.3: The structure of Dot Net:**

* It is a platform for application developers.
* It is a Framework that supports Multiple Language and Cross language integration.
* IT has IDE (Integrated Development Environment).
* Framework is a set of utilities or can say building blocks of your application system.
* .NET Framework provides GUI in a GUI manner.
* .NET is a platform independent but with help of Mono Compilation System (MCS). MCS is a middle level interface.
* .NET Framework provides interoperability between languages i.e. Common Type System (CTS).
* .NET Framework also includes the .NET Common Language Runtime (CLR), which is responsible for maintaining the execution of all applications developed using the .NET library.
* The .NET Framework consists primarily of a gigantic library of code.



**Class Library:**

.NET Framework includes a set of [standard](https://en.wikipedia.org/wiki/Standard_library) class libraries. The class library is organized in a hierarchy of [namespaces](https://en.wikipedia.org/wiki/Namespaces). Most of the built-in APIs are part of either system or Microsoft namespaces. These class libraries implement a large number of common functions, such as file reading and writing, graphic rendering, database interaction, and XML document manipulation, among others. .NET class libraries are available to all [CLI compliant languages](https://en.wikipedia.org/wiki/List_of_CLI_languages). .NET Framework class library is divided into two parts: [Framework Class Library](https://en.wikipedia.org/wiki/Framework_Class_Library) (FCL) and [Base Class Library](https://en.wikipedia.org/wiki/Base_Class_Library) (BCL).

FCL includes a small subset of the entire class library and is the core set of classes that serve as the basic [API](https://en.wikipedia.org/wiki/API) of CLR. Classes in and some classes.inSystem.dll and System.core.dll are part of FCL. FCL classes are available in .NET Framework as well as its alternative implementations including [.NET Compact Framework](https://en.wikipedia.org/wiki/.NET_Compact_Framework), [Microsoft Silverlight](https://en.wikipedia.org/wiki/Microsoft_Silverlight) and [Mono](https://en.wikipedia.org/wiki/Mono_(software)).

BCL is a superset of FCL and refers to the entire class library that ships with .NET Framework. It includes an expanded set of libraries, including [Windows Forms](https://en.wikipedia.org/wiki/Windows_Forms), [ADO.NET](https://en.wikipedia.org/wiki/ADO.NET), [ASP.NET](https://en.wikipedia.org/wiki/ASP.NET), [Language Integrated Query](https://en.wikipedia.org/wiki/Language_Integrated_Query) (LINQ), [Windows Presentation Foundation](https://en.wikipedia.org/wiki/Windows_Presentation_Foundation) (WPF), [Windows Communication HYPERLINK "https://en.wikipedia.org/wiki/Windows\_Communication\_Foundation"Foundation](https://en.wikipedia.org/wiki/Windows_Communication_Foundation) (WCF) and [Workflow Foundation](https://en.wikipedia.org/wiki/Workflow_Foundation) (WF). BCL is much larger in scope than standard libraries for languages like [C++](https://en.wikipedia.org/wiki/C%2B%2B), and comparable in scope to [standard libraries of Java](https://en.wikipedia.org/wiki/Java_Class_Library).

**.NET Core:**

.NET Core is a free and open-source partial implementation of the .NET Framework. It consists of CoreCLR and CoreFX, which are partial forks of CLR and BCL respectively.NET Core comes with an improved JIT compiler, called RyuJIT.

**Assemblies:**

Compiled CIL code is stored in CLI assemblies. As mandated by the specification, assemblies are stored in Portable Executable (PE) file format, common on Windows platform for all DLL and EXE files. Each assembly consists of one or more files, one of which must contain a manifest bearing the metadata for the assembly. The complete name of an assembly (not to be confused with the file name on disk) contains its simple text name, version number, culture, and public key token. Assemblies are considered equivalent if they share the same complete name, excluding the revision of the version number.

A private key can also be used by the creator of the assembly for strong naming. The public key token identifies which private key an assembly is signed with. Only the creator of the keypair(typically .NET developer signing the assembly) can sign assemblies that have the same strong name as a previous version assembly, since the creator is in possession of the private key. Strong naming is required to add assemblies to Global Assembly Cache.

**C++/CLI:**

Microsoft introduced C++/CLI in Visual Studio 2005, which is a language and means of compiling Visual C++ programs to run within the .NET Framework. Certain portions of the C++ program still run within an unmanaged Visual C++ Runtime, while specially modified portions are translated into CIL code and run with the .NET Framework's CLR.

Assemblies compiled using the C++/CLI compiler are known as mixed-mode assemblies, since they contain native and managed code within the same DLL, Such assemblies are also difficult to reverse engineer, since .NET de compilers such as .NET Reflector only reveal the managed code.

**Security:**

.NET Framework has its own security mechanism with two general features: Code Access Security (CAS), and validation and verification. CAS is based on evidence that is associated with a specific assembly. Typically the evidence is the source of the assembly (whether it is installed on the local machine or has been downloaded from the intranet or Internet). CAS uses evidence to determine the permissions granted to the code. Other code can demand that calling code be granted a specified permission. The demand causes CLR to perform a call stack walk: every assembly of each method in the call stack is checked for the required permission; if any assembly is not granted the permission a security exception is thrown.

Managed CIL byte code is easier to reverse-engineer than native code, unless obfuscated. NET decompile programs enable developers with no reverse-engineering skills to view the source code behind unobfuscated .NET assemblies. In contrast, apps compiled to native machine code are much harder to reverse-engineer and source code is almost never produced successfully, mainly because of compiler optimizations and lack of reflection. One concern is over possible loss of trade secrets and the bypassing of license control mechanisms. To mitigate this, Microsoft has included Dotfuscator Community Edition with Visual Studio .NET since 2002.[[b]](https://en.wikipedia.org/?title=.NET_Framework) Third-party obfuscation tools are also available from vendors such as vmware, V.i. Labs, Xenocode, Red Gate Software. Method-level encryption tools for .NET code are available from vendors such as Safe Net.

**Memory management:**

CLR frees the developer from the burden of managing memory (allocating and freeing up when done); it handles memory management itself by detecting when memory can be safely freed. Instantiations of .NET types (objects) are allocated from the managed heap; a pool of memory managed by CLR. As long as there exists a reference to an object, which might be either a direct reference to an object or via a graph of objects, the object is considered to be in use. When there is no reference to an object, and it cannot be reached or used, it becomes garbage, eligible for collection.

.NET Framework includes a garbage collector (GC) which runs periodically, on a separate thread from the application's thread, that enumerates all the unusable objects and reclaims the memory allocated to them. It is a non-deterministic, compacting, mark-and-sweep garbage collector. GC runs only when a certain amount of memory has been used or there is enough pressure for memory on the system. Since it is not guaranteed when the conditions to reclaim memory are reached, GC runs are non-deterministic. Each .NET application has a set of roots, which are pointers to objects on the managed heap (*managed objects*). These include references to static objects and objects defined as local variables or method parameters currently in scope, as well as objects referred to by CPU registers. When GC runs, it pauses the application, and for each object referred to in the root, it recursively enumerates all the objects reachable from the root objects and marks them as reachable. It uses CLI metadata and reflection to discover the objects encapsulated by an object, and then recursively walk them. It then enumerates all the objects on the heap (which were initially allocated contiguously) using reflection. All objects not marked as reachable are garbage. This is the *mark* phase. Since the memory held by garbage is not of any consequence, it is considered free space. However, this leaves chunks of free space between objects which were initially contiguous. The objects are then *compacted* together to make used memory contiguous again. Any reference to an ect invalidated by moving the object is updated by GC to reflect the new location. The application is resumed after the garbage collection is over.

GC used by .NET Framework is also generational. Objects are assigned a *generation*; newly created objects belong to *G*eneration *0*. The objects that survive a garbage collection are tagged as *G*eneration *1*, and the Generation 1 objects that survive another collection are Generation *2* objects. .NET Framework uses up to Generation 2 objects. Higher generation objects are garbage collected less frequently than lower generation objects. This helps increase the efficiency of garbage collection, as older objects tend to have a longer lifetime than newer objects. Thus, by eliminating older (and thus more likely to survive a collection) objects from the scope of a collection run, fewer objects need to be checked and compacted.

**Performance:**

When an application is first launched, the .NET Framework compiles the CIL code into executable code using its just-in-time compiler, and caches the executable program into the .NET Native Image Cache. Due to caching, the application launches faster for subsequent launches, although the first launch is usually slower. To increase speed of the first launch, developers may use the Nativ Image Generator utility to manually compile and cache any .NET application, ahead-of-time.

The garbage collector, which is integrated into the environment, can introduce unanticipated delays of execution over which the developer has little direct control. "In large applications, the number of objects that the garbage collector needs to deal with can become very large, which means it can take a very long time to visit and rearrange all of them.

.NET Framework provides support for calling Streaming SIMD Extensions (SSE) via managed code from April 2014 in Visual Studio 2013 Update 2. However, Mono has provided support for SIMD Extensions as of version 2.2 within the Mono.Simd namespace; before. Mono's lead developer Miguel de Icaza has expressed hope that this SIMD support will be adopted by CLR's ECMA standard.Streaming SIMD Extensions have been available in x86 CPUs since the introduction of the Pentium III. Some other architectures such as ARM and MIPS also have SIMD extensions. In case the CPU lacks support for those extensions, the instructions are simulated in software

**Registration Page:**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.UI;

using System.Web.UI.WebControls;

using System.Data;

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

{

Class1 obj = new Class1();

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

{

txtdob\_CalendarExtender.EndDate = DateTime.Now;

}

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

{

try

{

string qry = "insert into reg values('" + txtname.Text + "','" + txtemail.Text + "','" + txtpass.Text + "'," + txtmobile.Text + ",'" + txtloc.Text + "','" + txtdob.Text + "','" + ddlgender.SelectedItem.Text + "')";

int i = obj.inupdel(qry);

if (i > 0)

{

Label12.Text = "Register Sucessfully Completed";

txtname.Text = txtemail.Text = txtdob.Text = txtloc.Text = txtmobile.Text = txtpass.Text = "";

}

else

{

Label12.Text = "Not Yet Registered";

}

}

catch (Exception)

{

Label12.Text = "EmailId Already exits ";

}

}

}

**Login Page:**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.UI;

using System.Web.UI.WebControls;

using System.Data;

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

{

Class1 obj = new Class1();

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

{

}

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

{

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

{

string qry = "select \* from reg where email='" + TextBox1.Text + "' and pass='" + TextBox2.Text + "'";

DataSet ds = new DataSet();

ds = obj.select(qry);

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

{

Session["stuid"] = ds.Tables[0].Rows[0][0];

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

Session["pas"] = ds.Tables[0].Rows[0][3];

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

}

else

{

Label6.Text = "Invalid Username and Password";

}

}

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

{

if (TextBox1.Text == "admin" && TextBox2.Text == "admin")

{

Session["aname"] = TextBox1.Text;

Response.Redirect("~/Admin/Adminhome.aspx");

}

else

{

Label6.Text = "Invalid Username and Password";

}

}

}

}

**User Home:**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.UI;

using System.Web.UI.WebControls;

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

{

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

{

if (Session["stuid"] == null && Session["name"] == null)

{

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

}

else

{

Label2.Text = Session["name"].ToString();

}

}

}

**Update Password :**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.UI;

using System.Web.UI.WebControls;

public partial class User\_Update\_Password : System.Web.UI.Page

{

Class1 obj = new Class1();

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

{

}

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

{

if (TextBox1.Text == Session["pas"].ToString())

{

try

{

string qry = "update reg set pass='" + TextBox2.Text + "' where name='" + Session["name"].ToString() + "'";

int j = obj.inupdel(qry);

if (j > 0)

{

Response.Write("<script>alert('updated successfully')</script>");

TextBox1.Text = TextBox2.Text = TextBox3.Text = "";

}

}

catch (Exception ex)

{

Response.Write(ex.Message);

}

}

else

{

Response.Write("<script>alert('Please Enter correct Password')</script>");

TextBox1.Text = TextBox2.Text = TextBox3.Text = "";

}

}

}

**Feedback Page:**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.UI;

using System.Web.UI.WebControls;

using System.Data;

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

{

Class1 obj = new Class1();

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

{

if (!IsPostBack)

{

string x = Session["name"].ToString();

TextBox1.Text = x;

}

}

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

{

string qry = "insert into feedback values('" + TextBox1.Text + "','" + TextBox3.Text + "','" + TextBox4.Text + "')";

int i = obj.inupdel(qry);

if (i > 0)

{

Response.Write("<script>alert('Succesfully sent to Admin')</script>");

clear();

}

else

{

Response.Write("<script>alert('Not Yet sent Please Fill the Message')</script>");

}

}

public void clear()

{

TextBox3.Text = TextBox4.Text = "";

}

protected void Button2\_Click(object sender, EventArgs e)

{

clear();

}

}

**User Details:**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.UI;

using System.Web.UI.WebControls;

using System.Data;

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

{

Class1 obj = new Class1();

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

{

bind();

}

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

{

Response.Redirect("~/User/Update Password.aspx");

}

public void bind()

{

string a=Session["name"].ToString();

string qry = "select \* from reg where name='" + a + "'";

DataSet ds = obj.select(qry);

DataList1.DataSource = ds;

DataList1.DataBind();

}

}

**Search Page:**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.UI;

using System.Web.UI.WebControls;

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

{

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

{

}

protected void img\_click(object sender, ImageClickEventArgs e)

{

if (TextBox1.Text.Trim() != "")

{

Response.Redirect("~/User/" + TextBox1.Text + ".aspx");

}

}

}

**Admin Module:**

**Feedback Page:**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.UI;

using System.Web.UI.WebControls;

using System.Data;

public partial class Admin\_feedbackusers : System.Web.UI.Page

{

Class1 obj = new Class1();

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

{

if (!IsPostBack)

{

bind();

}

}

public void bind()

{

string qry = "select \* from feedback";

DataSet ds = obj.select(qry);

GridView1.DataSource = ds;

GridView1.DataBind();

}

}

**User details:**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.UI;

using System.Web.UI.WebControls;

using System.Data;

public partial class Admin\_Userdetails : System.Web.UI.Page

{

Class1 obj = new Class1();

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

{

bind();

}

public void bind()

{

string qry = "select \* from reg ";

DataSet ds = obj.select(qry);

GridView1.DataSource = ds;

GridView1.DataBind();

}

}

**Upload File:**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.UI;

using System.Web.UI.WebControls;

using System.IO;

using System.Text;

using System.Windows.Forms;

using System.Text.RegularExpressions;

public partial class Admin\_Uploadfile : System.Web.UI.Page

{

Class1 obj = new Class1();

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

{

Panel1.Visible = true;

Panel2.Visible = false;

if (!IsPostBack)

{

Random rd = new Random();

int i = rd.Next(9999);

string x = i.ToString();

txtid.Text = x;

}

}

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

{

if (FileUpload1.HasFile)

{

string ex = Path.GetExtension(FileUpload1.FileName);

if (ex == ".txt")

{

string filename = Path.Combine(Server.MapPath("~/Text"), FileUpload1.FileName);

Session["f"] = Path.GetFileName(FileUpload1.FileName);

FileUpload1.SaveAs(filename);

string temp = "";

string[] keyword = txtkeypoint.Text.Split(',');

for (int i = 0; i < keyword.Length; i++)

{

string s = keyword[i];

FileStream file = new FileStream(filename, FileMode.Open, FileAccess.ReadWrite, FileShare.None);

StreamReader sr = new StreamReader(file);

while (!sr.EndOfStream)

{

temp = sr.ReadToEnd();

if (temp.Contains(s))

{

Regex r = new Regex(s);

temp = r.Replace(temp, "<a href='../User/" + s + ".aspx'>" + s + "</a>");

//string g = filename + temp;

break;

}

}

sr.Close();

File.WriteAllText(filename, temp);

TextBox5.Text = File.ReadAllText(filename);

Panel2.Visible = true;

}

}

else

{

Response.Write("<script>alert('Please Upload Only Text Files')</script>");

}

}

else

{

Response.Write("<script>alert('Please Upload The File')</script>");

}

}

protected void Button3\_Click(object sender, EventArgs e)

{

//string f = Path.GetFileName(FileUpload1.FileName);

string f1 = Session["f"].ToString();

try

{

string qry = "insert into Filedetails values('" + txtid.Text + "','" + txtfname.Text + "','" + txtdate.Text + "','" + f1 + "','" + txtkeypoint.Text + "')";

int j = obj.inupdel(qry);

if (j > 0)

{

clear();

Response.Write("<script>alert('File Has been Uploded Succesfully')</script>");

}

else

{

Response.Write("<script>alert('Not Yet Uploaded')</script>");

}

}

catch (Exception ex1)

{

Response.Write(ex1.Message);

}

}

protected void Button2\_Click(object sender, EventArgs e)

{

clear();

}

public void clear()

{

txtdate.Text = txtfname.Text = txtid.Text = txtkeypoint.Text = "";

}

}

**File Details:**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.UI;

using System.Web.UI.WebControls;

using System.Data;

public partial class Admin\_FileDetails : System.Web.UI.Page

{

Class1 obj = new Class1();

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

{

if (!IsPostBack)

{

bind();

}

}

public void bind()

{

try

{

string qry = "select \* from Filedetails ";

DataSet ds = obj.select(qry);

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

{

GridView1.DataSource = ds;

GridView1.DataBind();

}

}

catch (Exception ex)

{

Response.Write(ex.Message);

}

}

}

**User Feedback:**

using System.Web.UI;

using System.Web.UI.WebControls;

using System.Data;

using System.Data.SqlClient;

public partial class Admin\_UsersFeedback : System.Web.UI.Page

{

Class1 cs = new Class1();

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

{

if (!IsPostBack)

{

bind();

}

}

public void bind()

{

string qry = "select \* from feedback";

DataSet ds = cs.select(qry);

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

{

GridView1.DataSource = ds;

GridView1.DataBind();

}

else

{

Response.Write("<script>alert('No Queries Present')</script>");

}

}

protected void GridView1\_SelectedIndexChanged(object sender, EventArgs e)

{

}

}

**Forgot Password:**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.UI;

using System.Web.UI.WebControls;

using System.Data;

using System.Data.SqlClient;

using System.Net.Mail;

using System.Net;

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

{

Class1 cs= new Class1();

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

{

}

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

{

try

{

string qry = "select Password from User2 where EmailId='" + email.Text + "'";

DataSet ds = cs.select(qry);

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

{

MailMessage msg = new MailMessage();

msg.From = new MailAddress("mayorpharmacy3@gmail.com");

msg.IsBodyHtml = true;

msg.To.Add(new MailAddress(email.Text));

msg.Subject = "A Mail Regarding Password";

msg.Body = "Your Password from RentalXpress Website is : " + ds.Tables[0].Rows[0][0].ToString();

SmtpClient smtp = new SmtpClient("smtp.gmail.com", 587);

NetworkCredential nc = new NetworkCredential("mayorpharmacy3@gmail.com", "pharmacy333");

smtp.UseDefaultCredentials = false;

smtp.Credentials = nc;

smtp.EnableSsl = true;

smtp.Send(msg);

Response.Write("<script>alert('Mail Sent Successfully')</script>");

email.Text = "";

}

else

{

//lblmsg.Text = "MailId Doesn't Exist";

Response.Write("<script>alert('MailId not existed')</script>");

}

}

catch (Exception)

{

//lblmsg.Text = "Failure in sending Mail";

Response.Write("<script>alert('Failure in sending Mail')</script>");

}

}

.

**5.2 Frontend Design  
5.2.1 Main Page**

****

Fig 5.2.1 Main page

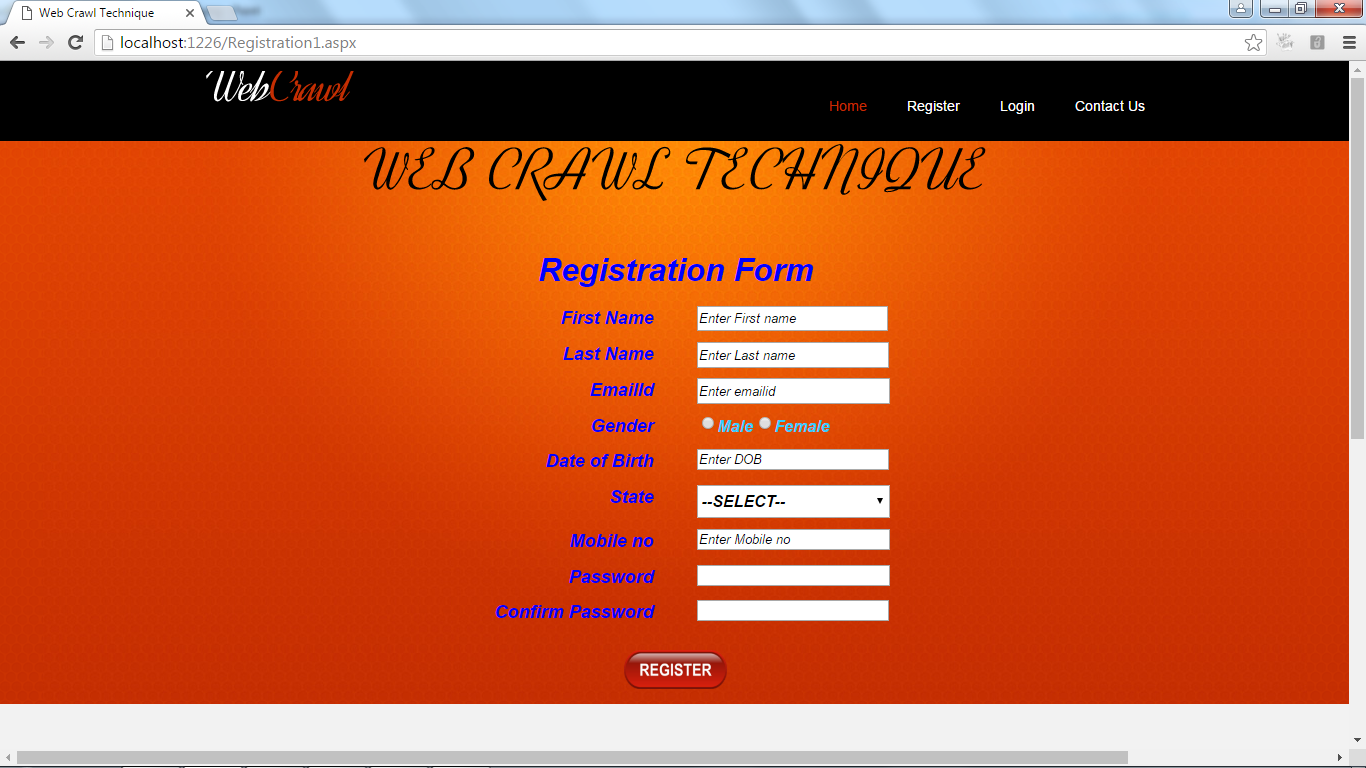
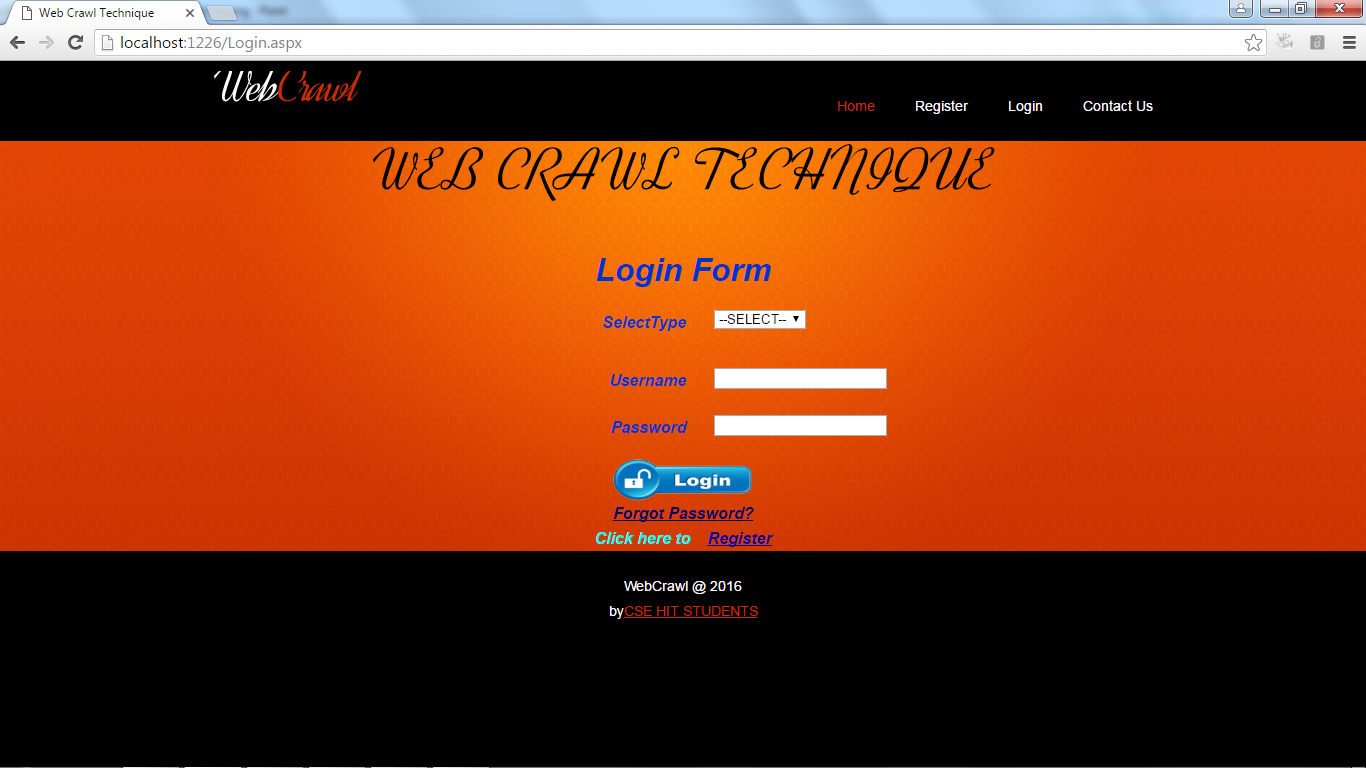
**5.2.2 Registration page** ****

Fig: 5.2.2 Registration page

**5.2.3 Login page**



5.2.3 Login page

**5.2.4 User home**



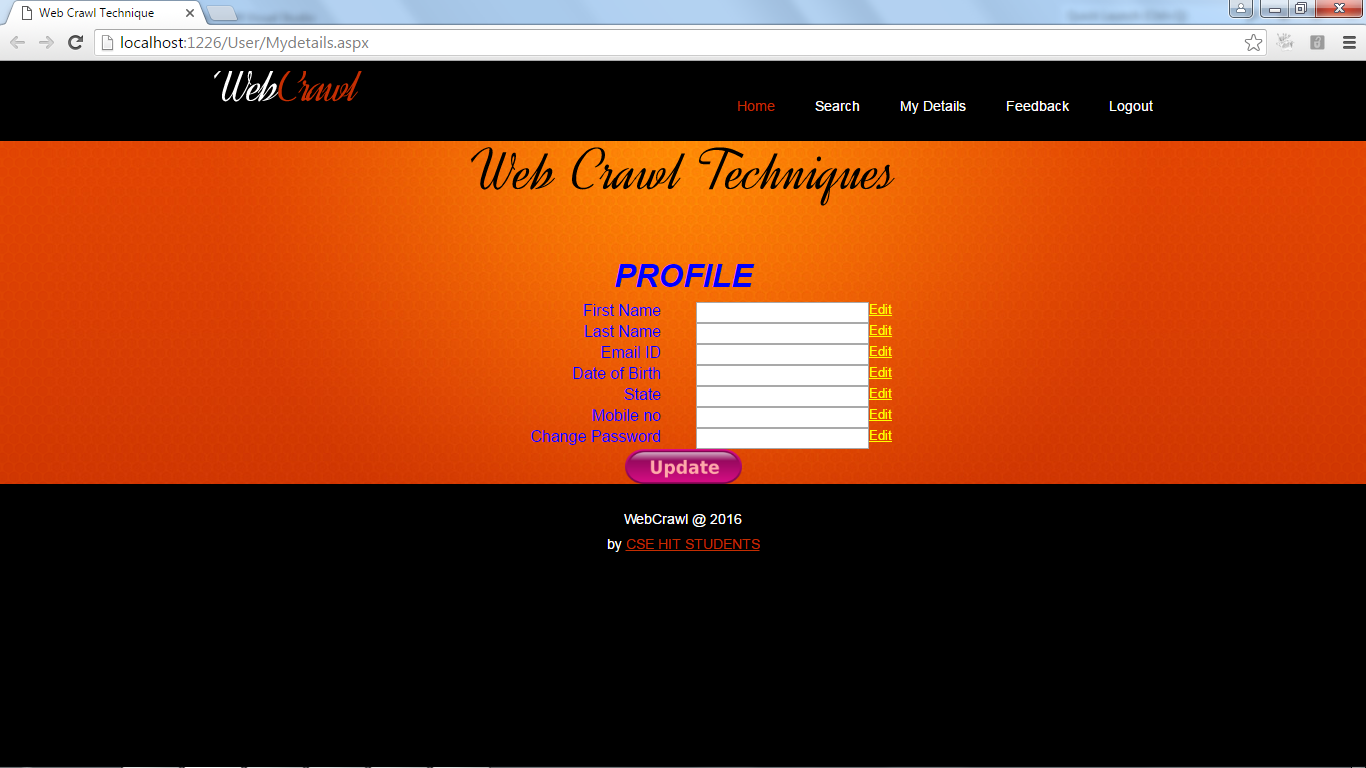
Fig 5.2.4.User home

**5.2.5 Search page**



 Fig: 5.2.5 Search page

**5.2.6 My Details**

**** Fig: 5.2.6 My Details

**5.2.7 Feedback Page**

****Fig 5.2.7 Feedback

**5.2.8 Admin Login**

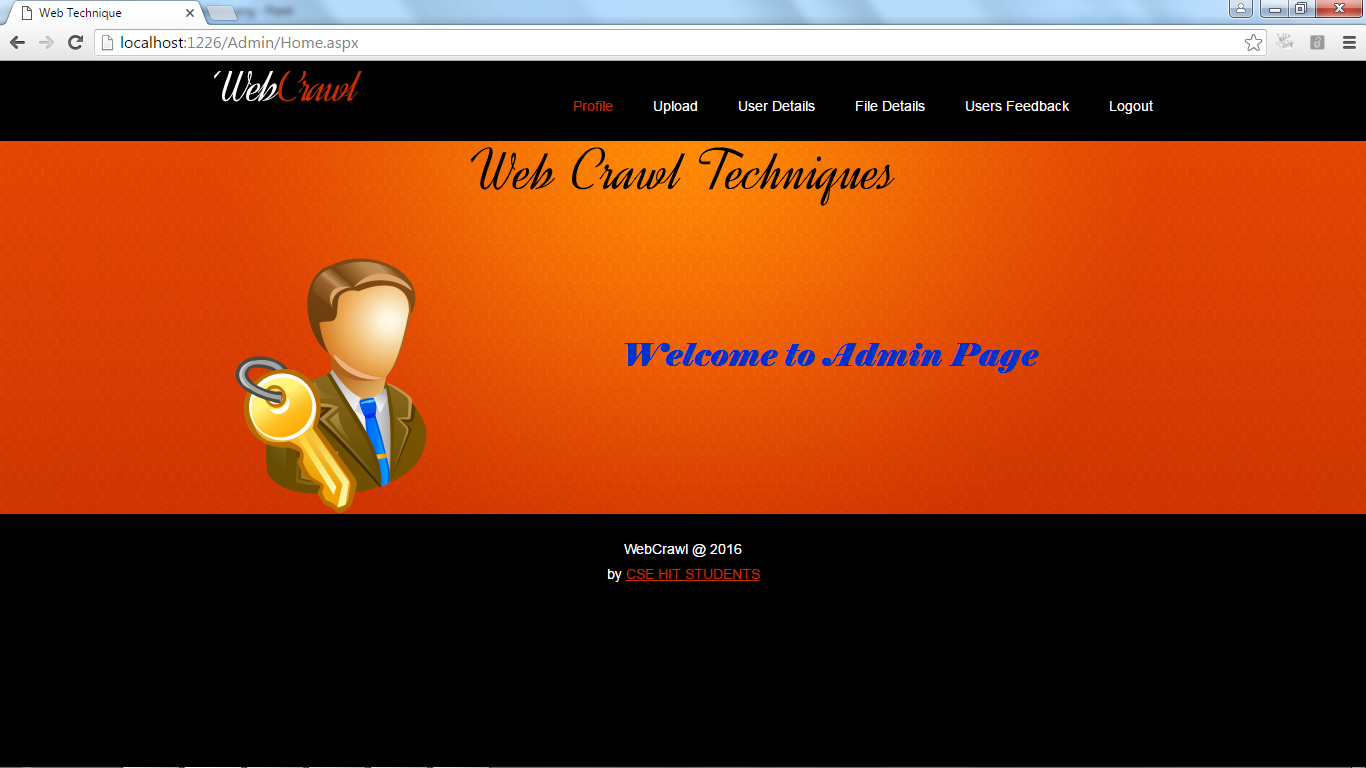
****

Fig 5.2.8 Admin Login

**5.2.9Upload file**

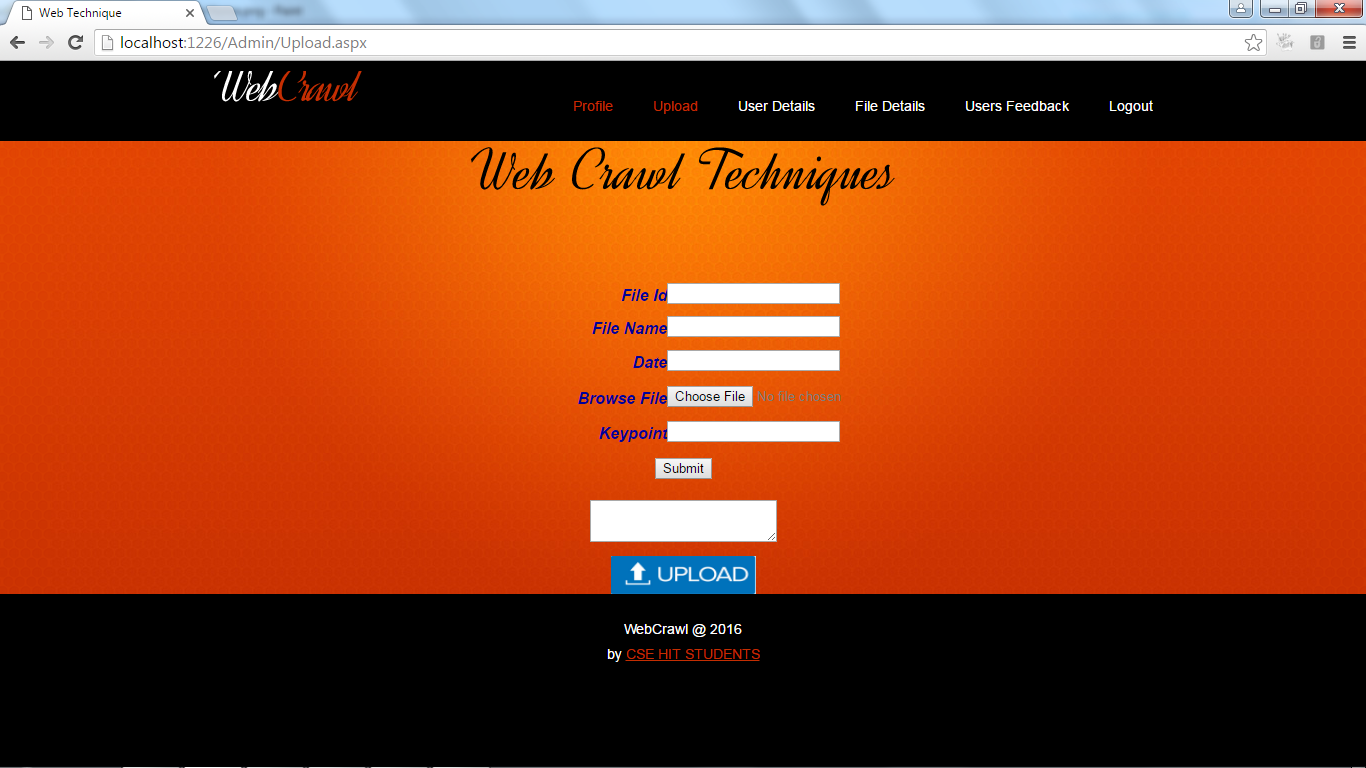
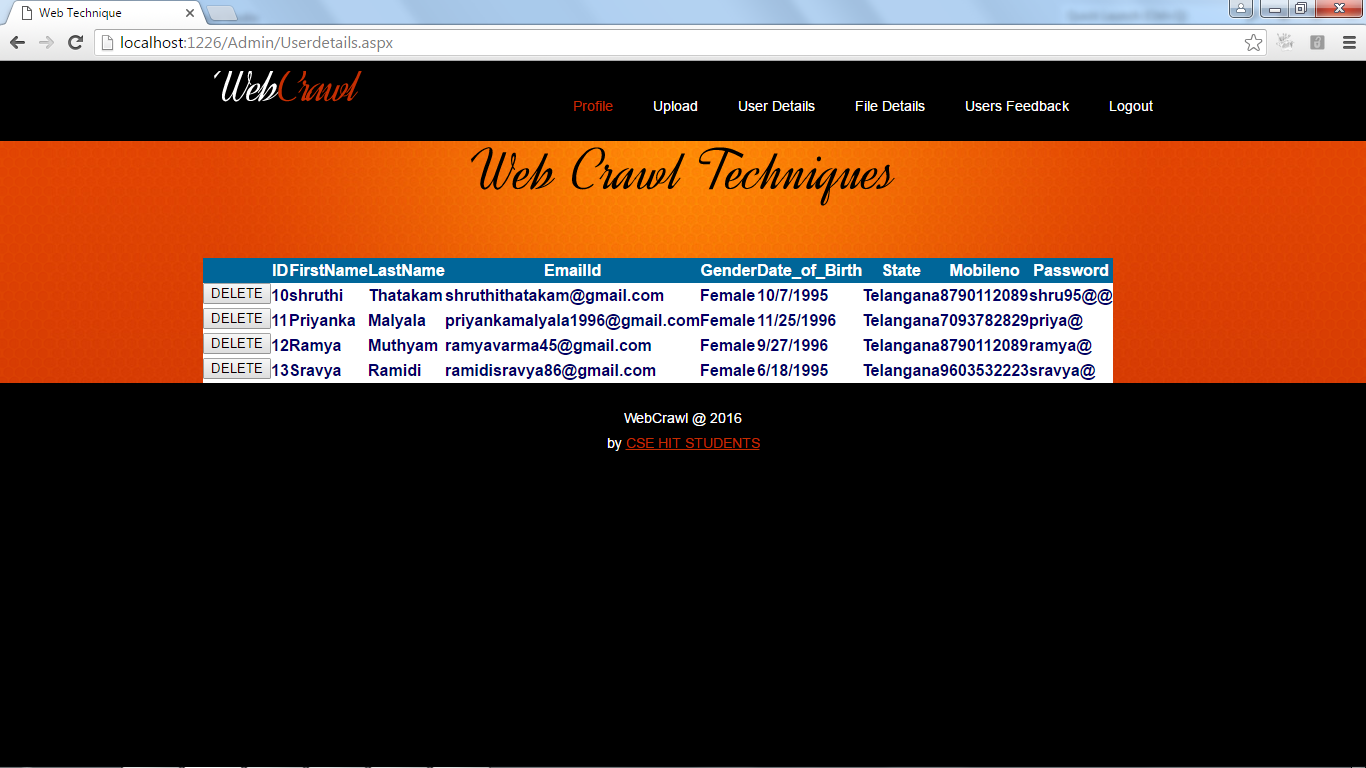


Fig 5.2.9 Upload File

**5.2.10 User Details**



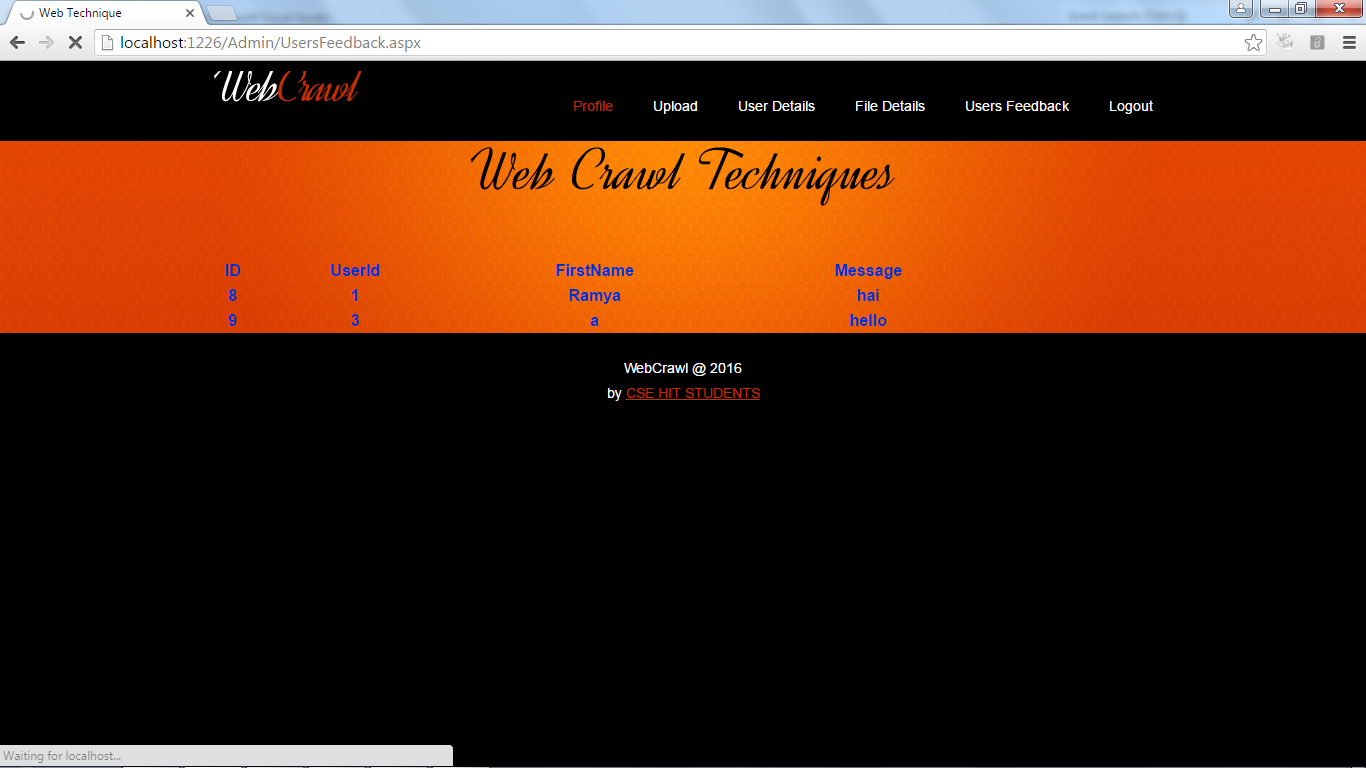
5.2.10user details

**5.2.11 File Details**

****

5.2.11. File Details

**5.2.12 User Feedback**

****5.2.12 User Feedback

**5.2.13 Forgot Password**

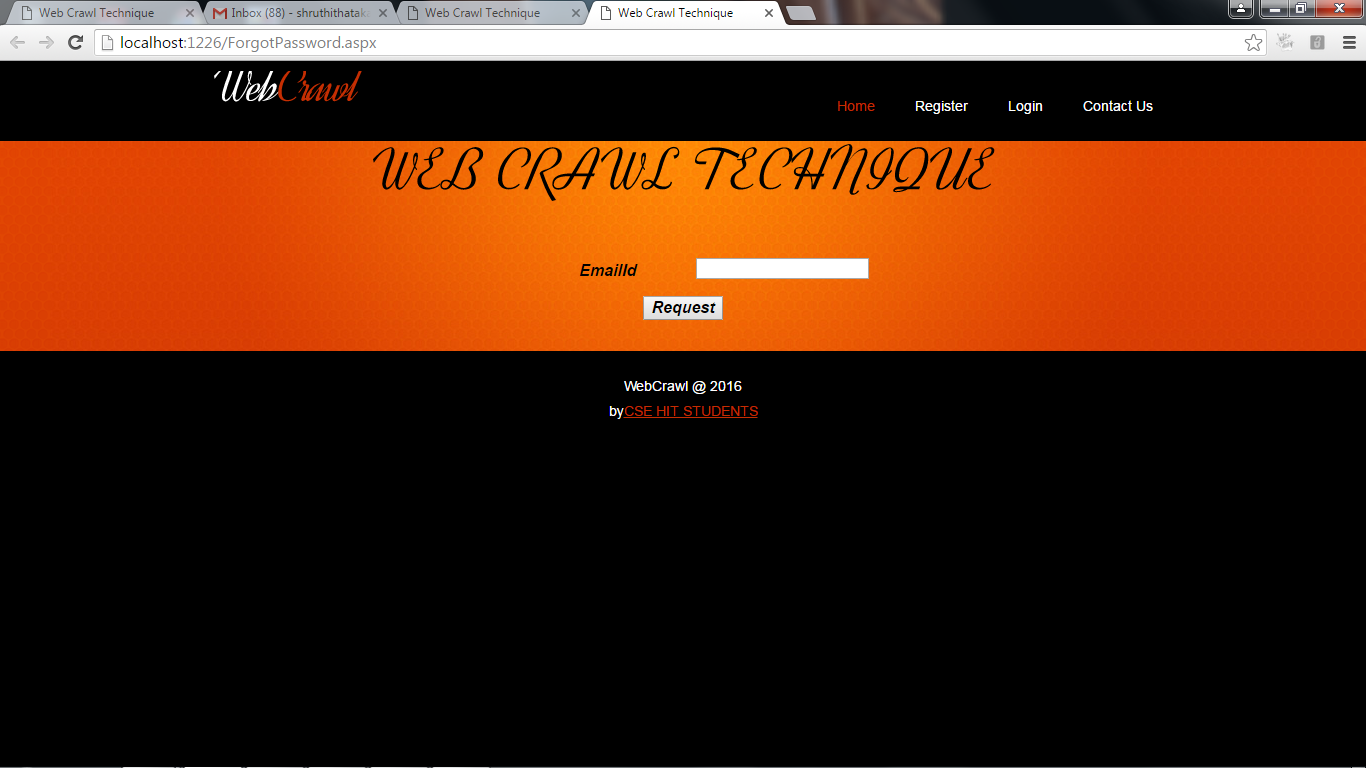
****

Fig 5.2.13 Forgot Password

**CHAPTER-6**

**TESTING**

**6.1 Introduction**

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub assemblies, assemblies and/or a finished product It is the process of exercising software with the intent of ensuring that the

Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of test. Each test type addresses a specific testing requirement.

**TYPES OF TESTS 4**

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application .it is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

UNIT TESTING

MODULE TESTING

SUB-SYSTEM TESING

SYSTEM TESTING

ACCEPTANCE TESTING

Component Testing

Integration Testing

User Testing

UNIT TESTING

MODULE TESTING

SUB-SYSTEM TESING

SYSTEM TESTING

ACCEPTANCE TESTING

Component Testing

Integration Testing

User Testing

Fig: 6.1 Testing

**Integration testing**

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfaction, as shown by successfully unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components.

**Functional test**

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

Functional testing is centered on the following items:

Valid Input : identified classes of valid input must be accepted.

Invalid Input : identified classes of invalid input must be rejected.

Functions : identified functions must be exercised.

Output : identified classes of application outputs must be exercised.

Systems/Procedures: interfacing systems or procedures must be invoked. Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identify Business process flows; data fields, predefined processes, and successive processes must be considered for testing. Before functional testing is complete, additional tests are identified and the effective value of current tests is determined.

**System Test**

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

**White Box Testing**

White Box Testing is a testing in which in which the software tester has knowledge of the inner workings, structure and language of the software, or at least its purpose. It is purpose. It is used to test areas that cannot be reached from a black box level.

**Black Box Testing**

Black Box Testing is testing the software without any knowledge of the inner workings, structure or language of the module being tested. Black box tests, as most other kinds of tests, must be written from a definitive source document, such as specification or requirements document, such as specification or requirements document. It is a testing in which the software under test is treated, as a black box .you cannot “see” into it. The test provides inputs and responds to outputs without considering how the software works.

**6.2 Unit Testing:**

Unit testing is usually conducted as part of a combined code and unit test phase of the software lifecycle, although it is not uncommon for coding and unit testing to be conducted as two distinct phases.

**Test strategy and approach**

Field testing will be performed manually and functional tests will be written in detail.

**Test objectives**

* All field entries must work properly.
* Pages must be activated from the identified link.
* The entry screen, messages and responses must not be delayed.

**Features to be tested**

* Verify that the entries are of the correct format
* No duplicate entries should be allowed

All links should take the user to the correct page.

# 6.3 Integration Testing

Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects.

The task of the integration test is to check that components or software applications, e.g. components in a software system or – one step up – software applications at the company level – interact without error.

**Test Results:** All the test cases mentioned above passed successfully. No defects encountered.

**6.4 Acceptance Testing**

User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements.

**Test Results:** All the test cases mentioned above passed successfully. No defects encountered.

**Test Case Reports:**

**Test1:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| TEST CASES | EXPECTED OUTPUT | ACTUAL OUTPUT | | CONDITION |
| Registration | Registration Successfully | Registration Successfully | | Pass |
| Login | Login Successfully | Login Successfully | | Pass |
| User Profile Update | Profile Updated Successfully | Profile Updated Successfully | | Pass |
| Search | Search successfully | | Bank Added successfully | Pass |

T**able: 6.7.1 Positive Test Cases for Web Crawl in .NET Documents**

**Test2**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| TEST CASES | EXPECTED OUTPUT | ACTUAL OUTPUT | | CONDITION |
| Registration | Registration Successfully | Registration Failed | | Fail |
| Login | Login Successfully | Login Failed | | Fail |
| Update Use Profile | Profile Updated Successfully | Profile Updated Failed | | Fail |
| Search | Search successfully | | Search Failed | Fail |

**Table: 6.7.2 Negative Test Cases for a Web Craw in .NET Documents**

**CHAPTER-8**

**CONCLUSION**

In this paper, we proposed and implemented FOCUS, a supervised forum crawler. We reduced the forum crawling problem to a URL type recognition problem and showed how to leverage implicit navigation paths of forums, i.e. entry-index-thread (EIT) path, and designed methods to learn ITF regexes explicitly. Experimental results on 160 forum sites each powered by a different forum software package confirm that Focus could effectively learn knowledge of EIT path and ITF regexes from as few as 5 annotated forums. We also showed that FoCUS can effectively apply learned forum crawling knowledge on 160 unseen forums to automatically collect index URL, thread URL, and page-flipping URL string training sets and learn the ITF regexes from the training sets. These learned regexes could be applied directly in online crawling. Training and testing on the basis of forum package makes our experiments manageable and our results applicable to many forum sites. Moreover, FoCUS can start from any page of a forum, while all previous works expect an entry page is given. Our test results on 9 unseen forums show that FOCUS is indeed very effective and efficient and outperforms the state-of-the-art forum crawler, iRobot. The results on 160 forums show that FOCUS can apply the learned knowledge to a large set of unseen forums and still achieve a very good performance. Though, the method introduced in this paper is targeted at forum crawling, the implicit EIT-like path also apply to other sites, such as community Q&A sites, blog sites, and so on. In the future, we would like to handle forums which use JavaScript, include incremental crawling, and discover new threads and refresh crawled threads in a timely manner. The initial results of applying FOCUS-like crawler to other social media are very promising. We would like to conduct more comprehensive experiments to further verify our approach and improve upon it.

**CHAPTER-9**

**FUTURE SCOPE AND ENHANCEMENTS**

**FUTURE SCOPE:**

In this research work, various challenges in the area of Hidden web data extraction and their possible solutions have been discussed. Although this system extracts, collects and integrates the data from various hidden websites successfully, this work could be extended in near future. In this work, a search engine shell has been created which was tested on a particular domain. This work could be extended for other domains by integrating this work with the unified search interface.

**10 BIBILIOGRAPHY**

REFERENCES

1. CISCO, “Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update ,2011-2016,” Tech. Rep., 2012.
2. Y. Li, Y. Zhang, and R. Yuan, “Measurement and Analysis of a Large Scale CommercialMobile Internet TV System,” in ACM IMC, pp. 209–224, 2011.
3. T. Taleb and K. Hashimoto, “MS2: A Novel Multi-Source Mobile-Streaming Architecture,”in IEEE Transaction on Broadcasting, vol. 57, no. 3, pp. 662–673, 2011.
4. X. Wang, S. Kim, T. Kwon, H. Kim, Y. Choi, “Unveiling the BitTorrent Performance in Mobile WiMAX Networks,” in Passive and Active Measurement Conference, 2011.
5. A. Nafaa, T. Taleb, and L. Murphy, “Forward Error Correction Adaptation Strategies for Media Streaming over Wireless Networks,” in IEEE Communications Magazine, vol. 46, no. 1, pp. 72–79, 2008.